UNIVERSIDAD POLITÉCNICA DE MADRID
FACULTAD DE INFORMÁTICA

# Una Propuesta de Modelado del Estudiante Basada en Ontologías y Diagnóstico Pedagógico-Cognitivo no Monótono

Tesis doctoral

Julia Clemente Párraga
Licenciada en Informática

Madrid, noviembre de 2010

*A José Enrique, José María y Ángel, mis tres luces*

*A mis padres*

*Veritas filia temporis*
Aulus Gellius (125 a. C.)

# Agradecimientos

Queda

**Resumen**

Los recientes avances tecnológicos han encontrado un potencial campo de explotación en la educación asistida por computador. A finales de los años 90 surgió un nuevo campo de investigación denominado Entornos Virtuales Inteligentes para el Entrenamiento y/o Enseñanza (EVIEs), que combinan dos áreas de gran complejidad: Los Entornos Virtuales (EVs) y los Sistemas de Tutoría Inteligente (STIs). De este modo, los beneficios de los entornos 3D (simulación de entornos de alto riesgo o entornos de difícil uso, etc.) pueden combinarse con aquéllos de un STIs (personalización de materias y presentaciones, adaptación de la estrategia de tutoría a las necesidades del estudiante, etc.) para proporcionar soluciones educativas/de entrenamiento con valores añadidos.

El Modelo del Estudiante, núcleo de un SIT, representa el conocimiento y características del estudiante, y refleja el proceso de razonamiento del estudiante. Su complejidad es incluso superior cuando los STIs se aplican a EVs porque las nuevas posibilidades de interacción proporcionadas por estos entornos deben considerarse como nuevos elementos de información clave para el modelado del estudiante, incidiendo en todo el proceso educativo: el camino seguido por el estudiante durante su navegación a través de escenarios 3D; el comportamiento no verbal tal como la dirección de la mirada; nuevos tipos de pistas e instrucciones que el módulo de tutoría puede proporcionar al estudiante; nuevos tipos de preguntas que el estudiante puede formular, etc. Por consiguiente, es necesario que la estructura de los STIs, embebida en el EVIE, se enriquezca con estos aspectos, mientras mantiene una estructura clara, estructurada, y bien definida.

La mayoría de las aproximaciones al Modelo del Estudiante en STIs y en IVETs no consideran una taxonomía de posibles conocimientos acerca del estudiante suficientemente completa. Además, la mayoría de ellas sólo tienen validez en ciertos dominios o es difícil su adaptación a diferentes STIs. Para vencer estas limitaciones, hemos propuesto, en el marco de esta tesis doctoral, un nuevo mecanismo de Modelado del Estudiante basado en la Ingeniería Ontológica e inspirado en principios pedagógicos, con un modelo de datos sobre el estudiante amplio y flexible que facilita su adaptación y extensión para diferentes STIs y aplicaciones de aprendizaje, además de un método de diagnóstico con capacidades de razonamiento no monótono. El método de diagnóstico es capaz de inferir el estado de los objetivos de aprendizaje contenidos en el SIT y, a partir de él, el estado de los conocimientos del estudiante durante su proceso de aprendizaje.

La aproximación al modelado del estudiante propuesta ha sido implementada e integrada en un agente software (el agente de modelado del estudiante) dentro de una plataforma software existente para el desarrollo de EVIEs denominada MAEVIF. Esta plataforma ha sido diseñada para ser fácilmente configurable para diferentes aplicaciones de aprendizaje.

El modelado del estudiante presentado ha sido implementado e instanciado para dos tipos de entornos de aprendizaje: uno para aprendizaje del uso de interfaces gráficas de usuario en una aplicación software y para un Entorno Virtual para entrenamiento procedimental. Además, se ha desarrollado una metodología para guiar en la aplicación del esta aproximación de modelado del estudiante a cada sistema concreto.

**Abstract**

Recent technological advances have found a potential field of exploitation in computer-aided education. At the end of the 90's a new research field emerged, the so-called Intelligent Virtual Environments for Training and/or Education (IVETs), which combines two areas of great complexity: Virtual Environments (VE) and Intelligent Tutoring Systems (ITS). In this way, the benefits of 3D environments (simulation of high risk or difficult-to-use environments, etc.) may be combined with those of an ITS (content and presentation customization, adaptation of the tutoring strategy to the student requirements, etc.) in order to provide added value educational/training solutions.

The Student Model, core of an ITS, represents the student's knowledge and characteristics, and reflects the student's reasoning process. Its complexity is even higher when the ITSs are applied on VEs because the new interaction possibilities offered by these environments must be considered as new key information pieces for student modelling, impacting all the educational process: the path followed by the student during their navigation through 3D scenarios; non-verbal behavior such as gaze direction; new types of hints or instructions that the tutoring module can provide to the student; new question types that the student can ask, etc. Thus, it is necessary for the ITS structure, which is embedded in the IVET, to be enriched by these aspects, while keeping a clear, structured and well defined architecture.

Most approaches to SM on ITSs and IVETs don't consider a complete enough taxonomy of possible knowledge about the student. In addition, most of them have validity only in certain domains or they are hard to be adapted for different ITSs. In order to overcome these limitations, we have proposed, in the framework of this doctoral research project, a new Student Modeling mechanism that is based on Ontological Engineering and inspired on pedagogical principles, with a wide and flexible data model about the student that facilitates its adaptation and extension to different ITSs and learning applications, as well as a rich diagnosis method with non-monotonic reasoning capacities. The diagnosis method is able to infer the state of the learning objectives encompassed by the ITS and, from it, the student's knowledge state during the student's process of learning.

The proposed student modelling approach has been implemented and integrated in a software agent (the student modeling agent) within an existing software platform for the development of IVETs called MAEVIF. This platform was designed to be easily configurable for different learning applications.

The proposed student modeling has been implemented and it has been instantiated for two types of learning environments: one for learning to use the graphical user interface of a software application and a Virtual Environment for procedural training. In addition, a methodology to guide on the application of this student modeling approach to each specific system has been developed.

# ÍNDICE GENERAL

# ÍNDICE DE FIGURAS

# ÍNDICE DE TABLAS

CAPÍTULO 1

# INTRODUCCIÓN

La unión entre los campos de la Educación y la Informática ha llevado a la construcción de un conjunto de sistemas y aplicaciones denominados genéricamente Sistemas de Enseñanza Asistida por Ordenador (*Computer Assisted Instruction*, CAI). Estos sistemas han ido evolucionando notablemente. En primer lugar, aparecieron los programas lineales en los años 50. Estos sistemas, basados en la teoría conductista (Skinner (1950)), mostraban el conocimiento de una forma lineal; ningún factor como, por ejemplo, la aptitud del alumno podía alterar el orden de enseñanza establecido en su programación. Los sucesores de estos sistemas fueron los programas ramificados (Crowder (1959)) que se diferenciaba en su capacidad de actuar según la respuesta del alumno, aunque seguían presentando un número fijo de temas. A continuación, a finales de los sesenta y principios de los setenta, surgieron los sistemas generativos o adaptativos. La filosofía educativa en la que se basaban estos sistemas es distinta; los alumnos aprenden mejor enfrentándose a problemas de dificultad adecuada que atendiendo a explicaciones sistemáticas. Son sistemas que son capaces de generar un problema de acuerdo al nivel de conocimientos del alumno, construir una solución al problema y analizar la respuesta del alumno en base a esa solución. Sin embargo, la solución a un problema, en general, puede no ser única. Además, son sistemas que son aplicables fácilmente a dominios como la aritmética, pero no a otros dominios de enseñanza en los que la dificultad para generar problemas sea mayor.

Aunque ya en 1970 aparece el que es considerado por muchos investigadores como el primer artículo sobre SITs (Carbonell (1970)), es a comienzos de los ochenta (Sleeman and Brown (1982)) cuando se empiezan a distinguir estos sistemas de los anteriores. Los SITs son una evolución de los sistemas CAI en el sentido de que incorporan técnicas de Inteligencia Artificial. Son sistemas que se distinguen de los sistemas anteriores fundamentalmente en que la secuencia de enseñanza en ellos no está predeterminada por el diseñador del sistema sino que proporcionan al estudiante bastante libertad a la hora de seleccionar los temas formativos y los problemas, y conseguir una apropiada valoración de sus conocimientos y habilidades. Además, a diferencia de los sistemas de enseñanza tradicionales, los SITs se caracterizan por manejar separadamente la materia que se enseña y las estrategias que controlan la interacción con el estudiante.

## 1.1 Sistemas Inteligentes de Tutoría

En la literatura comparada se pueden encontrar algunas definiciones de SIT, como las que se presentan a continuación:

Woolf (1984) define los SITs como *Sistemas que modelan la enseñanza, el aprendizaje, la comunicación y el dominio del conocimiento del experto y el entendimiento del estudiante sobre ese dominio.*

Según VanLehn (1988) *Un SIT es un sistema software que utiliza técnicas de Inteligencia Artificial (IA) para representar el conocimiento e interactúa con los estudiantes para enseñárselo .*

Giraffa et al. (1997) delimita los SITs como *Sistemas que incorporan técnicas de IA (Inteligencia Artificial) a fin de crear un ambiente que considere los diversos estilos cognitivos de los alumnos que utilizan el programa.*

Desde el punto de vista de la autora de este trabajo, todas estas definiciones expuestas, así como algunas otras no incluidas aquí, carecen de precisión a la hora de definir los objetivos de los SITs, su esencia e, incluso, sus características, por lo que se hace necesaria una definición que clarifique adecuadamente la verdadera naturaleza de estos sistemas. Por todo ello, en este trabajo se propone la siguiente definición de SIT:

*Un SIT es un sistema software que, tratando de imitar la función de un tutor humano y adaptándose al comportamiento del estudiante, tiene como objetivo orientar y facilitar el proceso de su aprendizaje, ofreciéndole la ayuda que requiera en todo momento.*

Las características de un SIT, esbozadas en la definición previa, dotan al sistema de un conjunto de capacidades fundamentales como las siguientes:

- El sistema es capaz de acomodarse al aprendizaje individual (*aprendizaje a medida*).
- Permite una interacción con el estudiante mediante preguntas y respuestas.
- Es capaz de detectar conceptos que el estudiante no tiene claros o son erróneos, partiendo de los errores que comete, modelando al estudiante e interpretando su conocimiento en términos del conocimiento del experto.

Para lograr su objetivo, los SITs requieren, por un lado, de capacidad de tutoría que permita al sistema decidir cuándo hacer una sugerencia o cuándo dar explicaciones adecuadas y, por otro lado, de un conocimiento profundo en el dominio de la materia que se enseña, para que el sistema pueda responder a las posibles preguntas planteadas por el estudiante y seguirle la pista en su modo de resolución del problema; además, el sistema debe tener una presentación amena y estimulante de la materia o temas que se traten, y un buen tiempo de respuesta, todo ello con el objetivo de evitar que el estudiante se distraiga.

### 1.1.1 Arquitectura de un SIT

Un SIT consta de cuatro módulos (Wenger (1987)) que interaccionan entre sí y que gestionan todos los aspectos mencionados previamente:

- *Módulo del experto.* Representa el conocimiento relevante en el dominio o materia que se enseña, y resuelve problemas, como lo haría un experto, a partir de una base de conocimientos que contiene lo que el estudiante va a aprender. Consiste en un modelo dinámico

de la materia a enseñar que permite generar distintas soluciones a un problema, en vez de tratar una solución prediseñada.

- *Modelo del estudiante.* Contiene información sobre el estudiante individual; de hecho, debe incluir información sobre las acciones del estudiante para construir un modelo del conocimiento del mismo y de su forma de aprendizaje. Es el componente del SIT que permite al sistema acomodarse al estudiante, suministrándole un aprendizaje a medida, así como decidir la mejor estrategia pedagógica a seguir en su formación. Para lograr este aprendizaje adaptable es necesario un mecanismo que haga uso de este modelo y que permita inferir el estado cognitivo del usuario (qué sabe el estudiante) a partir de su comportamiento. A este mecanismo se le denomina *Diagnóstico Cognitivo*. Precisamente hemos de advertir que es en este módulo del SIT en el que se centra el presente trabajo.

- *Módulo de Tutoría.* Es el componente del SIT que representa al profesor y debe ser capaz de aplicar las estrategias de tutoría apropiadas, y en los tiempos adecuados, modelando las propiedades deseables de un tutor humano. Por lo tanto, contiene las estrategias, reglas y procesos que dirigen las interacciones del sistema con el estudiante para tomar decisiones acerca del material que debe presentar, qué preguntas o ejemplos sugerir y, porqué y cuándo se debe interrumpir al estudiante.

- *Módulo de Comunicación.* Es el módulo del SIT que permite la comunicación entre el alumno y el sistema. Debe facilitar la eliminación de ambigüedades en las respuestas de los estudiantes para facilitar la tarea del resto de componentes.

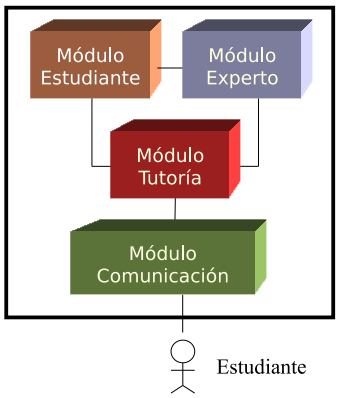


**Figura 1.1:** Arquitectura de un SIT

Una descripción más detallada de una posible arquitectura de un SIT se presenta en (Petrushin (1995)).

### 1.1.2 Clasificación de los SITs

Los SITs se pueden clasificar de acuerdo a sus objetivos operacionales en los siguientes tipos (Petrushin (1995)):

- *Sistemas de consulta.* Se diseñan para proporcionar ayuda al estudiante a través, bien de la respuesta a preguntas planteadas por el alumno (sistema llamado pregunta-respuesta),

bien mediante la solución a problemas propuestos por el sistema; tras esta información, el sistema puede también ofrecer una explicación, si ésta es solicitada por el estudiante.

- *Sistemas de diagnóstico.* Se articulan para indicar al estudiante nociones incorrectas sobre el área de estudio que le han llevado a cometer errores en el transcurso de la resolución de un tipo particular de problemas. Este tipo de sistemas consta de una interfaz, un sistema que resuelve problemas de un cierto tipo, un sistema de diagnóstico de errores y un modelo del estudiante. Pueden considerarse como un complemento de un SIT de consulta. A estos sistemas se les suele denominar sistemas inteligentes de ejercitación, puesto que se usan para ofrecer al estudiante ejercicios de resolución de problemas cuando el profesor, o el propio sistema, genera una secuencia de problemas partiendo de un conjunto de parámetros especificados por el referido profesor o por el mismo estudiante.

- *Sistemas de control.* Se utilizan para guiar la actividad cognitiva del estudiante. Este tipo de sistemas constituyen una extensión de un SIT de diagnóstico, incluyendo el conocimiento relacionado con los objetivos de funcionamiento del sistema presentes en el material de formación y las estrategias de enseñanza. Se distingue entre la enseñanza de conceptos y la enseñanza de habilidades.

- *Sistemas de supervisión.* Se diseñan para seguir la pista de la actividad del usuario en alguna tarea y proporcionarle ayuda cuando se detecten acciones incorrectas o irracionales. Este tipo de sistemas intentan comprender lo que el usuario hace, evalúan cómo están realizando una acción particular y deciden si se necesita o no ayuda y cómo prestársela. Se diferencian de los SITs de control en que no conocen el objetivo de la actividad del usuario y son menos interactivos para evitar atraer la atención del usuario más allá del problema particular. Un SIT de este tipo puede considerarse como una extensión de un SIT de diagnóstico.

En esta clasificación, los dos últimos tipos de SITs son los más complejos y se suelen denominar tutores inteligentes o sistemas expertos de tutoría.

Es importante destacar de esta posible clasificación que no todos los SITs incluyen los cuatro componentes mencionados en la sección 1.1.1 e incluso, dependiendo de las características de un SIT concreto, cada componente puede presentarse con diferentes niveles de complejidad.

Íntimamente relacionado con lo anterior, es el enfoque adoptado por el sistema que implica adaptar alguno de sus componentes en base a ello. revista iberoamericana

## 1.2 Objetivos y tareas

El análisis exhaustivo del estado del arte de los MEs (sección 2), nos ha llevado a la conclusión de que la mayoría de las aproximaciones que han sido propuestas hasta el momento para el ME, aunque representan diferentes tipos de información del estudiante y usan diferentes métodos para inferir el estado cognitivo del estudiante, se caracterizan en general por lo siguiente:

- Presentan una taxonomía parcial de conocimiento acerca del estudiante. La mayoría de las propuestas existentes no poseen un modelo de datos rico y amplio sobre los diversos tipos de conocimientos del estudiante que permita obtener información crucial sobre su estado de conocimiento, las trazas de su comportamiento durante diferentes sesiones de aprendizaje, etc.

- Son modelos válidos sólo para ciertos dominios o es difícil su adaptación para ser aplicados a diferentes SITs.
- No consideran las características individuales del estudiante con suficiente nivel de detalle.
- No tratan el conocimiento incompleto del estudiante. El ME razona a partir de la observación del comportamiento del alumno frente a los problemas que se le plantean y, también, a partir del estado de su conocimiento que es, probablemente, parcialmente desconocido en un determinado instante. En estos escenarios, el SIT no puede detener el proceso de enseñanza, sino que debería tratar este conocimiento incompleto para permitir al método de diagnóstico del ME seguir avanzando adecuadamente en su proceso de razonamiento y, en consecuencia, seguir proporcionando información sobre el estado de los conocimientos del estudiante al Módulo de Tutoría a fin de que realice eficazmente su tarea.

Con el objetivo de abordar estas limitaciones, el presente trabajo se propone, como objetivo principal:

> *Desarrollar un nuevo modelo del estudiante basado en un enfoque pedagógico, que se caracterice por ser fácilmente adaptable, extensible, y reutilizable para su aplicación a diferentes tipos de SITs, y con un método de diagnóstico pedagógico-cognitivo con capacidades de razonamiento no monótono.*

Del objetivo general enunciado se han extraído los siguientes objetivos específicos que se detallan en la sección 4.1.1:

- *Desarrollar un ME con una amplia taxonomía de conocimiento que posibilite expresar muchos tipos de conocimientos sobre el estudiante en un ME potente.*
- *Desarrollar un ME con un potente formalismo de representación* que permita una representación de los conceptos con diferentes niveles de abstracción, y que sirva de soporte para compartir y reutilizar conocimiento. El uso de ontologías ha ayudado a alcanzar esta meta.
- *Desarrollar un nuevo método de diagnóstico pedagógico-cognitivo.*

  En este trabajo se ha considerado, como parte clave del diagnóstico cognitivo, lo que denominamos diagnóstico pedagógico, y que definiremos de la forma siguiente:

  > *Mecanismo capaz de inferir el estado de los objetivos de aprendizaje, alcanzados o no por el estudiante, a partir de su comportamiento.*

  El ME a desarrollar debe ser capaz de obtener un diagnóstico pedagógico y, a partir de él, debe ser capaz también de proporcionar un diagnóstico cognitivo, definido de la forma siguiente:

  > *El diagnóstico cognitivo es el mecanismo capaz de inferir el estado de los conocimientos del estudiante a partir de su comportamiento.*

- *Desarrollar un nuevo método de diagnóstico con capacidades de razonamiento no monótono* acorde con la naturaleza, también no monótona, del razonamiento humano.

- *Proponer un ME flexible y fácilmente adaptable* especializándolo, en concreto, para su uso en Entornos Virtuales de Aprendizaje y, más en concreto, en Entornos Virtuales de Entrenamiento/Formación procedimental.

Los objetivos mencionados anteriormente serán alcanzados mediante la realización de las siguientes tareas de investigación:

- T1. Identificar y describir la metodología de investigación usada en esta tesis.
- T2. Identificar y analizar las diferentes aproximaciones al modelado del estudiante.
  - T2.1. Identificar y analizar las taxonomías de los conocimientos del estudiante utilizadas en los Modelos del Estudiante propuestos hasta el momento actual.
  - T2.2. Identificar y analizar los métodos de diagnóstico cognitivo presentes en los MEs.
- T3. Definir el enfoque de la solución propuesta en este trabajo, y diseñar a alto nivel la arquitectura.

  Definir teóricamente el enfoque propuesto para resolver el problema del Modelado del Estudiante, y diseñar a alto nivel la arquitectura. El enfoque está basado en el diseño pedagógico y soportado por ontologías como formalismo de representación de los conocimientos en el ME.
- T4. Desarrollar la ontología del estudiante incluida en el ME.

  La ontología del ME contendrá los diferentes tipos de conocimientos del estudiante considerados en la aproximación propuesta. Será necesario identificar los métodos, lenguajes, herramientas, y la propia metodología de construcción utilizados en el desarrollo de esta ontología.
- T5. Especificar los tipos de entornos de aprendizaje a considerar.

  Describir el tipo de entornos de aprendizaje en los que es aplicable la solución en base a la arquitectura propuesta.
- T6. Diseñar los prototipos de demostración.

  Diseñar varios ejemplos de prototipos (consistentes con lo especificado en la tarea previa) para realizar las pruebas.
- T7. Diseñar el método de diagnóstico basado en razonamiento no monótono.
- T8. Desarrollar una jerarquía de reglas de diagnóstico del ME.
- T9. Implementar los componentes específicos del diagnóstico.
- T10. Crear una metodología de aplicación del modelo a un sistema específico.

  Describir la metodología a seguir para adaptar el Modelado del Estudiante propuesto en su aplicación a un sistema específico.
- T11. Integrar los componentes específicos del diagnóstico con el resto de componentes del ME.
- T12. Evaluar el funcionamiento de la arquitectura.

## 1.3 Estructura del trabajo

La tesis que se presenta se ha estructurado en diez capítulos. Se inicia con la introducción al área de estudio tratada y una presentación de los SITs y, en concreto, del Modelado del Estudiante, donde se han descrito cuáles son los objetivos fundamentales a alcanzar con este trabajo. Después se describirá el estado del arte del modelado del estudiante y se hará un análisis minucioso de las aproximaciones que se han considerado más interesantes, hasta la actualidad, relativas al ME y al diagnóstico cognitivo subyacente (capítulo 2).

A continuación, en el capítulo 3, se presentan los fundamentos generales, teóricos y tecnológicos, requeridos como soporte para la realización del trabajo, así como para una mejor comprensión del resto del documento: la técnica de razonamiento no monótono aplicada en el trabajo (ATMS), el diseño instruccional base del enfoque del trabajo, y una visión general de las ontologías cuyo análisis ha facilitado la selección final del lenguaje y herramientas que se han considerado más adecuadas para el desarrollo del mecanismo de modelado del estudiante.

Una vez analizadas previamente las ventajas y limitaciones de los MEs propuestos por otros autores, y tras haber plasmado las conclusiones críticas correspondientes, se presenta en el capítulo 4 el planteamiento concreto del problema a tratar y las hipótesis del trabajo (en este último aspecto se hace especial hincapié en las características o supuestos relativos a los entornos de aprendizaje a los que va dirigida la descripción de la solución adoptada en el trabajo), para después, en el capítulo 5, abordarse la solución adoptada, centrándose en las ontologías propuestas para el Modelo del Estudiante, en el método de diagnóstico pedagógico propuesto y en la metodología a seguir para la aplicación de la solución a un sistema específico.

En el capítulo 6 se muestra la evaluación realizada para validar el funcionamiento de la arquitectura propuesta y el método de diagnóstico desarrollado, para describir, en el capítulo 7, las conclusiones obtenidas en todas las vertientes de esta tesis y las principales aportaciones de la solución.

Para concluir, el capítulo 8 esbozará las posibilidades que, a juicio de la autora, abre el presente trabajo, finalizando con las referencias bibliográficas utilizadas.

CAPÍTULO 2

# ESTADO DE LA CUESTIÓN

EL ME, núcleo de los SITs, y, en particular, el proceso de diagnóstico cognitivo inherente a este modelo, ha sido siempre una de las más importantes líneas de investigación en el área de los SITs, debido a su complejidad. La dificultad en resolver este problema consiste en proporcionar una respuesta eficiente a preguntas claves tales como las siguientes: ¿Qué tipos de conocimientos acerca del estudiante deberían sustentar el ME para que la tutoría pueda adaptarse a las características individuales actuales del estudiante y el proceso de diagnóstico pueda proporcionar información más completa sobre el estado cognitivo del estudiante?, ¿Qué características deberían tener los mecanismos de modelado del estudiante para que puedan ser aplicados a diferentes dominios?, ¿Cómo gestionar en el proceso de diagnóstico la existencia de posibles inconsistencias que puedan surgir en el diagnóstico cognitivo asociado a un estudiante a lo largo de su aprendizaje?, ¿Cómo resolver el diagnóstico para que, no sólo permita detectar el estado del conocimiento del estudiante sino también para que sirva como soporte esencial para que el tutor guíe adecuadamente a cada estudiante durante su aprendizaje?, etc.

Se han propuesto numerosas aproximaciones al ME en el campo de los SITs. Para desarrollar un mecanismo de modelado del estudiante que dé una respuesta satisfactoria a las preguntas anteriores, es necesario analizar previamente cómo las propuestas en este área han abordado estas cuestiones, para así llegar más lejos que ellas superando algunas de sus limitaciones. Por ello, a continuación se describen algunas de las aproximaciones de mayor interés en el campo de los SITs utilizando como criterio de clasificación el tipo de información que representan. Posteriormente, el análisis del estado de la cuestión se centra en taxonomías de conocimiento que merecen ser destacadas por su interesante contribución a este campo y en los métodos de diagnóstico cognitivo, cuya evolución ha venido forzada por el propio avance en el campo de la IA. Finalmente, se presentan algunos ejemplos de Entornos Virtuales Inteligentes para la Formación/Entrenamiento, centrando la atención en los SITs embebidos en ellos y, más en concreto, en el modelado del estudiante que realizan, foco de interés de este trabajo.

## 2.1 Modelado del estudiante en Sistemas Inteligentes de Tutoría

En el aprendizaje asistido por ordenador tradicional no se trata en profundidad el problema de modelar al estudiante. Se le representa mediante datos sin procesar o sin estructurar, tales como resultados cuantitativos en tests o juicios binarios sobre respuestas. Estos datos no soportarían ninguna inferencia compleja sobre el estado actual del estudiante sin un procesamiento adicional considerable mediante un componente de tutoría.

En los SITs, los modelos del estudiante han representado diversos tipos de información (Petrushin (1995)) y han usado diferentes métodos y técnicas para inferir el estado del estudiante (Gonzalez et al. (2006)). La mayoría de los SITs almacenan conocimiento de dominio mediante un modelo Superpuesto (Carr and Coldstein (1977)) o mediante reglas erróneas (Burton and Brown (1976)). Ambos tipos y ejemplos se describen en los siguientes apartados. Algunos SITs representan al estudiante como un subconjunto de un modelo cognitivo para el dominio (por ejemplo, el tutor LISP (Farrell et al. (1984), Anderson et al. (1990)).

Pocos trabajos se han hecho para representar de forma conjunta las características individuales del estudiante (estilo de aprendizaje, estado afectivo, capacidad de aprendizaje/inteligencia o diversos atributos individuales del estudiante) que permitan al sistema encaminar sus acciones en función de su conocimiento acerca de las necesidades particulares del estudiante. En esta línea, caben destacar los trabajos dirigidos a extender la definición de modelo mental (Norman (1987)) como los de Priest and Young (1988), o White et al. (1990) o el tratamiento de variables psicológicas tales como la motivación del estudiante por Soldato (1992). Asimismo, se han tratado específicamente los estilos de aprendizaje del estudiante. Por ejemplo, basados en el modelo de estilos de aprendizaje definido por Felder and Sylverman (Felder and Silverman, han surgido variadas e interesantes propuestas, como la incluida en la ontología de dominio que describe el material de aprendizaje de un curso de Gascueña et al. (2006) o, el modelado de estilos de aprendizaje con redes bayesianas de Carmona et al. (2008). En cuanto a la consideración del estado afectivo del estudiante, destaca también el modelo de comportamiento afectivo (ABM) para SITs de Hernández et al. (2008).

Otras variables, como son la capacidad de aprendizaje/inteligencia, estrategias de aprendizaje múltiples y variables no han sido aún tratadas en la mayoría de los sistemas existentes. Las limitaciones en este aspecto se han debido fundamentalmente a la carencia de medios estandarizados para determinar fácilmente el estado del estudiante no sólo en cuanto al conocimiento del dominio del experto y conceptos erróneos comunes sino también a través de sus características individuales.

Además, es importante señalar que la mayoría de los trabajos relacionados con el ME se refieren a la representación del conocimiento y procesos cognitivos del estudiante. Sin embargo, parece cada vez más unánime la creencia dentro de la comunidad de SITs de que la formación debería tener en cuenta el proceso metacognitivo del estudiante, es decir, lo que el estudiante sabe acerca de su propio conocimiento y proceso cognitivo. No sólo es deseable a menudo la formación de habilidades cognitivas sino, además, enseñar habilidades metacognitivas. Derry y sus colegas desarrollaron un sistema educativo, TAPS [DERR92] para problemas de expresiones aritméticas que hace hincapié en la adquisición de habilidades metacognitivas.

De acuerdo al tipo de información representada en un ME, éstos se pueden clasificar de la siguiente forma:

- Representación de información cuantitativa (de la materia que se pretende enseñar) o información de estado.

- Representación de información cualitativa o información del proceso de razonamiento del estudiante.

### 2.1.1 MEs con representación de la información de estado

Se pueden considerar dos tipos de ME con representación de la información de estado:

- MEs que representan sólo conocimiento de estado correcto, tales como: *Modelo Superpuesto, Modelo Diferencial* y *Modelo de Grafo Genético*, y
- MEs que representa conocimiento de estado correcto e incorrecto, tales como: *Modelo de Perturbación* y *Modelo de Errores*

#### 2.1.1.1 Modelo Superpuesto

En estos sistemas, el conocimiento del estudiante se representa sólo en términos de conocimiento "correcto" y es tratado como un subconjunto del conocimiento del experto. El objetivo de la formación es establecer la correspondencia más cerrada posible entre los dos conjuntos de conocimientos. Un ME basado en tal aproximación se denomina *Modelo Superpuesto* (MS).

En un MS, el ME es conceptualizado comparando el comportamiento del estudiante con el de un experto. Esta aproximación asume que todas las diferencias entre el comportamiento del estudiante y el del experto pueden explicarse como falta de habilidades o destrezas del estudiante (ver figura 2.1).

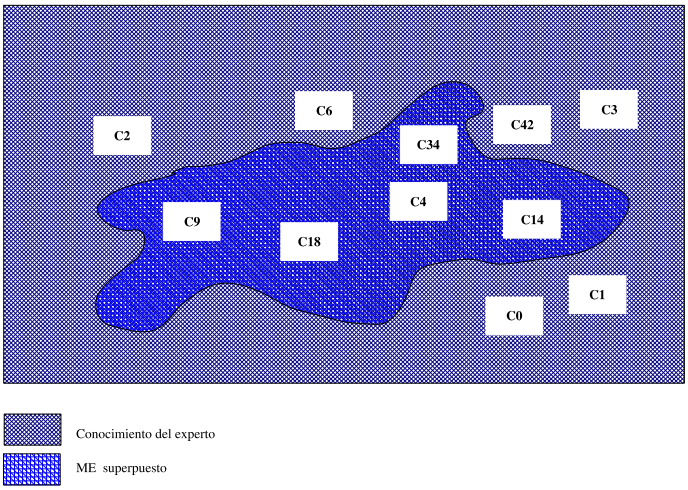


**Figura 2.1:** ME superpuesto

En un MS, el estudiante se representa mediante un mecanismo relativamente sencillo que soporta la inferencia acerca del estado cognitivo del estudiante relativo al experto ideal del dominio.

El principal inconveniente de este tipo de modelo del estudiante es que funciona bien sólo para sistemas donde el objetivo es estrictamente transmitir el conocimiento del experto al estudiante. Este modelo asume que el conocimiento del estudiante sólo puede ser un subconjunto del conocimiento de un experto, pero puede no ser éste el caso. El estudiante quizás posea conocimiento (posiblemente equivocado) que los expertos no tienen e incluso, si es un estudiante

novato, usar aproximaciones para resolver problemas diferentes de las utilizadas por los expertos. El MS no contempla ningún conocimiento o creencia que el estudiante pudiera tener y que difiera de los del experto. Este fallo ha llevado a una variedad de extensiones para el MS, como se verá más adelante.

Ejemplos destacados de sistemas que usan un MS en su aproximación al ME son:

- *SCHOLAR* [CARB70]: es un tutor de geografía considerado como uno de los primeros ejemplos de aplicación de las técnicas de IA a la formación asistida por computador (CAI). Usa una red semántica para representar el conocimiento de dominio (la geografía de Sudamérica) así como para el ME. La red está compuesta por unidades individuales. Estas unidades o nodos están formadas por propiedades (por ejemplo, nombre de la unidad) que constan a su vez de un nombre, conjunto de etiquetas y valor. Cada valor puede hacer referencia a otras unidades de la red.

  Esta aplicación soportaba diálogos combinados con iniciativas del propio estudiante. No sólo el computador preguntaba al estudiante sino que éste también podía, teóricamente, plantear preguntas al computador. El proceso de evaluar el funcionamiento del estudiante se realizaba por un generador de respuestas correctas a partir de la red semántica. El generador compara la respuesta con la dada por el estudiante y un proceso posterior de análisis de error se encarga de intentar separar en la respuesta el aspecto correcto del incorrecto para así obtener un diagnóstico. Para ello, se desarrolló una taxonomía de errores (error de pérdida de información, error por falta de un concepto, error por hechos mal archivados o memorizados, etc.). Una limitación fundamental de dicho sistema viene dada por la propia representación semántica que dificulta el representar conocimiento de procedimiento.

- *BIP* (Barr (1975)): es un laboratorio tutorial de resolución de problemas de programación en BASIC. Este programa desarrollaba habilidades de procedimiento requeridas en el aprendizaje del lenguaje de programación BASIC. Para lograr este objetivo ([SHUT94]), el programa seleccionaba problemas basándose en lo que el estudiante ya conoce (funcionamiento pasado), qué habilidades deberían ser a continuación enseñadas, y el análisis de dichas habilidades requeridas (problemas en el currículo). Los ejercicios eran dinámica e individualmente seleccionados por persona (de una colección de 100 problemas ejemplo). Se aplican heurísticas al ME para identificar las habilidades a enseñar, seleccionándose los ejercicios que más se adecuaban a esas habilidades. Esta selección se basaba en la información contenida en una red llamada CIN (*C*urrículo *I*nformation *N*etwork) que relacionaba tareas en el currículo con temas en el conocimiento de dominio. De este modo, una tarea de programación en el tutor se representaba en términos de sus componentes de requerimientos de habilidades. Basándose en una tarea de análisis, BIP sabía qué componentes de habilidades necesitaba para resolver un problema de programación particular, problema que incluía habilidades tales como: iniciar variables numéricas, usar bucles *for* con literal como valor final, etc.

- *GUIDON* [CLAN83]: un tutorial construido sobre el sistema experto de diagnóstico médico *MYCIN* [SHOR76]. A los estudiantes que usan GUIDON se les proporcionaba una descripción inicial de los síntomas de un paciente que sufre algún caso de meningitis (el dominio de tutoría). El estudiante, a continuación, procedía a realizar preguntas para intentar desarrollar una hipótesis acerca de la forma particular de enfermedad que tenía el paciente. GUIDON observaba este diálogo y podía interrumpir para preguntar y lograr

que el estudiante clarificase su estrategia, aconsejarle, e incluso podía redirigir el diálogo (un tipo de tutor entrenador).

Una importante contribución del proyecto GUIDON es el descubrimiento de que Bases de Conocimientos (BCs) como MYCIN no son suficientes para proporcionar una tutoría inteligente. En particular, esto significa que una representación del conocimiento basada en reglas, por sí misma, no basta para asegurar un buen modelo experto de tutoría. Por ejemplo, las reglas de MYCIN eran muy superficiales, obviando muchos de los pasos intermedios que dan los expertos. Además, sus reglas no eran homogéneas, mezclaban varios tipos de conocimiento dispares, a menudo en las condiciones de la misma regla. Estos problemas llevaron a desarrollar una nueva representación de la base de conocimientos, denominada NEOMYCIN [CLAN83]. Esta es una segunda contribución de GUIDON, la presentación del modelo psicológico de cómo los expertos médicos realizaban el diagnóstico. El modelo demuestra la necesidad de separar varios tipos dispares de conocimiento, demostrando también así la necesidad de estructurar el conocimiento de hechos para complementar el conocimiento basado en reglas. Por lo tanto, el ME requerirá una representación del conocimiento muy diversa y sofisticada no una representación sencilla. Otra importante contribución de GUIDON es el uso de razonamiento basado en evidencias para construir el ME, lo que le proporciona tolerancia para ruido en los datos, un importante problema al modelar al estudiante.

Sin embargo, también hay críticas importantes a este sistema (Kass (1989)). En primer lugar, al usar un modelo superpuesto (o modelo diferencial en el caso de GUIDON2, el sucesor de GUIDON, que se verá más adelante) sin librería de errores, el ME está limitado a un subconjunto del modelo del experto. Esta suposición es demasiado fuerte, incluso para estudiantes de medicina con una buena formación en el dominio. En segundo lugar, GUIDON asume que sus estudiantes están completamente familiarizados con los conceptos, hechos y pruebas de laboratorio referenciados durante la tutoría. Así, la tutoría se limita al propio proceso de diagnóstico y no a conceptos objetivos en el dominio de diagnóstico. En tercer lugar, el sistema no puede desarrollar nuevas nociones de posibles MEs. GUIDON no puede descubrir que un estudiante está usando una regla que es desconocida por el sistema y recordar esta regla para modelados posteriores. Un buen sistema con esta capacidad podría reconocer cuándo un estudiante usa una nueva regla válida con la que el sistema no está familiarizado, añadiéndola a la base de conocimientos del experto, de tal modo que éste pueda aprender también. Finalmente, GUIDON no entiende de limitaciones de aprendizaje del estudiante (por ejemplo, que el estudiante después de una larga sesión de tutoría sea menos capaz de asimilar nuevo conocimiento). Este sistema no tiene un modelo para estudiantes que aprenden y olvidan, o para la actuación de un estudiante bajo diferentes cargas cognitivas.

- *WUSOR*: es un sistema para aprender estrategias de juego (en concreto, para el juego *Hunt the Wumpus*). La primera no usaba técnicas de modelado, la segunda (Carr and Coldstein (1977)) usaba un modelo superpuesto sencillo y la tercera, WUSOR-III ([GOLD82]), fue la versión más interesante con respecto al ME, por lo que nos centraremos más adelante en esta última.

#### 2.1.1.2 Modelo Diferencial

El Modelo Diferencial (MD) es una modificación del MS. Este modelo divide el conocimiento del estudiante en dos categorías: a) conocimiento que el estudiante debería conocer y, b) conocimiento que no se espera que el estudiante conozca. A diferencia del MS, el MD no asume que todas las "lagunas.en el Modelo del Estudiante son igualmente indeseables. El MD reconoce e intenta representar explícitamente tanto el conocimiento del estudiante como las diferencias estudiante-experto. El MD puede verse como un MS sobre el conocimiento esperado, que es básicamente un subconjunto del modelo general del experto (ver Figura 2.2).

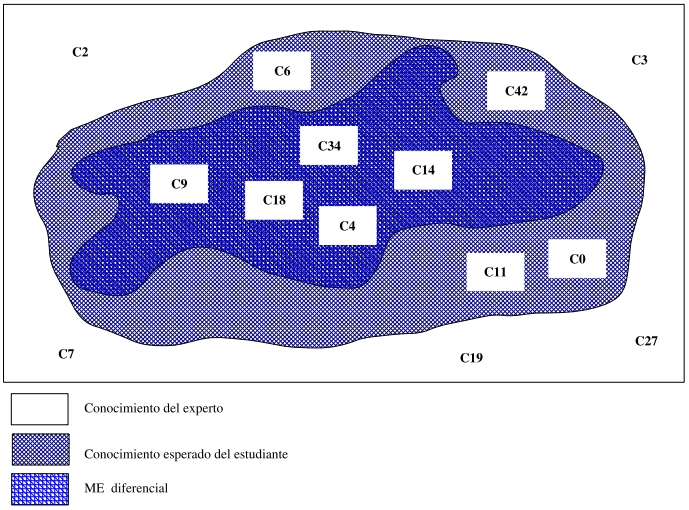


**Figura 2.2:** ME diferencial

Aunque el MD no es tan estricto acerca del conocimiento del estudiante, adolece de la mayor parte de las dificultades que presenta un MS, ya que asume que el ME es esencialmente un subconjunto del experto y, por lo tanto, sigue siendo incompleto.

Ejemplos destacados de sistemas que usan un MD en su aproximación al ME son:

- *WEST* (Burton and Brown (1976)): fue desarrollado para ayudar a los estudiantes en el aprendizaje/práctica de habilidades relacionadas con la manipulación de expresiones aritméticas. El objetivo de WEST consistía en moverse en un tablero de juego (*How the West Was Won*). No sólo están implicadas las técnicas aritméticas básicas sino que también se requieren conocimientos estratégicos. El tutor controlaba en segundo plano los movimientos del estudiante, interviniendo sólo cuando estaba claro que el estudiante vacilaba. En este caso, el tutor proporcionaba sugerencias para mejorar las habilidades del estudiante.

- *GUIDON2* (Clancey (1986)): es el sucesor de GUIDON. Usa una aproximación diferencial para modelar al estudiante. El modelo experto se usaba para generar un rango razonable de predicciones, el conjunto de acciones e hipótesis que GUIDON2 esperaba que el estudiante debería tener -porque son las que el experto también debería tener (London and Clancey (1982)). Normalmente, este rango de predicciones eran suficiente para modelar al estudiante. Si no, GUIDON2 comienza una búsqueda dirigida por los datos para encontrar un nuevo rango. El modelador se llama *IMAGE* en esta versión de GUIDON y contiene

un conjunto de reglas que actualizan el modelo después de cada paso en el diálogo. Estas reglas usan la misma forma de razonamiento basado en la evidencia que MYCIN, de tal modo que el modelador puede incluir grados de creencia para los elementos en el ME. Las ventajas e inconvenientes de este sistema se comentaron al describir su antecesor GUIDON.

#### 2.1.1.3 Modelo de Grafo Genético

Una elaboración del Modelo Superpuesto es el modelo de Grafo Genético (MGG), que usa un tipo de red semántica llamada *grafo genético*. Esta red contiene suposiciones acerca de las formas en que el estudiante desarrolla diversos aspectos de destreza ([GOLD82]). El conocimiento del estudiante se describe en términos de nodos del grafo. Estos nodos representan hechos, reglas o procedimientos que describen las potenciales habilidades del estudiante. Los nodos están conectados por arcos que describen su comportamiento en el aprendizaje. El progreso del estudiante se muestra por los posibles caminos a través del grafo y se asume que el estudiante progresará a través de un camino de aprendizaje particular en una secuencia general correspondiente a un orden parcial del grafo. El mecanismo de aprendizaje necesario (por ejemplo, generalización, discriminación, etc.) para moverse de un nodo al siguiente también forma parte del ME (Brecht and Jones (1988)).

Un ejemplo de esta aproximación es el sistema *WUSOR-III*.El modelado en *WUSOR-III* se enfocaba sobre cómo el conocimiento es adquirido por el estudiante. El conocimiento de cómo un estudiante aprende información en un dominio permite una rápida predicción de qué elementos aprenderá a continuación. Esto ayudaba no sólo a construir el modelo, sino que además ayudaba al tutor inteligente proporcionándole una indicación de los tópicos que serían apropiados para la ayuda y tópicos sobre los cuales el estudiante tiene tendencia a equivocarse en un punto dado de la sesión. El sistema consta de cinco módulos: el experto, ME, los componentes de tutoría, un módulo llamado el *psicólogo* responsable de actualizar el ME, y un quinto módulo, el *grafo genético*, una estructura de datos que contiene el conocimiento de dominio usado por el experto y representado como reglas. Además de este conocimiento, también se añaden varias formas de enlaces (cuatro tipos: generalización, analogía, refinamiento y desviación que relacionan las reglas. Estos enlaces representan la forma en que un individuo aprende nuevos conceptos a partir de los conceptos que ya conoce. De este modo, se propone el grafo genético como un modelo psicológico de cómo los estudiantes adquieren conocimiento del dominio.

WUSOR-III modela al estudiante como un par de especialistas, uno para resolver el problema y el otro para aprender. El ME se construye como un modelo superpuesto al grafo genético. El conocimiento del especialista que resuelve el problema es simplemente un modelo superpuesto del conocimiento de dominio, que constituye los nodos en el grafo. El especialista que aprende consiste en un conjunto de estrategias de aprendizaje de nuevo material. Los tipos de enlaces entre los nodos del grafo reflejan las estrategias de aprendizaje particular que pueden usarse para aprender información de dominio en el grafo. El ME también superpone estos enlaces en el grafo, indicando las estrategias de aprendizaje usadas por el estudiante en la adquisición de nueva información acerca del dominio. Inicialmente, el ME es un modelo superpuesto con unos pocos nodos en el grafo genético. Cuando el estudiante adquiera conocimiento nuevo, el modelo crecerá con los caminos de enlace entre el conocimiento previamente adquirido y el nuevo conocimiento. Entre las principales contribuciones de WUSOR al modelado del estudiante está la noción de grafo genético, que representa un modelo psicológico del estudiante, centrándose no sólo en el conocimiento del estudiante sino también en su aprendizaje. Una ventaja del grafo

genético es, además, su capacidad para predecir qué conocimiento el estudiante probablemente adquirirá a continuación. La frontera del grafo sirve como medio para limitar la búsqueda necesaria cuando se intenta reconocer el conocimiento de un estudiante así como para ayudar a seleccionar el material de tutoría (proporciona un conjunto de elementos que tienden a ser más relevantes y comprensibles para el estudiante en un punto concreto de la sesión). Finalmente, una ventaja potencial de la estructura del grafo genético es que puede usarse sobre él mismo. El dominio del grafo genético podrían ser las mismas estrategias de aprendizaje, de tal modo que un sistema podría no sólo realizar la tutoría sobre un dominio subyacente sino, además, sobre las habilidades de aprendizaje usadas para aprender acerca de tal dominio.

Sin embargo, también WUSOR tiene muchos inconvenientes. En primer lugar, el grafo genético está basado en un modelo superpuesto. La falta de completitud de tal aproximación limita enormemente el modelado de capacidades de WUSOR. En segundo lugar, la representación del conocimiento del dominio subyacente no es estructurada. WUSOR usa reglas de producción sencillas para representar su conocimiento. Se necesita una estructura profunda en el conocimiento de dominio así como técnicas de representación múltiples. En tercer lugar, WUSOR adolece del problema del ruido. El conocimiento actual que representa la frontera del ME es a menudo incierto, ya que los estudiantes adquieren nuevos conocimientos que pueden alternativamente usar correctamente y luego incorrectamente. Además, los estudiantes a menudo no aplican lo que aprenden inmediatamente. Por lo tanto, el conocimiento en la frontera del ME puede ser muy bien conocido por el estudiante pero la falta de uso llevará al módulo psicólogo a dudar de este conocimiento. Además, WUSOR no es capaz de reconocer cualquier comportamiento del estudiante más allá del conocimiento que tiene almacenado en el grafo genético. Esto significa que el autor no sólo debe codificar el conocimiento de dominio, sino también, todos los posibles enlaces que conectan este conocimiento. Finalmente, otro principal problema es que el grafo genético es más bien *ad hoc*. No hay bases teóricas para los tipos de enlaces del grafo, la justificación de su presencia es su necesidad intuitiva.

Parece que esta elaboración mejoraría el MS estándar y soportaría una visión más compleja del estudiante con pocos gastos computacionales adicionales. Sin embargo, el problema fundamental está en construir el grafo. Por esta razón, el progreso en perfeccionar los grafos genéticos para los procesos y pasos del aprendizaje del estudiante en dominios particulares ha sido lento.

#### 2.1.1.4 Modelo de Perturbación y Modelo de Errores

Mientras que los diferentes modelos vistos en el apartado anterior representan al estudiante sólo en términos de conocimiento çorrecto", un Modelo de Perturbación (MP) (Kass (1989)), combina normalmente el modo superpuesto estándar con una representación del conocimiento defectuoso. En los MPs el estudiante no se considera como un mero subconjunto del experto sino que, más bien, poseerá conocimiento potencialmente diferente, en cantidad y calidad, del conocimiento del experto. La técnica común para implementar un modelo de perturbación es representar el conocimiento del experto y luego aumentar esta representación con conocimiento explícito de los conceptos posiblemente equivocados. Por lo tanto, el ME está solapado sobre el modelo general aumentado (ver figura 2.3).

Como puede verse en la figura 2.3, el MP puede también representar el conocimiento y creencias del estudiante más allá del límite del modelo experto.

Cuando el estudiante demuestra un modelo general más o menos consistente pero incorrecto, se llama comúnmente un *concepto erróneo*. Un *error* se refiere a algún defecto estructural en un procedimiento que a menudo se manifiesta por sí mismo en el comportamiento defectuoso.

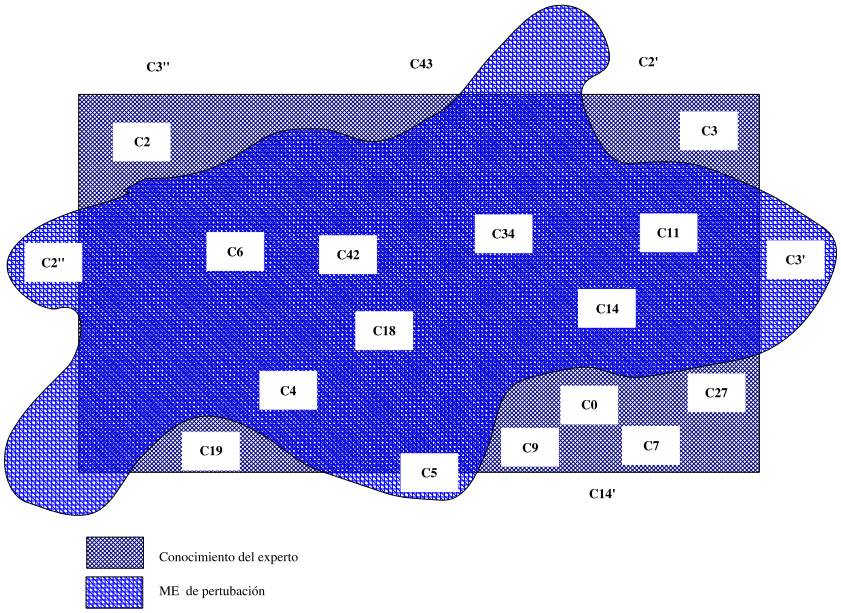


**Figura 2.3:** Modelo de Perturbación

A pesar de las definiciones anteriores, los términos “error” y “concepto erróneo” se usan frecuentemente de forma indiscriminada y representan perturbaciones para el modelo experto. Una colección fija de errores y conceptos erróneos se denomina generalmente *librería de errores* o *catálogo de errores*. Esta librería de errores no es un ME, sólo será útil para ayudar a reconocer las causas particulares de errores y de conceptos erróneos específicos. Cuando el estudiante progresa, el MP puede actualizarse para considerar la presencia o ausencia de errores conocidos en la librería de errores.

La ventaja de este modelo con respecto a los MSs es que, la inclusión de los errores permite una comprensión más sofisticada del estudiante. Ejemplos destacados de sistemas que han usado un MP en su aproximación al ME son:

- *BUGGY* (Burton (1982)): es un sistema diseñado para modelar las habilidades incorrectas (*buggy*) en el cálculo de restas de niños. Más tarde, se utilizó para ayudar a los profesores a reconocer los errores aritméticos de los estudiantes. Otro programa perteneciente al mismo proyecto que BUGGY es *DEBUGGY*. Este último sistema lee las respuestas de una prueba de un estudiante y propone el error o errores que con mayor probabilidad causaron los errores del estudiante. También pertenece al mismo proyecto el programa *IDEBUGGY*, versión interactiva de DEBUGGY, con la capacidad añadida de generar dinámicamente problemas para ayudar a identificar los errores del estudiante. El objetivo de todos estos programas es construir un modelo de diagnóstico del estudiante, es decir, un modelo del conjunto de instrucciones o reglas incorrectas capaz de duplicar el comportamiento del estudiante.

  El modelo de diagnóstico se representa en BUGGY usando una red de procedimientos para simular el comportamiento del estudiante. Dicha red consiste en una parte conceptual, usada por el programa para representar las técnicas utilizadas en las restas, y una parte operacional consistente en métodos (tanto correctos como erróneos) para realizar la acción asociada a la parte conceptual. En los programas BUGGY, esta red está formada por

una rejilla llamada *rejilla de técnicas*, que representan todas las posibles formas en que un estudiante podría intentar resolver los problemas de restas. Dado que esta red de procedimientos incluye tanto métodos erróneos como correctos, estos sistemas usan un modelo de perturbación para modelar al estudiante.

El ME de BUGGY requiere un pequeño conjunto de problemas para determinar los procedimientos erróneos del estudiante. Cuando los problemas resueltos por el estudiante se presentan a DEBUGGY o IDEBUGGY, se prueban varias combinaciones de métodos correctos y erróneos con el objetivo de encontrar un camino a través de la rejilla de técnicas que mejor encaje en la ejecución del estudiante. Un conjunto de heurísticas en este proceso reduce el número de modelos a probar, usando para ello características del dominio tales como dependencias o incompatibilidades entre las técnicas manejadas. Según Burton (1982), el sistema DEBUGGY reconoce 110 errores primitivos, y 20 errores múltiples comunes.

Una contribución de estos sistemas puede ser la evidencia empírica contundente de que los estudiantes cometen errores sistemáticos al calcular restas con múltiples columnas. Brown y Burton probaron que incluso muchos de los peores estudiantes eran muy consistentes en la aplicación de un procedimiento para resolver problemas. Su pobre actuación era debida al uso del procedimiento incorrecto y no a la falta de inteligencia o de interés como creían muchos profesores hasta ese momento. Esta observación fue muy importante para modelar al estudiante pues implicaba que los errores del estudiante se debían a conceptos erróneos identificables y, además, podían ser modelados. Otra contribución es que el sistema DEBUGGY también introdujo un método para tratar el ruido aleatorio en los datos de las pruebas realizadas por el estudiante. A pesar de la consistencia en los procedimientos erróneos del estudiante, éste puede cometer fallos ocasionales por copiar o por descuidos que complican el proceso de modelado.

Además, estos sistemas proporcionan mediante la rejilla de técnicas un modelo detallado del procedimiento de restar. Incluso un procedimiento sencillo como restar puede ser bastante complejo y hay muchas maneras de que un estudiante cometa un error.

Ciertas críticas fundamentales también pueden ser realizadas sobre los sistemas BUGGY. En primer lugar, el listado de errores de Brown y Burton tiende a ser superficial. Muchos de los errores parecen ser diferentes manifestaciones de unos pocos errores fundamentales. El modelo puede usarse para reproducir los fallos de los estudiantes pero no para explicar por qué el estudiante ha cometido tales errores. Por lo tanto, BUGGY no proporciona un modelo profundo de los procedimientos de cálculo de restas del estudiante. En segundo lugar, el conocimiento experto del sistema, aunque codifica el conocimiento de procedimiento necesario para realizar las restas, es una caja negra, es decir, no puede usarse para justificar o explicar el razonamiento del experto o el estudiante. Finalmente, todo el conocimiento del sistema BUGGY tuvo que ser codificado a mano. Se tuvieron que leer detenidamente las pruebas de los estudiantes e intentar determinar qué error o combinación de errores podrían explicar las respuestas proporcionadas. Cuando los errores fueron descubiertos, se añadieron al sistema. No hay capacidad de aumentar el conocimiento del sistema directamente desde la interacción (no hay adaptabilidad).

- *LMS* (Sleeman and Smith (1981)): es un sistema para probar técnicas de álgebra. LMS usa reglas de producción para representar las técnicas correctas necesarias para resolver ecuaciones algebraicas sencillas. Puede aumentarse este conjunto de reglas con reglas erróneas

adicionales que representan los conceptos equivocados que el estudiante podría tener. La construcción de un ME consiste en seleccionar del conjunto de reglas (correctas y erróneas) una lista ordenada de reglas de producción capaces de simular el funcionamiento del estudiante con problemas de álgebra.

El modelado en LMS tiene lugar en dos fases. La primera fase es fuera de línea. Cuando LMS genera un conjunto completo de posibles MEs a partir de las reglas correctas y erróneas, se queda con aquéllas que proporcionan resultados para conjuntos de problemas predefinidos. La segunda fase tiene lugar en línea, usando los modelos generados en la primera fase para determinar si el comportamiento del estudiante puede explicarse mediante cualquiera de dichos modelos. La fase en línea se desarrolla en etapas diseñadas para aumentar un modelo incremental del estudiante. En cada uno de estos pasos el estudiante debe resolver un conjunto de problemas diseñados para probar una técnica algebraica concreta.

La potencia de LMS es que usa una búsqueda sistemática para construir el ME. Los modelos potenciales se limitan a modelos generados a partir del modelo existente, añadiendo o bien una regla correcta o una errónea para la técnica que está siendo probada o, quizás, una regla incorrecta correspondiente a una técnica anterior (se añadió esta capacidad cuando se descubrió que incluso aunque un estudiante pueda realizar una técnica correctamente una vez, después puede usar un método incorrecto para realizar esta técnica).

Sleeman presentó unos estudios que indicaban que el modelo de reglas de producción simple es insuficiente. El análisis de los errores algebraicos de los estudiantes y de las entrevistas sugirió que había cuatro clases de errores de estudiante en álgebra (Sleeman et al. (1989)) pero únicamente una clase de errores (errores manipulativos -se trata del uso de una variante de una regla correcta, al omitir una etapa o sustituirla por otra) es explicada por el modelo LMS original. Consecuentemente, se añadieron después más reglas erróneas.

LMS hace tres importantes contribuciones al estudio del modelado del estudiante. A diferencia de DEBUGGY, que usaba heurísticas para limitar el número de modelos que tenían que ser buscados, LMS sistemáticamente busca el espacio de posibles modelos, usando un método incremental para construir el ME que limite efectivamente el tamaño del espacio de búsqueda de forma tal que la técnica de búsqueda sea computacionalmente factible. En segundo lugar, el modelado dirigido por los datos es posible. En concreto, el mecanismo de construcción del modelo en LMS es relativamente independiente del dominio.

Muchas de las críticas que se pueden plantear a este sistema se deben a que se centra en un aspecto particular del problema de modelado. En primer lugar, los modelos producidos no son modelos psicológicos del aprendizaje del estudiante. Aunque pueda constituir un modelo deseable para álgebra sencilla, no hay bases teóricas para afirmar que LMS trabajaría en otros dominios (por ejemplo, en dominios que no estén bien definidos o con gran cantidad de conocimientos objetivos -de hecho). Además, aunque el modelo para álgebra pueda explicar el comportamiento de un estudiante, no proporciona ninguna indicación sobre por qué los estudiantes tienen este conjunto concreto de reglas erróneas o por qué los estudiantes, en general, cometen algunos tipos de equivocaciones y no otros. Aunque Sleeman reconoce este problema, no ofrece ninguna solución en LMS. Como segunda crítica, se puede plantear que LMS puede sufrir de no-monotonía en el modelo. Al ser éste construido incrementalmente, es posible que una regla posterior pueda, por ejemplo, subsumir una regla existente ya en el modelo. Tal interacción podría causar que el modelo no

siga explicando el funcionamiento del estudiante previamente observado. Una solución a este problema parece ser un chequeo más amplio del modelo cuando una nueva regla se añada para asegurar que capacidades explicativas previas no se han perdido. Finalmente, un problema práctico con LMS es la cantidad de información codificada a mano que debe ser explícitamente proporcionada por el constructor del sistema. El diseñador no sólo debe codificar las reglas básicas y reglas erróneas del dominio, sino, también, gran cantidad de problemas usados para distinguir entre varias reglas durante las etapas de construcción en línea del modelo. Esto requiere que el constructor del sistema sea capaz de anticiparse mucho al comportamiento del estudiante.

- *PROUST* (Johnson and Soloway (1984)): es un sistema desarrollado para identificar errores no sintácticos de estudiantes principiantes en los programas en lenguaje PASCAL. En los trabajos anteriormente descritos parece claro que hay una carencia de modelos psicológicos profundos, es decir, que el modelo vaya más allá de simular simplemente el comportamiento del estudiante para ser capaz, además, de explicar las causas subyacentes de tal comportamiento. Entre las aproximaciones que intentan construir estos tipos de MEs cabe destacar PROUST. Otros ejemplos en esta línea son el tutor GREATERP y la teoría reparadora (*Repair*) (Brown and VanLehn (1988)). Estos últimos se describirán más adelante.

  Para conseguir sus objetivos, PROUST debe abordar el alto grado de variabilidad de la programación, identificar correctamente dónde se equivoca un estudiante al escribir un programa y determinar, para la tutoría, las intenciones del estudiante al escribir el programa de esta forma. PROUST usa una estructura de meta y una base de conocimientos de relaciones entre metas y planes para identificar los errores del programa. Las descripciones de problemas se proporcionan a PROUST incluyendo una especificación del problema en un lenguaje especial orientado a la meta. Cuando se analiza un programa del estudiante, PROUST selecciona una submeta a partir de la descripción del problema e intenta encontrar estados en el programa del estudiante que la satisfagan. Cuando no puede encontrarse una equiparación entre la submeta y el código del estudiante, PROUST intenta encontrar una equiparación usando una librería de errores que contiene errores típicos. También intentará reestructurar las submetas para encontrar una mejor equiparación. PROUST continua seleccionando submetas y prueba a equipararlas con el código hasta que todas las submetas hayan sido probadas. Se generan así todas las posibles interpretaciones y, al final, PROUST seleccionará la interpretación que explique más completamente el programa del estudiante. La librería de errores de PROUST fue clasificada usando un árbol denominado *GAP*, que clasifica errores potenciales reflejando la forma en que tienden a aparecer. Los planes erróneos se representan como árboles parciales GAP y proporcionan información sustancial acerca del error, de forma independiente del problema.

  Aunque sus autores, Jonson y Soloway, no afirman que PROUST usa un modelo psicológico de cómo programan los estudiantes, este sistema va más allá de un simple modelado del comportamiento del estudiante, ya que es capaz de identificar no sólo el plan usado por el estudiante al escribir el programa, sino también sus intenciones al usar el plan. Aunque PROUST no podría usarse para explicar por qué el estudiante usó un plan equivocado, identifica los conceptos erróneos del estudiante al usar un plan concreto para completar una meta particular. Esto proporciona información útil para la corrección del plan.

  La principal limitación de este sistema es que está implementado fuera de línea, pues el tutor sólo tiene acceso al producto final sobre el que basar su diagnóstico de los errores

del estudiante. De este modo, los programas del estudiante terminados son presentados a PROUST, que imprime el diagnóstico.

Hay varias aproximaciones para el desarrollo y representación de librerías de errores, que dan lugar a los diferentes Modelos de Errores existentes (Holt et al. (1994)):

- *Teorías enumerativas de errores*: Enumeran todos los errores basándose en el análisis empírico de los errores del estudiante. Un gran obstáculo en el desarrollo de una librería de errores enumerativa es el esfuerzo que se requiere para formarla y mantenerla. Esta aproximación supone típicamente el análisis de una gran base de datos de interacciones estudiante-tutor. Ejemplo puro de este tipo de teorías es *ACTP* (Wenger (1987)) aunque suelen utilizarse combinaciones de características de más de un tipo de teoría de errores. Por ejemplo, la librería de errores de PROUST es una colección enumerativa de errores de programación de Pascal (tabla 2.1).

- *Teorías generativas de errores*: Generan errores basándose en un conjunto de conceptos erróneos subyacentes. Estas teorías ofrecen mecanismos plausibles para explicar errores en términos de su generación a partir de un modelo cognitivo subyacente. Las teorías generativas no sólo poseen un lenguaje para expresar manifestaciones de errores sino que, además, incluyen un modelo de ejecución que proporciona el contexto para interpretar los errores observados.

  Las aproximaciones generativas pueden soportar un modelo de perturbación del estudiante más sofisticado que una simple librería de errores. Sin embargo, una dificultad de esta aproximación es que pueden generarse muchos errores improbables y la inclusión de un chequeo de plausibilidad para cada error generado es excesiva. Un modelo generativo de errores preciso que logre una buena cobertura del dominio pero que produzca pocos errores improbables es extremadamente difícil de crear. Por ello, el esfuerzo que se necesita para crear así este modelo no es muy diferente del esfuerzo en catalogar ejemplos de errores desde la observación empírica en el modelo enumerativo, a pesar de que éste suponga el análisis de una gran base de datos.

  Un ejemplo de la aproximación generativa es la teoría reparadora (*Repair*) propuesta por VanLehn y Brown ([VANL82]). Su objetivo es proporcionar un modelo más profundo para los errores aritméticos de los estudiantes. En particular, la teoría es capaz de generar errores evitando aquellos errores que no han sido observados o no son intuitivos. La base fundamental de la teoría es la proposición de que cuando un estudiante alcanza un punto muerto en la realización de un procedimiento, intentará reparar dicho procedimiento y continuar normalmente. Un punto muerto tiene lugar cuando el estudiante llama a un subprocedimiento que desconoce, o cuando ha estado realizando un procedimiento incorrecto que lleva a un callejón sin salida. En este punto, el estudiante intenta reparar el procedimiento realizando una o más operaciones para regresar a una forma del problema que sepa cómo resolver. La secuencia *punto muerto-reparación* se modela en cuatro partes: (a) un sistema de producción restringido para representar los componentes de una técnica y sus alternativas erróneas. Las reglas en el sistema se compilan en un grafo generalizado AND/OR, y se limitan a una acción sencilla y una cláusula meta. Además, las reglas sólo se disparan una vez y las reglas especializadas son disparadas antes que las reglas más generales. (b) Un conjunto de principios que se usan para borrar fragmentos del procedimiento correcto usados para simular el punto muerto alcanzado por el estudiante. Este borrado es restringido; por ejemplo, si dos reglas tienen la misma meta y una

es un caso especial de otra, la regla más general no puede borrarse sin primero borrar la más específica. (c) Un conjunto de heurísticas de reparación que se usan para proponer correcciones cuando se alcanza un punto muerto. Este rango va desde un simple abandono o salto del área del problema, cambiar el enfoque, o intentar reglas análogas para superar el punto muerto. (d) Un conjunto de criterios para filtrar algunas reparaciones. El conjunto de heurísticas de reparación tenderá a generar reparaciones que son, o bien no intuitivas o que no son observadas en los estudiantes actuales. Dentro de estas últimas heurísticas se incluyen reglas como no cambiar un número una vez que ha sido escrito o no permitir un blanco en mitad de la respuesta. Tanto estos criterios como las heurísticas reparadoras son específicas del dominio de la aritmética, por lo que la teoría no es general. Brown y Van Lehn opinan que es posible producir un conjunto de heurísticas que puedan ser adaptadas al dominio particular de un sistema.

La propuesta de la teoría reparadora da lugar a varias características interesantes. En primer lugar, las reparaciones son independientes de los puntos muertos. Se separa la causa del problema del método usado para su resolución. En segundo lugar, el problema a resolver es local. Cuando se alcanza un punto muerto, la teoría reparadora intenta fijar el problema sobre las bases del estado actual de los acontecimientos. Finalmente, debido al uso de una pila para interpretar una regla, al resolver un problema sólo se puede llamar a submetas existentes o a acciones primitivas. Esto limita las acciones que pueden ocurrir durante una reparación, proporcionando una teoría más ceñida.

- *Teorías reconstructivas*: Reconstruyen errores sobre la base de errores observados, usando posiblemente una forma de generalización basada en la explicación. Esto implica responder a la pregunta: *¿Qué clase de concepto erróneo conocido más profundo podría haber llevado al error observado?*. Ejemplos puros de esta aproximación son *ACM*, *PIXIE* y *ADVISOR* ([WENG87]).

La Tabla **??** presenta una clasificación de algunos ejemplos existentes de teorías de errores. Dado que este tipo de sistemas combinan a menudo características de más de un tipo, la clasificación se hace en dos pasos: La clasificación principal (que aparece por filas) es refinada con diferencias adicionales (columnas). Por lo tanto, los ejemplos puros de cada tipo de teorías están en la diagonal.

Además de los inconvenientes esbozados en algunos de los ejemplos presentados anteriormente, otros inconvenientes importantes de estos modelos de perturbación son:

- La tarea de inferir errores a partir de las interacciones de los estudiantes con un sistema computerizado también tiene problemas. Los diagnósticos pueden resultar equivocados, particularmente si el error existente no está dentro de la librería ni es una combinación de elementos de ella. Si se usa una aproximación enumerativa o generativa, la cobertura de la librería de errores resultante es un problema.

- Otro inconveniente de algunos de los modelos de perturbación, tales como DEBUGGY, es que, aunque pueden descubrir qué errores ha cometido el estudiante, no explican por qué han ocurrido.

- Como el Modelo Solapado, el modelo de errores enumerativo puede criticarse por no tener una representación del conocimiento de dominio subyacente. Por otro lado, en las teorías generativas, el diagnóstico puede suponer la participación de muchas hipótesis,

**Tabla 2.1:** Ejemplos de tipos de teorías de errores.

| **Tipos de teorías** | ***Enumerativa*** | ***Reconstructiva*** | ***Generativa*** |
|---|---|---|---|
| ***Enumerativa*** | ACTP: lista extensamente definida de errores observables | DEBUGGY, LMS: combinación de errores enumerados que reconstruyen errores observados | MENO-II: errores enumerados reconocidos y atribuidos a conceptos erróneos enumerados |
| ***Reconstructiva*** | PROUST: reconstruye intenciones de diseño usando una librería de planes erróneos | ACM, PIXIE, ADVISOR: reconstruyen errores desde un lenguaje de primitivas neutrales | Young & O'Shea [YOUN81]: procedimientos incorrectos reconstruidos mediante manipulaciones que explican la naturaleza de los errores |
| ***Generativa*** | Bonar & Soloway [BONA85]: librería de generadores de errores abstractos que explican los orígenes de los errores observados | REPAIR: genera errores repitiendo el tratamiento de puntos muertos | REPAIR/STEP [MATZ82]: reducción de la ocurrencia de errores en el aprendizaje erróneo |

de tal modo que el conocimiento obtenido por un diagnóstico particular puede no ser particularmente útil.

- Aunque la información adicional en el modelado de perturbaciones amplía las formas de explicar el comportamiento del estudiante, también posee nuevos problemas. El espacio de búsqueda que debe tratarse en la construcción y mantenimiento del ME es enormemente extenso, y por este motivo, normalmente se requieren técnicas de búsqueda heurística para podar el espacio de búsqueda.

- Se ha cuestionado (Sleeman et al. (1989)) la utilidad de un ME *basado en errores* como solución, cuando se encontró que, para tutelar álgebra, volver a enseñar era tan efectivo como remediar errores específicos. Otros estudios han encontrado que incluso dentro de un dominio seleccionado, los errores diagnosticados varían enormemente entre las diversas escuelas, clases estudiadas, etc., limitando ampliamente la generalidad de cualquier sistema [PAYN91]. Es más, no se ha mostrado que los patrones de errores y estilos de aprendizaje que existen en un área de estudio se apliquen a todas o sólo a algunas áreas de dominio.

#### 2.1.1.5 Otras aproximaciones al ME

La aplicación de diversos métodos de representación del conocimiento existentes han dado lugar a la aparición de ciertas aproximaciones para el ME que, en muchos casos, se pueden considerar variaciones o extensiones de algunas de las aproximaciones anteriormente vistas. Entre ellas cabe destacar:

- *Aproximación de Elsom-Cook*. La forma en que el modelo se implementa puede considerarse una variación del MS visto anteriormente. Elsom-Cook. (Elsom-Cook (1990)) afirma que un tutor no necesita conocer exactamente el estado del conocimiento del estudiante en cualquier instante. Sugiere que un sistema puede mantener un intervalo de confianza alrededor de los límites inferior y superior del conocimiento del estudiante. De este modo, describe una sencilla implementación de esta aproximación en un sistema para tutoría de LISP, usando técnicas de aprendizaje automático estándares, y que consiste básicamente en lo siguiente: a partir del comportamiento del estudiante, el sistema induce los límites inferior y superior de lo que el estudiante conoce. Entonces, basándose en el conocimiento de dominio del sistema, se usa deducción para generar predicciones y, a su vez, problemas que prueben estas predicciones. De este modo, el estudiante puede tener un comportamiento que coincida con la predicción del tutor, y las deducciones usadas en el sistema no tienen por qué ser necesariamente modelos que se adecuen a los procesos de razonamiento del estudiante. Estos modelos limitados pueden ser más tratables en su construcción que los modelos exactos, debido a que los requesitos de corrección son menos precisos.

- *Modelo Basado en Restricciones* (MBR). Este modelo (Ohlsson (1994)) amplía la aproximación del MS permitiendo un razonamiento mucho más sofisticado acerca de los conceptos de dominio, más allá de si son conocidos o no. La representación del dominio de conocimiento en MBR se realiza mediante un conjunto de restricciones sobre el estado de los problemas, y el ME es una lista de restricciones que el estudiante ha violado en el proceso de resolución del problema. Una restricción se caracteriza por una *cláusula de relevancia* y una *cláusula de satisfacción*. La cláusula de relevancia es una condición que debe ser verdadera antes de que la restricción sea relevante para la solución actual. Dada esta cláusula, la cláusula de satisfacción debe ser verdadera para que la solución sea correcta. Las restricciones que constituyen el conocimiento de dominio pueden ser consideradas como un conjunto parcial de reglas, donde cada restricción es una regla de la forma *If (condición de relevancia) es verdadera, then (condición de satisfacción) debe ser también verdadera*. Las restricciones podrían ser también consideradas como una librería de errores implícita. Una librería de errores contiene una lista de errores que han sido empíricamente recogidos a partir de las soluciones del estudiante y que determinan qué errores contiene la respuesta actual. De este modo, en el MBR, cada restricción representaría un error de la forma *When (cláusula satisfacción) no se cumple (no se mantiene la certeza de la condición de relevancia), then el estudiante ha introducido un error*. El ME es el conjunto de restricciones que han sido violadas y representan los conceptos de dominio con los que el estudiante no está conforme. Ohlsson argumenta que es peligroso asumir que el conjunto de restricciones relevantes que el estudiante no violó representa lo que el estudiante conoce, pues podría haber satisfecho inadvertidamente una restricción, sin comprender el concepto subyacente. Sin embargo, una restricción violada siempre representará o una creencia inconsistente o un descuido.

La principal ventaja de este enfoque es su robustez (no depende de la estrategia que haya seguido el alumno para resolver el problema y, por tanto, puede modelar a estudiantes que tengan patrones de comportamiento inconsistentes, es decir, que usen estrategias diferentes para problemas diferentes) y flexibilidad (suficientemente flexible para reconocer soluciones innovadoras como correctas). El modelo es computacionalmente sencillo y no prescribe una estrategia tutorial particular, aunque no está claro cómo esta aproximación se generalizaría a través de otros dominios y estrategias tutoriales.

Este modelo se implementó con éxito en *SQL-Tutor* de Mitrovic (Mitrovic (1998), Mitrovic and Ohlsson (1999)). Este SIT enseña el lenguaje de base de datos SQL para estudiantes de universidad. El sistema contiene 406 restricciones. Las restricciones se dividen en dos categorías: sintácticas y semánticas. Las restricciones sintácticas verifican la sintaxis de la consulta del estudiante. Las restricciones semánticas verifican que las respuestas del estudiante contienen la información requerida; compara la respuesta del estudiante con una solución ideal permitiendo variaciones válidas. Las restricciones se almacenan en el sistema como predicados Lisp y representan un error específico. El ME consiste en una lista de todas las restricciones que han sido relevantes para las respuestas del estudiante. Cada restricción tiene tres contadores: el número de veces que la restricción fue relevante para la solución del estudiante, el número de veces que fue relevante para la solución ideal, y el número de veces que ha sido violada por el estudiante. Los valores de estos contadores se usan para determinar el siguiente problema a presentar. Martin (1999) sugiere extender el ME de SQL-Tutor para incluir conocimiento de alto nivel. Los motivos fundamentales para añadir conocimiento profundo al MBR (y, en concreto, a SQL-Tutor) son: (a) para mejorar la selección del siguiente problema a presentar. Un MBR sólo tiene las violaciones de restricciones individuales para realizar esta elección y, por lo tanto, sólo puede presentar nuevos conceptos en términos de restricciones individuales. Esto puede llevar a la presentación de muchos problemas por la falta de enfoque en áreas de especial dificultad. También, la inclusión de conceptos más generales permitiría al estudiante seleccionar el área general en el que desea recibir tutoría. (b) para ayudar al profesor a comprender el progreso del estudiante. Las restricciones tienen una representación tan específica del dominio del problema que no pueden representar una visión global de la capacidad o progreso del estudiante. Una descripción de más alto nivel de áreas de dificultad debería ser útil para profesor y estudiante, respectivamente. (c) para ayudar a la realimentación (*feedback*). Un sistema que puede determinar el concepto que hay detras de un error puede proporcionar ayuda acerca de este concepto también. Para conseguir lo anterior, Bred Martin propone la inclusión de una jerarquía de conceptos que incremente el poder del MBR para guiar el proceso pedagógico. De este modo, las restricciones se agruparon en conceptos básicos del lenguaje de consulta de SQL. Estos conjuntos fueron luego divididos repetidamente en grupos de restricciones que comparten subconceptos comunes, hasta que los conceptos formaban un árbol con restricciones individuales como nodos hoja. Los nodos de nivel más alto en el árbol (a parte de la raíz) representan los principios fundamentales de las consultas de SQL (ordenación, expresiones, alias, tablas All, etc.). Añadiendo estos conceptos de alto nivel, los errores de concepto de los estudiantes pueden deducirse a partir de las restricciones que violaron. Además, permite encontrar los conceptos que un estudiante encuentra difíciles.

- *Modelos del estudiante de diagnóstico difuso de Katz y Lesgold* (Katz and Lesgold (1992)). En este tipo de modelos se usan procedimientos estadísticos para propagar cambios a

partir de acciones observables en variables de conocimiento locales (p.e. habilidad para medir resultados de tests) a variables más globales (p.e. habilidad para usar el equipo). La presencia o ausencia de las variables de conocimiento (p.e. habilidad para usar el equipo de test) se representa mediante una distribución de probabilidad en cinco niveles de la variable de conocimiento, desde *ningún conocimiento* a *conocimiento desarrollado completamente*. Esta elaboración en la representación del conocimiento permite acciones de tutoría de granularidad más fina basadas en un conjunto particular de variables de conocimiento. El modelo está basado en el concepto de conjuntos difusos y ejemplificado de una forma característica para un dominio particular.

- *Aproximaciones de Gilmore y Self.* Gilmore and Self (1988) exploraron el aprendizaje automático como forma de representar al estudiante en un sistema para aprendizaje de conceptos, con un ME en el que se aplicaba un tipo de sistema de clasificación *ID3* (Quinlan (1984)). Un defecto fundamental de esta aproximación es que requiere que todos los ejemplos sean probados a la vez y usando una métrica artificial para determinar las características más discriminantes. Además, sólo aprende conceptos conjuntivos y usa restricciones de búsqueda no reales. Estos mismos autores también analizaron el aprendizaje por esquemas como un método para modelar al estudiante. Sin embargo, los autores concluyeron que los mecanismos de aprendizaje son arbitrarios, sin soporte psicológico. En general, ha habido pocos éxitos en la importación directa de técnicas de aprendizaje como mecanismos para modelar al estudiante. Sin embargo, Woolf [WOOL92], más optimista, sugiere que hay numerosas formas en que el aprendizaje automático puede ser aplicado útilmente para modelar al estudiante.

### 2.1.2 MEs con representación de la información de proceso

Clancey (Clancey (1986)) clasifica las bases de conocimientos, incluyendo los MEs, según las diferentes formas en que describen los procesos. Mediante dicha clasificación se pueden identificar y estudiar tipos de representaciones cualitativas. El ME puede, por lo tanto, considerarse capaz de simular el proceso por el cual el estudiante resuelve un problema. El ME explica el comportamiento del estudiante mediante un modelo (de dominio) general y un procedimiento de inferencia (cómo el estudiante interpreta las observaciones y el modelo general). Un modelo de simulación de un estudiante debería ser capaz de predecir lo que el estudiante hará a continuación, así como trabajar "hacia atrás" partiendo del comportamiento del estudiante para generar una explicación. Este modelo debería ser un modelo de proceso ejecutable. Pocos de los sistemas existentes emplean MEs de proceso ejecutable.

A diferencia de los sistemas descritos anteriormente, que contienen sólo información de estado, un ejemplo de sistema considerado como modelo de proceso (o de simulación) es el tutor de Lisp denominado *GREATERP* (Farrell et al. (1984)) implementado por Anderson, Corbett, Koedinger y Pelletier. Además, Anderson, Boyle y Yost (Anderson et al. (1985)) implementaron un tutor de Geometría usando para ambos el mismo sistema subyacente.

- GREATERP (Kass (1989)) es un tutor para programadores en Lisp principiantes, y el Tutor de Geometría se diseñó para ayudar a los estudiantes a aprender a contestar pruebas de geometría de escuela superior. Ambos tutores (y otros) están basados en la teoría *ACT** (Anderson (1983), Anderson et al. (1985)). Las características principales de esta teoría son: (a) un sistema de reglas de producción donde cada regla representa una unidad de capacidad. (b) Las reglas de producción están dirigidas por la meta, es decir, una regla sólo se

puede disparar si se puede equiparar una meta. (c) Una memoria de trabajo que almacena todo el conocimiento actual pero limitada en tamaño. (d) Un alumno llegará a estar más capacitado mediante la adquisición de nuevas producciones. A este proceso se le llama compilación del conocimiento.

El ME en el tutor de Lisp está implementado como un GRAPES (sistema de producción restringido a la meta -*Goal-Restricted Production System*) que implementa las características de ACT*. Ambos sistemas de tutoría, GREATERP y el tutor de Geometría, tienen un experto, llamado el estudiante ideal, cuyo conocimiento de dominio es codificado en GRAPES, así como un catálogo de errores con los fallos más comunes o malas estrategias utilizadas por los estudiantes. El estudiante ideal junto con el catálogo de errores forman el modelo ideal y de fallo (IBR).

El ME está construido mediante la técnica denominada *Traza del Modelo*. Anderson, Corbett, Koedinger y Pelletier ([ANDE95]) distinguen entre dos tipos diferentes de modelado del alumno, denominados traza del conocimiento y traza del modelo. La traza del conocimiento consiste en determinar qué sabe el alumno, que engloba tanto el conocimiento correcto sobre el dominio como sus errores. La traza del modelo pretende analizar el procedimiento de resolución de problemas que utiliza el alumno. Esta traza es útil en sistemas que intentan dar respuesta a peticiones de ayuda del alumno y proporcionarle pistas e información cuando no consigue continuar la resolución de un problema. Para poder ofrecer esta ayuda, el sistema requiere ser capaz de analizar y criticar la solución en curso y tener una idea de la línea de razonamiento que está siguiendo el alumno. Por otro lado, la traza del conocimiento resulta útil para la evaluación del alumno y toma de decisiones pedagógicas, como qué material/problema debe proponerse a continuación.

Básicamente, con el método de traza del modelo, el tutor de Lisp realiza la traza del comportamiento del estudiante a partir de su modelo ideal y de fallos. En cualquier momento, pueden existir muchas reglas de producción candidatas a ser aplicadas por el tutor, que infiere qué regla es la apropiada comparando las acciones del estudiante con los resultados de aplicar cada una de esas reglas. Si la regla seleccionada es una regla correcta (una regla del ME ideal) la sesión continua. Si, de lo contrario, la regla es errónea, el tutor interrumpirá para corregir la situación. Este método de tutoría se basa en la evidencia empírica de que el mejor momento para corregir los conceptos equivocados del estudiante es cuando suceden. Como consecuencia de esta filosofía, el ME nunca está más lejos de un paso del modelo correcto. Esto permite un modelado rápido y una tutoría más efectiva.

GREATERP y el tutor de Geometría requieren un gran manejo de conocimiento predefinido para el ME ideal y para el catálogo de errores. Una vez proporcionado este conocimiento, los sistemas pueden funcionar muy bien, ya que restringen el rango de posibles MEs en cualquier momento. Pueden considerarse ambos sistemas como una extensión del trabajo de Sleeman con LMS, incluyendo conocimiento profundo a nivel de unidades de cognición humana. Tal aproximación en estos tutores es tan completa que capturan un gran porcentaje de todos los errores del estudiante.

Una limitación fundamental de esta aproximación de traza del modelo es que no permite a los estudiantes cometer sus propios errores. Tan pronto como hay un paso erróneo, el tutor avisa del fallo y detiene al estudiante hasta que tome el paso correcto. No sólo se le impide al estudiante continuar con estos errores hasta su conclusión lógica (y llegar a estar completamente despistado) sino que también se le impide formarse una idea del error (es decir, que el error sea evidente). De esta forma, en algunas de las mejores experiencias

**Tabla 2.3:** Ejemplos de posibles combinaciones de modelos de dominio y de razonamiento

| | **Modelo de razonamiento** | |
|---|---|---|
| **Modelo general de dominio** | *De comportamiento* | *Funcional* |
| *De comportamiento* | MYCIN /GUIDON | NEOMYCIN |
| *Funcional* | Tutor LISP | PROUST, SOPHIE-III |

de aprendizaje que puede tener un estudiante, éste es bloqueado por tal aproximación. La traza del modelo es, pues, restrictiva ya que restringe la libertad del estudiante. Otra limitación de esta aproximación es que trabaja muy bien en el modelado de adquisición de capacidades de procedimiento pero no trabaja bien para dominios que están mal estructurados o no basados en reglas.

Esta noción de proporcionar explicaciones para procesos que suceden a lo largo del tiempo, como el comportamiento del estudiante, ejemplificada con los dos sistemas anteriores, es una de las características que definen a los modelos de simulación en contraposición con la naturaleza instantánea de los modelos de estado. Clancey (Clancey (1986)) distingue dos tipos de modelos de simulación:

- *Modelos de simulación de comportamiento*. Describen las acciones, lo que se observa que está haciendo el estudiante.
- *Modelos de simulación funcional*. Describen las creencias y metas, lo que el estudiante conoce y qué está intentando hacer.

El Tutor LISP simula el proceso de programación a través de patrones situación-acción pero sólo captura información de comportamiento del estudiante. Un modelo de comportamiento puede expresar las metas que hay detras de los pasos de resolución de problemas, pero éstas son estrictamente específicas del dominio. Sin embargo, un modelo funcional intenta representar el propósito deseado en la interacción formativa. Por ejemplo, en PROUST el código del estudiante se analiza en términos de su comportamiento resultante, más que a través de, únicamente, un análisis estático. Esto proporciona el potencial de generar explicaciones del comportamiento del estudiante a nivel de plan o propósito.

Las ventajas del modelo funcional sobre el modelo de comportamiento (Holt et al. (1994)) se deben a su potencial de generalizar la explicación del razonamiento, el modelado de conceptos erróneos y la anticipación de situaciones, a través de dominios. Las explicaciones desde los modelos de comportamiento, a diferencia de los funcionales, no pueden distinguir la diferencia entre errores de hecho, errores en las metas y planes subyacentes, y errores en la traducción de planes en acciones.

La tabla 2.3 muestra las posibles combinaciones de MEs en cuanto al modelo general (modelo de dominio) y procedimiento de inferencia (modelo de razonamiento). Como se puede observar, el modelo de procedimiento de inferencia y el modelo general no necesitan ser del mismo tipo de modelo de proceso.

En resumen, el modelo de proceso es un modelo del estudiante más completo, ya que representa no sólo el comportamiento del estudiante en términos del modelo general sino también el mecanismo de inferencia usado para resolver el problema. Su principal ventaja es la mayor capacidad para explicar el comportamiento del estudiante, en especial, en el caso de modelos de proceso funcionales. Además, la generalidad del modelo de proceso tiene la ventaja de ser menos específico del dominio, prestándose de este modo a la ejecución mediante motores de inferencia estándares.

#### 2.1.2.1 Otros ejemplos de modelos de simulación de comportamiento

- *OLAE* (Martin and Vanlehn (1995)) es una herramienta que obtiene información sobre alumnos que resuelven problemas de introducción a la física, analiza los datos recogidos mediante métodos probabilísticos basados en redes bayesianas, y trata de determinar qué sabe el alumno. Para ello, este sistema genera automáticamente para cada problema una red bayesiana que relaciona el conocimiento, representado con reglas de primer orden, con acciones concretas (por ejemplo, ecuaciones escritas). Con la red obtenida y observando el comportamiento del estudiante, OLAE calcula las probabilidades de que el estudiante conozca y use cada regla.

  El grafo de resolución de problemas en OLAE es una red dirigida de unos 150 nodos de diferentes tipos (por ejemplo, nodos para representar si el alumno conoce o no una regla del dominio (*nodos de regla*), nodos para representar si el estudiante conoce ese hecho (*nodos de hecho*), nodos para representar si el estudiante ha realizado esa acción (*nodos de acción*), etc.).

  La red bayesiana se genera automáticamente a partir del modelo del dominio de la forma siguiente: si una determinada regla se puede usar para producir una conclusión a partir de ciertos antecedentes, se introduce un tipo de nodo en la red para representar la aplicación de dicha regla y un arco desde el nodo de aplicación de la regla hasta el nodo de hecho que represente su conclusión (si no existe este nodo, se crea). Por cada antecedente o hechos usados para justificar el disparo de la regla correspondiente, se introduce un arco desde su nodo de hecho asociado hasta el nodo de aplicación de la regla así como un arco desde el nodo de la regla hasta el nodo de aplicación de la regla. Si un determinado hecho tiene asociada una acción observable, se crea también un nodo de acción y se coloca un arco desde el nodo de hecho hasta el nodo de acción. Cuando la red bayesiana está completamente generada, el estudiante resuelve el problema. Entonces, OLAE obtiene las probabilidades a posteriori de que conozca una determinada regla, actualizando las probabilidades a través de los arcos de la red mediante algoritmos de propagación.

  Además de la red anterior, OLAE proporciona una segunda red bayesiana diseñada específicamente para el profesor que consulta el sistema una vez finalizado el proceso anterior. Forman parte de los nodos de esta red: los *nodos de regla* de la red bayesiana original (representan el resultado del proceso de inferencia del sistema) así como nodos dimensionales que almacenan información de variables más abstractas y que representan el dominio que tiene el estudiante sobre partes específicas del currículo (por ejemplo, de Cinemática). Como limitación en este sistema se puede considerar el que no estén directamente incluidos estos nodos en la red porque esto permitiría que: a) sus probabilidades se actualizaran a medida que evolucionan otras probabilidades de la red y b) si se adquiriera conocimiento sobre cómo domina el estudiante determinadas partes del currículo, este

conocimiento podría influir también en la probabilidad de que domine las reglas que lo componen.

Otra posible limitación del sistema OLAE es que actúa cuando el alumno ha terminado de resolver el problema. Su propósito no es proporcionar un soporte para enseñanza interactiva, sino sólo diagnosticar de forma precisa qué partes del dominio son conocidas por el estudiante.

#### 2.1.2.2 Otros ejemplos de modelos de simulación funcional

- *SPYROS* (Herzog and Zierl (1994)) es un SIT basado en lógica difusa sobre programación paralela. Este sistema, como muchos otros que manejan dominios basados en procedimientos, usa un conjunto de objetivos y planes con estructura arbórea para representar el conocimiento del dominio. Además, al árbol se le añaden planes y objetivos incorrectos con objeto de que el sistema sea capaz de interpretar la solución proporcionada por el alumno.

  El diagnóstico de la solución del alumno básicamente consiste en los siguientes pasos:

  - *Algoritmo de reconocimiento*. Dada una sentencia, encuentra todos los planes que concuerden con ella $\{P_1, \ldots, P_n\}$.
  - *Algoritmo de interpretación*. Selecciona uno de los planes localizados por el algoritmo de reconocimiento anterior. Para ello, el algoritmo usa técnicas difusas basadas en diez tipos de evidencia (por ejemplo, errores más comunes cometidos por los alumnos, nivel del alumno, explicaciones proporcionadas al alumno hasta el momento, dificultad relativa de los objetivos asociados a cada plan, etc.). Dicha información es procesada por un sistema experto con reglas difusas, y la asignación de un plan para la sentencia se realiza mediante una función de pertenencia que ordena los planes según el grado en que la sentencia los apoya, y usando una función no mónotona para asignar valores de pertenencia difusos a cada plan, de tal modo que aquél con mayor grado recibe un valor de pertenencia de 1 y el resto en el intervalo [0, 1).
  - *Algoritmo de diagnóstico*. Dado el conjunto de planes que determinan las sentencias del programa del alumno y el conjunto de objetivos correspondientes a esos planes, asigna un grado de corrección a cada objetivo (correcto, innecesario, incorrecto, etc.) para determinar el grado de corrección del programa del alumno.

  Los sistemas siguientes (SHERLOCK II, ML-MODELER y MDF) se basan en una idea similar a la de SPYROS:

- *SHERLOCK II* (Katz et al. (1994)) es un sistema basado en lógica difusa que proporciona tutoría a los alumnos para diagnosticar averías en aviación. En este sistema se usan variables de conocimiento a las que se asocia una distribución de probabilidad difusa con cinco valores desde *ningún_conocimiento* a *conocimiento_total* (ejemplos de variables son: habilidad para usar aparatos de medición, habilidad para interpretar los resultados de una prueba, etc.). La distribución asociada a una variable de conocimiento se inicia con la distribución uniforme (1/5, ... , 1/5) representando la ignorancia sobre el estado de conocimiento del alumno, aunque podría iniciarse a otros valores si se tuviera información sobre este estado. Además, la distribución se actualiza mediante varias reglas, aumentando

o disminuyendo dicha distribución de acuerdo a ciertos factores como el tipo de evidencia disponible (acciones del alumno, pistas que se le han presentado hasta el momento, etc.). La regla de actualización se especifica mediante dos parámetros: un *vector de escala* $V$ ($V = (v_1, ..., v_5)$, donde $v_1 = 0$). El vector de escala sirve para controlar la velocidad de actualización. Por ejemplo, para aplicar una actualización más lenta cuando el alumno se considere de nivel muy avanzado y no cometer una equivocación al clasificarlo como experto demasiado pronto) y un *porcentaje de cambio c* para controlar la razón o medida de actualización (por ejemplo, para que indicadores débiles que ocurran frecuentemente actualicen la variable lentamente e indicadores fuertes que ocurran con poca frecuencia actualicen la variable rápidamente). Es importante señalar que estos mecanismos de actualización son intuitivos, basados en ideas de cómo debe evolucionar la creencia conforme se adquiere nueva evidencia, pero no están relacionados con ninguna regla de teoría de la probabilidad y, por lo tanto, carecen de fundamento teórico.

Las variables de conocimiento se dividen en SHERLOCK II en: *variables locales*, usadas para evaluar habilidades específicas, y *variables globales* usadas para representar abstracciones sobre grupos de variables locales. Por ejemplo:
$X \doteq habilidadparausarlosequiposdemedición$ se relaciona con las variables locales:
$X_1 \doteq habilidadparausareosciloscopio$,
$X_2 \doteq habilidadparausarelmultímetro$ y
$X_3 \doteq habilidadparausareltermómetro$
de la forma siguiente:

$$X = (0.6X_1 + 0.2X_2 + 0.2X_3) \tag{2.1}$$

Este sistema presenta el peligro de desarrollar aproximaciones *ad-hoc* para procesar la evidencia en vez de usar métodos teóricos con validez demostrada y sufrir anomalías como, por ejemplo, que el resultado obtenido para X, de acuerdo a la relación expresada anteriormente, no refleje correctamente la relación entre las variables.

- *ML-MODELER* (Gürer et al. (1995)) es el módulo del estudiante de un sistema que permite tutoría adaptativa para la enseñanza de Química. El modelado del proceso de aprendizaje de un alumno es dinámico y, para ello, compara la traza de la solución del alumno con la traza de la solución experta, generando hipótesis sobre los errores del alumno, y usando razonamiento basado en casos infiere los métodos de aprendizaje que ha utilizado el estudiante para lograr el estado actual de conocimiento. Por lo tanto, ML-MODELER es capaz de modelar no sólo los errores y áreas conceptuales problemáticas para el alumno sino también el uso incorrecto de técnicas de aprendizaje (por ejemplo, analogía, generalización o especificación).

  La representación del conocimiento experto y del alumno se lleva a cabo mediante una red conceptual (denominada MOP). Esta red representa el problema, los conceptos usados para resolverlo y su solución. Una red conceptual del estudiante representa de esta forma un episodio de resolución de problemas y consta de características, hechos, conceptos y un conjunto de ecuaciones y procedimientos. El modelo del estudiante consiste en su estado de conocimiento y sus mecanismos de aprendizaje, mediante una red conceptual con los conceptos, procedimientos y mecanismos de aprendizaje que ML-MODELER cree que está usando el alumno.

En este sistema se usa la lógica difusa para la selección de las heurísticas y conceptos que mejor explican el comportamiento del estudiante. A diferencia de SHERLOCK II, se usan siete valores en vez de cinco (desde *definitivamente_no* a *definitivamente_sí*) para describir cada concepto y enlace en la red del estudiante. Las reglas de actualización de esos valores son las mismas que las de SHERLOCK II pero el vector de escala difiere ($V = (0,1,1,1,1,1,1)$) y no existen variables globales, como en ese sistema, paliando así sus anomalías.

- *Sistema MFD* (Beck et al. (1997)) es un tutor de matemáticas cuyo objetivo es enseñar operaciones básicas con diversos tipos de números (enteros, fracciones, números mixtos y decimales). Cada tipo de problema se considera en este sistema un tema y entre ellos hay relaciones de prerrequisito. Además, cada tema lleva asociados un conjunto de habilidades, que son pasos en el proceso de resolución del problema (por ejemplo, un tema es sumar fracciones y tiene asociadas como habilidades: {*encontrar el mínimo común múltiplo*, *calcular fracciones equivalentes*, *sumar numeradores* y *simplificar fracciones*}).

  El ME en MFD tiene dos tipos de información: un nivel de conocimiento para cada tema y factores generales relativos a cada alumno (como la capacidad de adquisición de nuevos conocimientos, denominada *factor de adquisición*, y capacidad de recordar conocimientos antiguos, denominada *recuerdo*). Para representar la incertidumbre de cada tema se usa un vector de creencias de siete componentes que suman 1. Estos valores indican la posibilidad aproximada de que el estudiante haya alcanzado el nivel de conocimiento correspondiente (por ejemplo, el vector de creencias (0.3, 0.6, 0.1, 0, 0, 0, 0) para un tema concreto significa que hay una probabilidad de 0.3 de que el estudiante tenga nivel 1, 0.6 de que tenga nivel 2, 0.1 de que tenga nivel 3 y que, con seguridad, no tiene nivel superior a 3 para dicho tema). La actualización de estos valores se realiza mediante fórmulas basadas en las de ML-MODELER, añadiendo a ellas otros factores como las pistas ofrecidas al estudiante y sus factores de adquisión y recuerdo. Las pistas tienen un índice, usado para describir el grado de información que se da al estudiante, de tal modo que, para índices menores que el índice de la pista de mayor información mostrada al estudiante hasta la actualidad, se usa la regla de aumento como regla de actualización: C = A, siendo A función del factor de adquisición, y para índices mayores que el índice de la pista, se usa la regla de disminución C = B, siendo B función del factor de recuerdo.

  El ME anteriormente descrito permite seleccionar el tema objetivo, generar el problema al nivel adecuado de dificultad, y proporcionar información al alumno de forma adaptativa. Las limitaciones fundamentales de este sistema, según los mismos autores, es que no se basa en un marco teórico sólido (no hay comprensión formal de cómo funciona) y que, en lugar de manejar el vector de creencias, se realiza una suma ponderada de sus componentes, donde cada valor del vector se pondera usando el nivel correspondiente. El valor único obtenido se utiliza para medir el conocimiento del estudiante en el tema. A pesar de que los propios autores parecen considerar este problema prioritario, no proponen alternativas en publicaciones posteriores (Beck and Woolf (1998)).

  Tanto SHERLOCK II, como los sistemas ML-MODELER y MFD usan, como se ha descrito anteriormente, ecuaciones *ad-hoc* para actualizar el ME basándose en la idea intuitiva de desplazar la distribución a la derecha o a la izquierda según la evidencia disponible. Esta técnica intenta imitar la forma en que razonan los humanos, pero las inconsistencias del modelo pueden provocar que el comportamiento del ME sea impredecible, concretamente en situaciones que no son previstas por sus autores.

- *Sistema ALLEN* (Torres et al. (1994)) es un SIT sobre análisis de circuitos. Al igual que SPYROS, este sistema usa reglas difusas y las trata numéricamente con definiciones y técnicas estándar en lógica difusa. El aprendizaje de los estudiantes se realiza en dos etapas:

  Una primera *fase de adquisición de conocimientos conceptuales* (estudio de la teoría y ejemplos en un entorno de hipertextos) en la que el sistema usa reglas difusas para inferir el conocimiento del estudiante y así seleccionar la estrategia instructora apropiada para la sesión posterior de resolución de problemas. La selección de dicha estrategia se basa en los patrones que ha seguido el estudiante durante la navegación a través del hipertexto, usando para ello reglas difusas. Una vez que todas las posibles reglas en la base de conocimientos difusa se aplican a un determinado estudiante, se le asigna un identificador nítido (*bueno*, *medio*, *por debajo de la media*, etc.) de acuerdo al valor de pertenencia más alto para todos los conjuntos difusos inferidos como consecuentes de las reglas aplicadas. Este identificador es el que se usa para seleccionar la estrategia instructora más adecuada que se aplicará en la fase posterior de resolución de problemas. La estrategia podrá modificarse si el comportamiento del estudiante durante esta fase indicara que puede necesitar otro tipo de interacción.

- La segunda *fase* es *de adquisición de habilidades*, para mejorar las habilidades del estudiante durante la resolución de problemas de aplicación de la teoría aprendida. La interacción con el estudiante en esta última etapa de resolución de problemas, como ya se ha descrito, es adaptativa, pudiéndose llevar a cabo bajo tres estrategias instructoras distintas.

  Tanto en el sistema SPYROS como en ALLEN es necesario, al usar reglas difusas, seleccionar los significados de los operadores *Y*, *O* y *NO* de la biblioteca de funciones de lógica difusa para poder interpretar esos operadores.

  En resumen, podemos decir que todos estos sistemas con lógica difusa en el ME usan esta técnica como alternativa de bajo coste en cuanto a esfuerzo de ingeniería de conocimiento, aunque una aplicación más minuciosa y basada en estudios empíricos de mayor envergadura hubiera permitido obtener MEs más precisos y consistentes.

- *POLA* - *P*robabilistic *O*n-*L*ine *A*ssessment ([CONA96a]), es una extensión y mejora del sistema OLAE. Ambos fueron realizados por el equipo de Kurt Vahn Lehn en la Universidad de Pittsburg. POLA es realmente el módulo de diagnóstico del estudiante en *ANDES* ([CONA97a] [CONA97b], [GERT98], [VANL96], [VANL98]), un sistema instructor inteligente para Física Newtoniana. Es una mejora del sistema OLAE puesto que permite construir el ME con la técnica de traza del modelo y, por lo tanto, a diferencia de éste que sólo realiza lo que Anderson y otros [ANDE95] denominan traza del conocimiento (como ya se vió anteriormente, consiste en determinar qué sabe el alumno, conocimiento correcto y errores), POLA realiza también la traza del modelo (seguimiento de la forma de resolver un problema). En el caso de POLA, determina no sólo las reglas que sabe el estudiante sino también, cuando existan varios caminos de resolución que sean consistentes con la acción que ha tomado el estudiante, el sistema es capaz de decidir qué camino es más probable que haya conducido al estudiante a la resolución del problema. A partir de esa información, el sistema proporciona nuevas capacidades como: contestar preguntas formuladas por el alumno, generar pistas a un nivel adecuado o tomar decisiones pedagógicas como proporcionar una ayuda, presentar un determinado material o elegir el siguiente problema a proponer. Para lograr este objetivo, el módulo de diagnóstico del sistema necesita conocer

el conjunto de posibles líneas de razonamiento que el estudiante puede seguir, denominado espacio de soluciones. La estructura que representa el conjunto se denomina grafo solución. Este grafo se construye automáticamente a partir de una base de conocimiento, de reglas de producción, y consta de la siguiente información: a) todos los planes para resolver el problema derivados de las reglas de la base de conocimiento; b) todos los caminos algebraicos de resolución que desarrollan esos planes y, c) razonamiento subyacente a esos planes. Un ejemplo de grafo de solución para un problema ejemplo puede verse en [CONA96b].

De acuerdo a la terminología de OLAE, los nodos de aplicación en POLA son nodos de tipo *AND* (para que una regla se aplique, la regla y sus antecedentes deben conocerse) y los nodos de hecho son de tipo *OR* (a ellos se puede llegar por varios caminos distintos). De este modo, el sistema genera un grafo *AND*/*OR* que representa todas las formas conceptualmente diferentes en las que se pueden combinar las reglas y los datos para llegar a la solución final. La red bayesiana se va construyendo de forma incremental a medida que el estudiante resuelve el problema. Así, con objeto de determinar los posibles caminos solución que ha escogido el alumno, se distingue entre las reglas que el alumno ha usado ya y las que pertenecen a su camino solución pero aún no han sido utilizadas. Estas últimas reglas no formaran parte de la red bayesiana que se usa para la inferencia.

En [VANL98] se trata de determinar las probabilidades a priori que tiene un alumno de conocer o no cada una de las 350 reglas en las que se dividió el dominio en el sistema ANDES. Para ello, se partió de un examen desarrollado por profesores de Física y el problema consistía en encontrar un algoritmo de diagnóstico, es decir, un algoritmo que dadas las respuestas de un estudiante a las preguntas del examen y las relaciones entre preguntas y reglas, determinara el subconjunto de reglas que eran conocidas por dicho estudiante. En este estudio se usan alumnos simulados y se modelan los aciertos casuales sin poseer conocimiento (adivinanzas -*guesses*) y errores no intencionados (descuidos -*slip*) mediante expresiones. Alguna de estas expresiones son mejorables según el trabajo de Eva00. Para medir la bondad del algoritmo de diagnóstico en ANDES se utilizan como medidas: (a) *precisión*, que se define como la proporción entre el número de reglas que el sistema ha diagnosticado correctamente como dominadas por el alumno simulado y el número de reglas que fueron diagnosticadas como dominadas, (b) *cobertura*, que se define como la proporción entre el número de reglas que el sistema ha diagnosticado correctamente como dominadas por el alumno simulado y el número de reglas que el alumno domina. Idealmente, ambos parámetros deben tomar valores próximos a 1.

Para modelar con redes bayesianas las relaciones entre preguntas y reglas se probaron varios esquemas alternativos. Se concluyó que el esquema que más se ajustaba era: *conocer una regla tiene influencia causal en contestar correctamente un problema (relación R ->P)*. Sin embargo, este modelo elegido tenía el problema de que las probabilidades a posteriori (después de evaluar las respuestas) parecían depender fuertemente de las probabilidades a priori que se especificaran. Este problema se resolvió basando el diagnóstico en la medida del cambio en la probabilidad, es decir, la diferencia entre la probabilidad a posteriori y la probabilidad a priori (en lugar de basar el diagnóstico en los valores de las probabilidades a posteriori).

- *HYDRIVE* (Mislevy and Gitomer (1996)) modela la habilidad que tiene un alumno para diagnosticar averías en el sistema hidráulico de los aviones F-15. El piloto describe a través de un vídeo algún problema en el funcionamiento del avión cuando está a punto de

aterrizar o despegar. Mediante la interfaz del sistema el estudiante podrá diagnosticar la avería por los procedimientos usuales y consultar vídeos de componentes y material técnico. El sistema observa el comportamiento del alumno con el objeto de evaluar cómo utiliza éste la información para dirigir las acciones encaminadas a diagnosticar la avería. El sistema de diagnóstico evalúa la calidad de acciones de diagnóstico de averías concretas y caracteriza el conocimiento del alumno mediante el uso de variables más generales, dividiendo tal conocimiento en tres partes: *conocimiento del sistema*, *conocimiento de estrategias*, y *conocimiento de procedimientos de resolución*, dividiéndose, a su vez, éstos en otros nodos y variables. Así, el diagnóstico producido por el sistema es suficientemente preciso para saber qué partes del conocimiento no posee el alumno y es lo bastante general para guiar las estrategias instructoras básicas (por ejemplo, si dada la respuesta del alumno se debe presentar una ayuda o proponer una situación más complicada).

Una característica importante de HYDRIVE es que el sistema de inferencias es mixto. Existe un intérprete de estrategias que, mediante un conjunto pequeño de reglas (aproximadamente 25), caracteriza la estrategia de resolución que está usando el alumno.

En la figura siguiente se muestra un ejemplo de los nodos y de cómo se modelan las relaciones entre ellos en la red bayesiana utilizada en el sistema HYDRIVE.

- *Aproximación de red bayesiana dinámica de Jim Reye* (Reye (1996), Reye (1998)). Se basa en modelado bayesiano del estudiante, apoyándose en la hipótesis de que el dominio de conocimiento se puede estructurar en una colección abstracta de temas que representan conocimiento conceptual o habilidades que el estudiante debe adquirir. Estos temas, según el autor, pueden estructurarse como relaciones de prerrequisitos. Fundamentándose en ello, Reye propone una red bayesiana basada en una parte central que conecta todos los nodos de conocimiento (*nodos student-knows*). Estos nodos se ordenan de acuerdo a una lista de relaciones de prerrequisito. Asimismo, la red bayesiana se basa en un grupo de nodos para cada nodo de conocimiento de ls parte central. En cada grupo aparecen el nodo de conocimiento y nodos adicionales relacionados con él (por ejemplo, nodos para medir el interés del alumno en el tema concreto, a los que Reye denomina *nodos student-interested-in*). La ventaja de esta estructuración es que las actualizaciones sucesivas de la red a medida que se adquiere nueva evidencia se realizan localmente afectando sólo a las partes de la red correspondientes al tema. Esto permite aumentar la eficiencia en los cálculos. Sin embargo, una limitación de este enfoque es que se basa en una estructura muy restrictiva dado que los nodos de la red no pueden utilizarse para representar diversos factores, a diferencia de otros sistemas vistos basados en redes bayesianas.

  En (Reye (1998)), Reye presenta una red bayesiana dinámica para modelar al estudiante. En este trabajo, el concepto de dinámico en el tiempo se mide en función de interacciones con el sistema en lugar de en función de intervalos de tiempo. De este modo, para cada i= 1, ... ,n se define el nodo $L_i$ = *estado del conocimiento del tema que posee el alumno después de la i-ésima interacción con el sistema*. Este nodo dependerá del nodo $L_{i-1}$ y del nodo $O_{i-1}$(resultado de la interacción n-ésima) que, a su vez, dependerá también de $L_{i-1}$.

- *Aproximación basada en redes bayesianas y test adaptativos del grupo de laboratorio ARIES*. Esta aproximación se realizó por James Greer et al. (Collins et al. (1996)), y se basa en el uso de redes bayesianas en test adaptativos. Estos tests pueden definirse como tests en los que la selección de la siguiente pregunta a proponer y la decisión de finalización del test

se adoptan dinámicamente basándose en la estimación actual del estado de conocimiento del alumno obtenido a partir de las respuestas a preguntas anteriormente planteadas (Millán et al. (2000)). Además, su trabajo se apoya en las jerarquías de granularidad (McCalla and Greer (1994)). La granularidad se refiere a los niveles de detalle o visiones que se adoptan de un concepto o entidad. La jerarquía de granularidad captura diferentes niveles de detalle en un tipo de red semántica. Esta jerarquía es usada por el grupo ARIES para evaluar al estudiante partiendo de un conjunto de preguntas de test. Inicialmente, se tiene un dominio de conocimiento estructurado en objetivos a aprender con diferentes niveles de logro específicos y un conjunto de preguntas que pueden ser de cualquier tipo (no es necesario que sean de tipo test) siempre que se garantice que se puede comprobar la corrección o incorrección de la respuesta del estudiante. Asimismo, se consideran diferentes tipos de relaciones: *relaciones de agregación* (permiten la descomposición de un objetivo en subobjetivos garantizando tests de contenido equilibrado), *relaciones de prerrequisitos* (permiten la estructuración del dominio y ayudan a establecer el orden de las preguntas en el test) y *relaciones objetivos-pregunta* entre objetivos de aprendizaje alcanzados y preguntas (son las que permitirán realizar el diagnóstico). Para seleccionar las preguntas, se propone en este trabajo elegir la pregunta más informativa definida como aquélla que maximiza cierta medida de utilidad. La utilidad de una pregunta $Q$ para un objetivo $O$ se define en este trabajo como la probabilidad de que se domine el objetivo $O$ dado que la pregunta se responde correctamente menos la probabilidad de no dominarlo dado que la pregunta se responde incorrectamente: $utilidad\,(Q) \doteq\mid P\,(O/Q) - P\,(\neg O/\neg Q)$.

En (Millán et al. (2000)) se proponen otras medidas de utilidad al considerar la medida anterior discutible, dado que el objetivo debería ser maximizar ambas probabilidades, no la diferencia en valor absoluto.

El criterio de finalización que proponen en este trabajo se basa en dos alternativas: a) que el nivel de conocimiento del objetivo de aprendizaje más grueso caiga por encima o por debajo de ciertos valores o b) en caso de que se requiera una evaluación más completa, usar los niveles de conocimiento de cada uno de los subobjetivos de aprendizaje (es decir, cada uno de los hijos del nodo objetivo de aprendizaje).

En este trabajo no se discierne cuál sería la estructura más adecuada de la red bayesiana (en (Collins et al. (1996)) aparecen diversas estructuras), cuál es el efecto de añadir o no nodos dimensionales, ni la dirección adecuada de las relaciones de agregación. Todos estos detalles se resuelven en (Millán et al. (2000)), donde se plantea, en esta línea, un posible modelo del estudiante basado en redes bayesianas.

En las pruebas de validez de esta aproximación se usaron, con tres estructuras de red bayesiana diferentes, alumnos simulados con probabilidades de responder correctamente de 0, 0.2 y 0.8, por lo que no se consideraron alumnos de nivel intermedio que son, en principio, más difíciles de diagnosticar por su comportamiento más impredecible.

## 2.2 Aproximaciones taxonómicas al conocimiento soportado por el modelado del estudiante

### 2.2.1 Taxonomía del conocimiento en el ME de Julita Vassileva

Julita Vassileva (Vassileva (1992)) presenta en un espacio tridimensional las tendencias en las investigaciones en el campo del modelado del estudiante. En concreto, una de las dimensio-

nes que presenta para su análisis son los niveles de conocimiento representados en el modelo, distinguiendo entre modelos que representan:

- *Evaluaciones del funcionamiento del estudiante* representadas como un número, un vector de parámetros, etc. Se pueden combinar los parámetros para formar estructuras o estereotipos pero no se puede realizar ningún mantenimiento de consistencia pues supondría algún conocimiento sobre el significado de los parámetros evaluados para encontrar algún conflicto. Todos los parámetros que describen al estudiante se enumeran por adelantado y la correspondiente acción del sistema tiene que ser predefinida.

- *Creencias del estudiante*. Muchos de los trabajos en modelado del estudiante durante los anos 70 y 80 se dirigen a este nivel de conocimiento, aunque aplicándose a diferentes dominios y partiendo de diferentes paradigmas. Las creencias se introducen en ellos como una unidad de representación del conocimiento. Puede distinguirse entre modelos enumerativos (con un conjunto predefinido de creencias que no pueden modificarse) y modelos generativos (son capaces de generar creencias mediante, generalmente, un mecanismo de inferencia implícitamente empotrado).

- *Meta-creencias*. Clancey deseaba aumentar las capacidades generativas con respecto a la generación de creencias por defecto. En el sistema GUIDON, Clancey representa un conjunto de meta-reglas explícitas o meta-creencias en el modelo que pueden ser aplicadas para cambiar la base de reglas existente y para generar errores.

- *Mecanismos deductivos* (lógico). Por ejemplo, PROUST es un ejemplo de diagnóstico en una representación a nivel lógico. El sistema infiere el plan de un estudiante para resolver un problema a partir de la solución final, es decir, la estrategia de combinación de reglas. ACM [LANG84] es otro ejemplo que infiere el plan del estudiante a partir de la solución final, usando técnicas de aprendizaje automático y una lógica estándar.

- *Meta-lógica*. Si la lógica presenta un conjunto de reglas de razonamiento, una meta-lógica presenta incluso conocimiento a más alto nivel, una estrategia para modificar o cambiar entre diferentes lógicas con objeto de resolver un problema dado. Van Lehn (VanLehn (1982)) crea la Teoría Reparadora, considerada como modelado del estudiante en ambos niveles, lógico y meta-lógico, intentando modelar un mecanismo de razonamiento alternativo.

### 2.2.2 Componentes de conocimiento en múltiples niveles de De Koning y Bredeweg

La extensión del MDG (Motor de Diagnóstico General) presentada por De Koning y otros (de Koning et al. (1995), de Koning and Bredeweg (1998), de Koning et al. (2000)) como procedimiento de diagnóstico en un SIT, permite modelar la experiencia, manejando las nociones de metaconocimiento y metadiagnóstico para representar varios tipos de experiencias. Ambas nociones, son entradas específicas de un marco multiestratificado que se describe a continuación (Bredeweg and Breuker (1993)).

#### 2.2.2.1 KADS: un marco multi-estratificado

Wielinga et al. (1992) propone un marco (KADS) para modelar la resolución de problemas, que se basa en distinguir los papeles que juegan varios tipos de conocimiento (ver Figura 2.4).

El *conocimiento de dominio* consiste en conceptos conectados mediante relaciones. El *conocimiento de inferencia* se refiere a la estructura de los pasos de inferencia (*Fuentes de Conocimiento* -FC) que desempeñan papeles (*metaclases*) en la resolución del problema, así como sus entradas y salidas. Las FC se caracterizan por el tipo de inferencia que realizan. Por ejemplo, una *FC abstracta* borra u oculta atributos de un concepto, mientras que una *FC específica* añade un atributo. Las metaclases tienen el nombre de papeles a desempeñar, y se asocian a conceptos de dominio. Por ejemplo, un concepto de dominio como *transistor-defectuoso* puede desempeñar el papel de *conclusión* o de *hipótesis*.

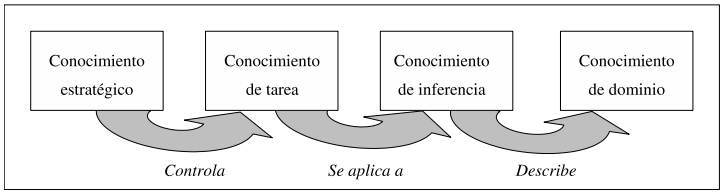


**Figura 2.4:** Las cuatro capas de KADS

El tercer tipo de conocimiento, *conocimiento de tarea*, describe qué metas están implicadas en la resolución de un problema particular y cómo las FC disponibles pueden ser ordenadas y aplicadas para satisfacer dichas metas. El conocimiento de inferencia proporciona una perspectiva de flujo de datos sobre el proceso de razonamiento; el conocimiento de tarea añade una visión de flujo específico de control.

El cuarto tipo de conocimiento, *conocimiento estratégico*, proporciona una visión más flexible y amplia sobre el control de razonamiento. Este tipo de conocimiento se usa, por ejemplo, para planificar una estructura de tarea, controlar su ejecución, y si fuera necesario, diagnosticar, reparar e incluso sustituir la estructura de tarea actual por otra, hasta que se alcance la meta deseada en la resolución del problema. En SITs, donde a menudo se recomienda una estrategia de resolución del problema concreta o varias, las estructuras de tarea fijas son bastante comunes y, por lo tanto, en estos casos, puede no requerirse especificaciones de conocimiento estratégico o sólo de una forma rudimentaria (Bredeweg and Breuker (1993)).

#### 2.2.2.2 Conocimiento de metanivel

En un SIT, la experiencia relevante del dominio formativo se divide normalmente sólo en dos tipos: conocimiento de dominio y estrategia de resolución del problema. La aproximación de 4 capas amplía esta visión para dividir la estrategia de resolución del problema en inferencias, estructuras de tarea (o planes) y estrategias. Esto permite manipular el razonamiento sobre aspectos de *metanivel*.

En su intento de representar la experiencia involucrada en los problemas de resolución de restas, en el marco del MDG, Self (Self (1993)) no distingue un nivel separado de estrategia de resolución del problema. No se presentan las metas de resolución del problema, las diferentes formas de llegar a tales metas, y el conocimiento requerido para razonar con y para reflexionar acerca de estas entidades, es decir, no se presentan aspectos relacionados con el razonamiento de metanivel. Como resultado de esto, no puede encontrarse un diagnóstico que trate con esta cuestión (es identificado, como ya veremos en apartados posteriores, como uno de los "problemas pendientes"de Self, el tercero).

#### 2.2.2.3 Representación de KADS en el paradigma del MDG

En general, cada entidad de conocimiento del marco KADS debe reformularse en términos de un componente, con sus entradas y salidas específicas, y una restricción que especifique cómo puede obtenerse la salida a partir de la entrada. Se usa la siguiente representación: normalmente, una entidad de conocimiento en la capa de dominio tiene tres aspectos: un nombre, una especificación de las condiciones bajo las cuales el conocimiento es aplicable, y una especificación del conocimiento que se proporciona por la entidad cuando es aplicable. En términos de componentes estos aspectos son el tipo, la entrada y la salida del componente. En la capa de inferencia, cada FC se representa como un tipo de componente cuyas entradas y salidas son las metaclases. Una estructura de tarea consiste en estos componentes de inferencia más una secuencia específica de trabajo de estos componentes.

En (Bredeweg and Breuker (1993)) se muestra un ejemplo que clarifica esta división de componentes de conocimiento.

### 2.2.3 Taxonomía propuesta por McCalla y Greer

McCalla and Greer (1994), miembros del laboratorio ARIES en la Universidad de Saskatchewan (Canadá), tomaron como base para la arquitectura soporte del modelado del estudiante la siguiente taxonomía del conocimiento: el conocimiento de un tutor se divide en un número de espacios conceptuales entre los que destacan cuatro: el *espacio meta del tutor*, el *espacio de creencias del tutor*, el *espacio meta del estudiante* y el *espacio de creencias del estudiante* (figura 2.5).

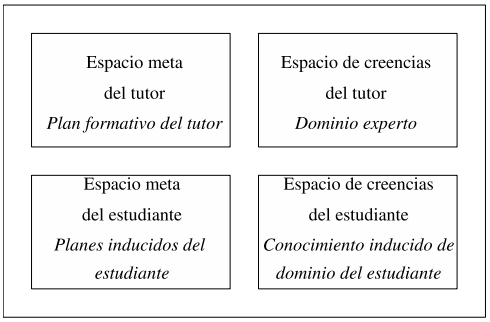


**Figura 2.5:** Espacio meta y espacio de creencias

El espacio meta del tutor incluye las metas pedagógicas y algoritmos de planificación pedagógicos para convertir esas metas en planes formativos. El espacio de creencias del tutor incluye el conocimiento que el tutor tiene acerca del dominio de conceptos que el estudiante está explorando. Este conocimiento puede incluir conocimiento acerca de tareas que ilustran estos conceptos, y conocimiento acerca de conceptos erróneos comunes. Ambos espacios permiten al tutor ser capaz de razonar explícitamente acerca de su propio conocimiento y de sus propias metas pedagógicas.

Por otro lado, el espacio meta del estudiante y el espacio de creencias del estudiante representan las creencias del tutor acerca del estudiante y ambos espacios constituyen el ME. El espacio meta del estudiante contiene las percepciones del tutor de las metas cognitivas del estudiante, así como algoritmos de reconocimiento de planes que el tutor puede usar para inducir los planes del estudiante cuando aprende conceptos en el dominio de tutoría. El espacio de

creencias del estudiante contiene las percepciones del tutor sobre los conceptos del dominio que el estudiante ha comprendido. Este espacio, a su vez, tiene dos subespacios interesantes: las creencias del estudiante acerca de las creencias del tutor y las creencias del estudiante acerca de las metas del tutor. Ambos subespacios son esenciales si el tutor debe predecir la reacción del estudiante al tutor.

Otra idea fundamental en la que se apoya la taxonomía del conocimiento en el ME introducida por McCalla y Greer es que dicho modelo debe ser capaz de razonar en muchos tamaños de grano. Para lograr este objetivo, proponen un esquema de representación que permita el razonamiento basado en *granularidad*. El término de granularidad se refiere al nivel de detalle o perspectiva desde la que es visto un concepto o entidad (una entidad puede ser considerada en un nivel refinado, detallado, o en un nivel más general, aproximado). Se puede pensar en un concepto o entidad descrito en varios niveles de detalle como una colección de modelos que pueden ser abstractos (*grano grueso*) o refinados (*grano fino*). El desplazamiento del enfoque de un tamaño de grano a otro se realiza mediante dos funciones denominadas respectivamente *articulación*, desde los objetos de tamaño más grueso a los objetos de tamaño más fino, y *simplificación*, desde los objetos de tamaño más fino a los objetos de tamaño más grueso. McCalla y Greer tomaron éstas y otras definiciones básicas de granularidad de aquéllas proporcionadas por Hobbs (Hobbs (1985)) e implementadas mediante una representación del conocimiento basada en redes semánticas jerárquicas a la que denominaron *jerarquía de granularidad* (puede verse en detalle la representación en McCalla and Greer (1994)).

La granularidad no afecta sólo a la organización del conocimiento, sino también al razonamiento. Este concepto es importante dado que los estudiantes parecen razonar en muchos tamaños de grano, de tal modo que pueden tener conocimiento profundo y superficial a la vez (esto es especialmente cierto en habilidades de resolución de problemas, por ejemplo para resolver problemas de programación recursiva). Por lo tanto, el ME deberá representar estas perspectivas multinivel en el espacio de creencias del estudiante. Además, los estudiantes tienen sólo una comprensión incompleta del entorno que están explorando desde la ignorancia total hasta un conocimiento parcial para los conceptos erróneos formados. El conocimiento parcial puede tomar la forma de conocimiento sólo acerca de alguna de las componentes que forman un concepto o procedimiento (por ejemplo, tener conocimiento de los pasos de reducción y casos base en la recursión, pero no de los pasos de composición), o puede tomar la forma de ignorancia sobre los detalles completos de un concepto o procedimiento (por ejemplo, conocer que una recursión consiste en pasos de reducción, casos base y pasos de composición pero no conocer que hay tipos específicos de recursión). La relación entre conocimiento parcial y conocimiento más completo también es una relación de granularidad, es decir, cuando un estudiante refina su entendimiento, el estudiante está articulando (Hobbs (1985)) su conocimiento en tamaños de grano más fino. Esta articulación parece suceder a lo largo de, al menos, tres dimensiones: *agregación* (cuando el estudiante comprende los componentes de algún concepto), *abstracción* (cuando el estudiante comprende las especializaciones de algún concepto), y *metas* (cuando el estudiante elabora sus metas en submetas).

De este modo, McCalla y Greer intentaron representar explícitamente las clases de conocimiento del estudiante en términos de granularidad y lo aplicaron en el sistema SCENT-3 (McCalla et al. (1988)).

Una vez que la granularidad ha sido explícitamente incorporada en un sistema de tutoría, puede ser usada para diagnosticar el comportamiento del estudiante, uno de los prerrequisitos clave para el modelado inteligente del estudiante. La dificultad está en reconocer lo que el estudiante está haciendo (sobre todo cuando se le permite moverse libremente por un gran es-

pacio de posibles comportamientos). Este problema se puede controlar restringiendo la libertad del estudiante al ámbito del conocimiento del estudiante (como se hace en el Tutor de LISP (Anderson et al. (1985)). Sin embargo, se puede permitir que el estudiante tenga libertad en su comportamiento mediante el uso de las jerarquías de granularidad, que son útiles así tanto para interpretar el rango del conocimiento del estudiante (grano fino o grano grueso) como para el reconocimiento por el propio sistema de tutoría del comportamiento del estudiante.

La construcción de un ME basado en granularidad comienza con el modelado externo. Examinando el comportamiento en el mundo, es posible en muchos dominios construir una jerarquía de granularidad genérica correspondiente a un dominio de contenido particular y a unas estrategias particulares anticipadas para resolver problemas en el dominio. La forma en que se usan estas jerarquías de granularidad en el modelado interno del estudiante es mediante *instanciación* de una jerarquía de granularidad específica para el uso que hace un estudiante individual en la resolución de un problema sencillo. Para realizar el razonamiento sobre las jerarquías de granularidad instanciadas, McCalla y Greer usaron una aproximación basada en casos con el objetivo de resumir la información (por ejemplo, para un simple problema de recursión en LISP, la jerarquía instanciada resultante puede ser muy compleja con muchas estrategias potenciales que se pueden instanciar cero o más veces). En esta aproximación, la jerarquía de granularidad particular instanciada, correspondiente al comportamiento en un cierto instante de un estudiante, se compara con una base de datos de las jerarquías construidas previamente para la misma tarea (los detalles pueden verse en McCalla and Greer (1994)).

Hay muchas ventajas del razonamiento basado en la granularidad. Algunas de las más importantes son:

- El razonamiento basado en granularidad permite un modelo detallado de las estrategias aparentemente usadas por el estudiante. Este modelo es realmente una jerarquía de modelos detallados, completamente revisables y útiles para aconsejar o informar en varios tamaños de grano.

- La aproximación posee un mecanismo para monitorizar la evolución en las estrategias del estudiante a lo largo del tiempo. Cuando el estudiante hace numerosos intentos para resolver un problema sencillo, las jerarquías de granularidad de estos intentos pueden ser analizadas para monitorizar cómo evolucionan las estrategias. Si el estudiante continúa centrándose en detalles no importantes, el sistema se dará cuenta de ello, si el estudiante ha dominado aparentemente ciertas estrategias o si ha tenido dificultad con ciertas estrategias, la selección de tareas posteriores puede tomarlo en consideración.

- Adaptación natural de la granularidad al contexto dentro de un dominio. En un SIT, el contexto para la interacción con el estudiante gira en torno completamente de los elementos de dominio. El contexto podría ser visto como un punto de vista compartido sobre el dominio. El establecimiento de este punto de vista es facilitado por el enfoque en modelos multi-granulados de una jerarquía de granularidad.

- Las jerarquías de granularidad proporcionan una herramienta útil para razonamiento. Además, la granularidad es una noción central en muchos campos de la IA, no sólo en SIT. Muchas de las nociones de granularidad en las que se ha trabajado han sido originalmente provocadas por investigaciones en otras áreas de IA tales como visión computacional y representación formal del conocimiento.

El segundo tema principal en el modelado del estudiante abordado por McCalla y Greer es el problema del mantenimiento del ME cuando cambia el conocimiento del estudiante a lo largo del tiempo, y cuando el conocimiento que tiene el tutor del estudiante es refinado. Según los autores, esto último requiere un sistema estándar de mantenimiento de la verdad (de Kleer (1986), Doyle (1987), etc.). Tales sistemas son capaces de reconocer los conflictos entre la información nueva y la información existente en algún modelo del mundo (representado en lógica proposicional).

Entre las cuestiones investigadas en el laboratorio ARIES sobre el mantenimiento del ME, y relacionado con el tema que nos ocupa, la taxonomía del conocimiento en el ME, Huang et al. (Huang et al. (1991a)) resolvieron cómo combinar el conocimiento estereotípico de los estudiantes con el conocimiento diagnosticado durante la interacción con el estudiante. En la Figura 2.6 se muestra la arquitectura de un sistema de modelado del estudiante basado en estereotipos y con consistencia limitada. La motivación para investigar cómo modelar espacios de creencias con consistencia limitada es el reconocimiento de que la tarea de mantenimiento del ME es extremadamente dura (Huang et al. (1991b)) y el mantenimiento total de la consistencia en todo el ME no sólo no es alcanzable en el mundo real sino que, además, no siempre es necesario.

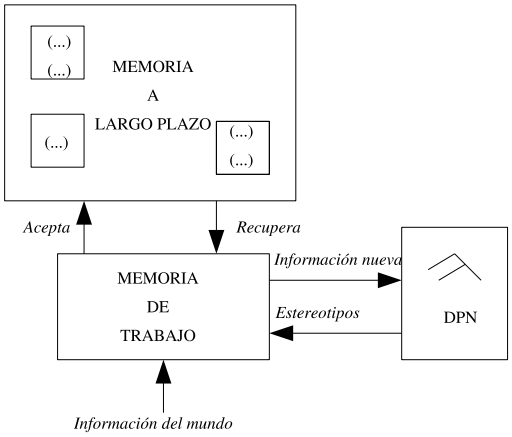


**Figura 2.6:** Sistema de mantenimiento del ME de Huang et al.

Brevemente, la arquitectura describe que hay dos clases de conocimiento en un ME: 1) las creencias del estudiante observadas y las justificaciones para dichas creencias, y 2) las creencias estereotípicas mantenidas por estudiantes típicos en varios pasos de su aprendizaje. Huang et al., representan las creencias observadas mediante proposiciones almacenadas en paquetes llamados *marcos mentales*. Sólo algunos de estos marcos son relevantes en alguno de los problemas en que el estudiante pudiera estar trabajando y se recopilan en una *memoria de trabajo*. Esta memoria es mantenida por un SRCE (*S*istema de *R*evisión de *C*reencias *E*volutivo) que realmente es un conjunto de algoritmos que produce un conjunto minimal de conflictos (conjunto minimal de cambios en las creencias del tutor sobre el estudiante) combinando la aproximación al diagnóstico de Reiter (Reiter (1987)) con el sistema ATMS de de Kleer (de Kleer (1986)). Este mantenimiento permite que la memoria de trabajo sea internamente consistente al menos a nivel de un sistema de mantenimiento de verdad. Cuando el estudiante cambia el foco de atención, algunos de los marcos de la memoria de trabajo (quizás modificados al añadir o borrar creencias) no son necesarios ya y son aceptados en la *memoria a largo plazo* (LTM). Similarmente, los marcos relevantes de la LTM deben ser recuperados en la memoria de trabajo. Huang et

al. (Huang et al. (1991b)) proporciona algoritmos (llamados en su conjunto SRCA, *S*istema de *R*evisión de *C*reencias para desplazar la *A*tención) que llevan a cabo las operaciones de aceptar y recuperar marcos, así como algoritmos eficientes para actualizar la memoria LTM cuando los marcos son aceptados en ella. La actualización de la LTM, aunque eficiente, no garantiza su consistencia total; las creencias del tutor sobre el estudiante pueden ser inconsistentes. Huang y McCalla muestran que esa aproximación a la consistencia limitada trabaja eficientemente en planificación formativa donde el tutor tiene un control de la agenda, pero no está tan claro cómo de bien trabajará para el problema dual de seguimiento de las creencias del estudiante.

El mecanismo de estereotipos de Huang et al. (McCalla and Greer (1994)) se utiliza para tener en cuenta cómo los estereotipos cambian cuando el aprendizaje del estudiante va avanzando. Las creencias estereotípicas se estructuran en jerarquías llamadas *redes de paquete por defecto* (DPN -*D*efault *P*ackage *N*etworks). Un nodo en el DPN representa algún concepto del dominio y las creencias del nodo son creencias del estudiante típico acerca de ese concepto. Ya que la comprensión de un estudiante sobre un concepto puede cambiar a lo largo del tiempo, las creencias son también clasificadas de acuerdo con las fases preestablecidas del aprendizaje en las que son apropiadas; los estudiantes novatos pueden comprender un concepto incompletamente o erróneamente mientras que un experto debería conocer los matices más finos del concepto. Los arcos que conectan los nodos en el DPN representan las relaciones entre los conceptos (por ejemplo, relación de prerrequisito) y se usan para mantener un seguimiento de las dependencias de datos entre conceptos cuando progresa el aprendizaje del estudiante. Las creencias estereotípicas apropiadas en una fase concreta del aprendizaje son introducidas desde el DPN en la memoria de trabajo cuando se necesitan para ser así usadas por el sistema de tutoría en el proceso de razonamiento sobre el conocimiento del estudiante. Si la comprensión del estudiante sobre el dominio varía, el DPN es capaz de actualizar el conocimiento estereotípico. El antiguo paquete de creencias estereotípicas puede ser eliminado de la memoria de trabajo y sustituido con conocimiento más apropiado para la actual fase de aprendizaje.

### 2.2.4 Taxonomía del conocimiento en el ME de Chen y Mizoguchi

Weiqin Chen y Riichiro Mizoguchi describen una ontología del ME y el agente ME en una arquitectura multiagente propuesta por ambos para sistemas de soporte del aprendizaje (Chen and Mizoguchi (1999), Chen and Mizoguchi (2004)).

La ontología (en filosofía) es el estudio de las clases de cosas que existen. En IA, el término ontología se usa para referirse a un conjunto de vocabulario de representación, en concreto, a los conceptos que deben englobar los términos en el vocabulario. La ontología del ME que describen Chen y Mizoguchi facilita la especificación explícita de la funcionalidad del agente ME, la reutilización y compartición de este agente y su comunicación fluida.

Hay dos tipos de ontologías: *ontología de tarea* y *ontología de dominio*. Una ontología de tarea es una teoría/sistema de vocabulario para el modelado de la estructura de resolución del problema inherente en todas las tareas existentes, independientemente del dominio. Una ontología de dominio es una teoría/sistema del vocabulario para clases específicas de objetos y relaciones que existen en algunos dominios. En los sistemas de soporte de aprendizaje multi-agente la ontología del ME es una clase de ontología de tarea, mientras que la ontología de dominio es el vocabulario del tema particular (matemáticas, química, etc.). Ambas ontologías son necesarias para la comunicación con el agente ME. La ontología presentada por ambos autores sólo hace referencia a la ontología del ME y circunscribiéndose sólo a la ontología de información

del ME, que proporciona la especificación externa de la funcionalidad del agente ME sin entrar en el ámbito de la estructura interna del ME.

En la Figura 2.7 se muestra la arquitectura propuesta por ambos autores (Chen and Mizoguchi (2004)). La arquitectura multi-agente de sistemas de soporte del aprendizaje se basa en la suposición de que los agentes comparten una ontología común usada para especificar su funcionalidad, para la construcción de los mensajes de soporte e interpretación y para facilitar la reutilización y compartición de los agentes.

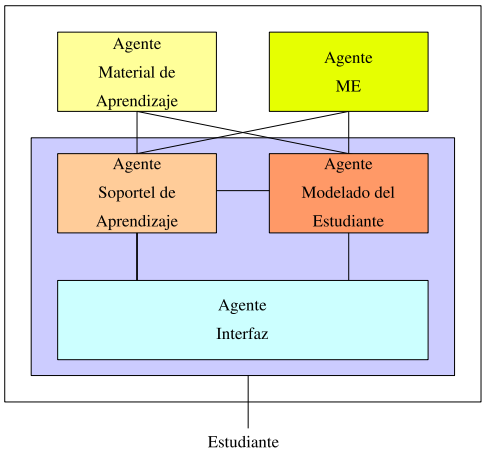


**Figura 2.7:** Una arquitectura multi-agente en sistemas de soporte del aprendizaje

Sin entrar en detalles, dado que nuestro interés en este punto es la taxonomía del conocimiento del ME, a continuación se describe brevemente cada agente en la arquitectura propuesta:

- El *Agente Material de Aprendizaje* se encarga del contenido y organización del conocimiento de dominio. Responde a preguntas de otros agentes que necesitan conocimiento de dominio para completar sus funciones, por ejemplo, el Agente Soporte del aprendizaje toma el conocimiento de dominio como objeto para planear la formación.

- El *Agente Soporte del Aprendizaje* es el responsable de planificar formaciones y generar realimentación. Selecciona los tópicos apropiados para enseñar y toma decisiones sobre las estrategias de enseñanza y acciones de soporte del aprendizaje.

- El *Agente Modelado del Estudiante* y *Agente ME*. Se separan ambos agentes puesto que, según los autores, el agente ME debería por sí mismo ser una entidad, un agente independiente, ya que su información interna podría ser también usada por otros agentes u otros sistemas o incluso por los propios estudiantes.

    - El Agente Modelado del Estudiante es el responsable de la construcción del ME. Deriva la información cognitiva del estudiante a partir de la información extra sobre el estudiante para construir un ME. Pide al Agente Soporte de Aprendizaje que le proporcione los problemas relacionados con el estudiante y así, el Agente Interfaz podrá transferir las respuestas/acciones del estudiante al Agente Modelado del estudiante.

- El agente ME es el responsable de hacer las preguntas a otros agentes sobre la información del estudiante y del propio modelo. Cada agente tiene varios módulos funcionales principales. Los módulos del agente ME son los que se muestra en la figura 2.8. El módulo de comunicación es común a todos los agentes y se encarga de controlar la comunicación con otros agentes (determinar mensajes ejecutivos, enviar y recibir mensajes, interpretar mensajes ejecutivos, etc.).

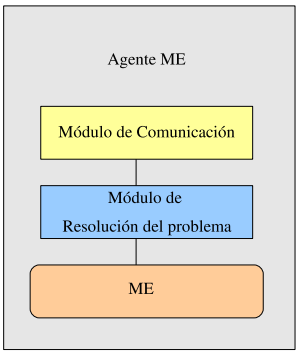


**Figura 2.8:** Agente ME

- El *Agente Interfaz* es el responsable de la interacción entre un estudiante y el sistema. Monitoriza las actividades del estudiante y pasa los mensajes a otros agentes relacionados tales como el Agente Modelado del Estudiante y el Agente Soporte de Aprendizaje.

#### 2.2.4.1 Ontología de la información del ME

La ontología del ME propuesta por Chen y Mizoguchi se muestra en la figura 2.9.

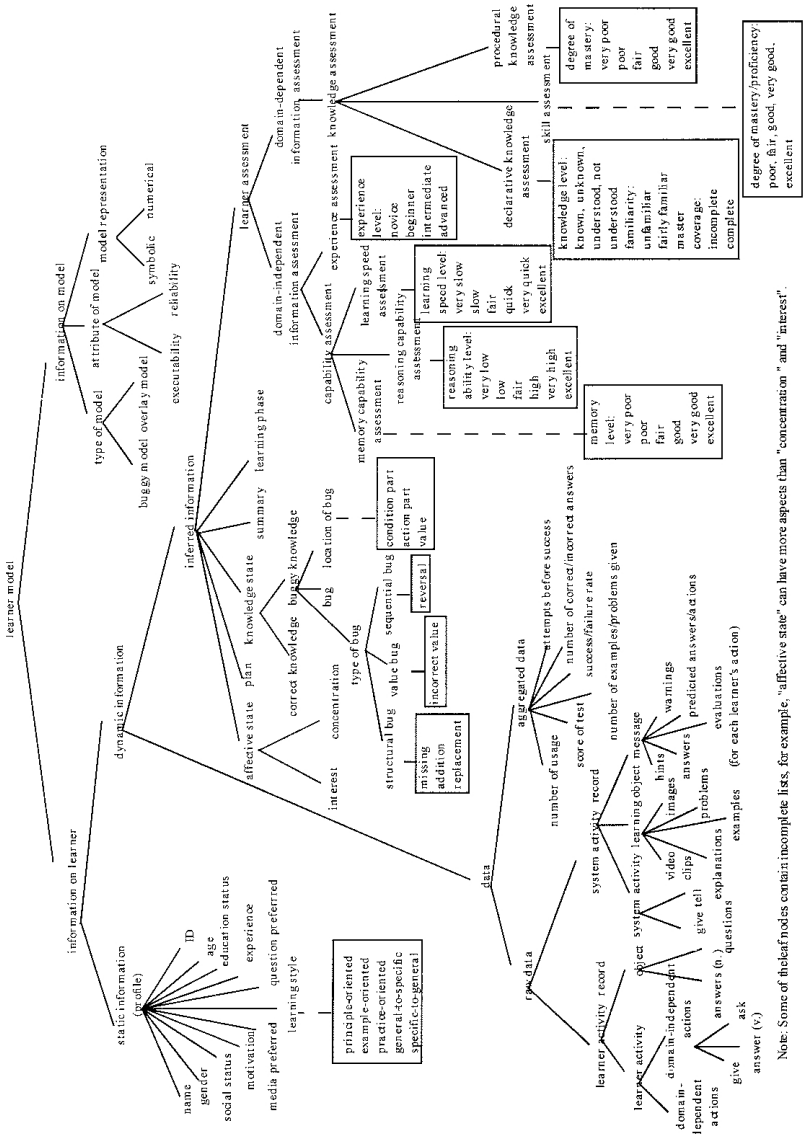


**Figura 2.9:** Ontología del ME (figura extraída de Chen and Mizoguchi (2004))

Como se puede observar, la información proporcionada por el agente ME puede entrar en una de estas dos categorías: información sobre el estudiante e información sobre el modelo. La información sobre el estudiante incluye información estática (el perfil del estudiante) e información dinámica (información conseguida dinámicamente a partir de la interacción entre el estudiante y el sistema). En la información dinámica del estudiante hay datos puros e información inferida (información obtenida a partir de los datos por razonamiento). El razonamiento puede hacerse en diferentes niveles. Por ejemplo, a partir de la respuesta del estudiante a una pregunta proporcionada por el sistema puede inferirse el conocimiento correcto o defectuoso del estudiante (error, tipo de error y localización del error). Similarmente, la habilidad del estu-

diante (memoria, velocidad de aprendizaje y capacidad de razonamiento) puede inferirse.

Usando esta ontología, los autores describen cómo con conceptos independientes de dominio es fácil representar diferentes clases de ME (modelo superpuesto, modelo defectuoso, etc.).Como ejemplo de aplicación, se re-implementaron dos sistemas de soporte del estudiante, WUSOR e IDEBUGGY, compartiendo un agente de ME y ejecutándose los agentes en diferentes computadores. La comunicación se realizaba a través de una LAN gestionada mediante CORBA/Horb (se pueden ver más detalles en Chen and Mizoguchi (2004)).

### 2.2.5 Taxonomía de Niu

Niu (Niu (2002)) distingue entre:

- Sistemas que modelan al estudiante representando el conocimiento de dominio como un *modelo de comportamiento*, como una red de procedimientos, sub-procedimientos, etc., hasta un conjunto de acciones primitivas. El principal conocimiento comunicado es el de comportamiento, es decir, conocimiento acerca de la realización de una tarea.

- Sistemas que modelan al estudiante representando el conocimiento de dominio como un *modelo declarativo* (Dillenbourg and Self (1992)). Este tipo de modelos representan lo que el sistema conoce, almacenan evaluaciones o descripciones del conocimiento del estudiante (usando valores simbólicos o numéricos), no el propio conocimiento. El conocimiento se representa en forma de un conjunto de hechos y reglas organizadas de tal forma que la máquina pueda razonar con ellas.

- *Sistemas de modelado del usuario/estudiante genéricos*. La motivación de estos sistemas es proporcionar la funcionalidad de modelado del usuario/estudiante como *shells* o productos servidor. Los sistemas *shells* de modelado del usuario/estudiante que forman parte de una aplicación ofrecen modelos de usuario/estudiante capaces de ser reutilizados y modificados. Según Kobsa (Kobsa (2001)), un sistema shell de modelado del usuario/estudiante debería poder representar lo siguiente:

    - Suposiciones acerca de uno o más tipos de usuarios característicos en modelos de usuarios individuales (por ejemplo: suposiciones acerca de su conocimiento, fallos de concepto, metas, planes, preferencias, tareas y habilidades), suposiciones que se puedan formar acerca del usuario basadas en la historia de la interacción y suposiciones adicionales acerca del usuario actual basadas en las suposiciones iniciales.
    - Las justificaciones sobre las suposiciones anteriores.
    - Características comunes relevantes de usuarios pertenecientes a subgrupos de usuario específicos del sistema aplicación (llamados *estereotipos*).
    - Estereotipos que se puedan obtener por generalización de las historias de interacción de muchos usuarios.
    - Clasificación de usuarios pertenecientes a uno o más subgrupos de usuarios para poder integrar las características típicas de estos subgrupos en el modelo de usuario individual actual.
    - El comportamiento del usuario para ser almacenado, particularmente su interacción pasada con el sistema.

Por otro lado, los sistemas servidor de modelado del usuario/estudiante son componentes software centralizados que ofrecen sus servicios a varias aplicaciones en paralelo. Comparados con componentes embebidos de modelado del usuario, los anteriores proporcionan ventajas importantes: la información se mantiene y se procesa en un repositorio central o integrado virtualmente; la información del usuario/estudiante adquirida por una aplicación puede ser empleada por otras aplicaciones; es más conveniente actualizar la información del modelado del usuario/estudiante y liberar a los clientes de las tareas de modelado del usuario/estudiante, etc.

- *Sistemas de modelado del estudiante descentralizados.* Este tipo de modelado fue descrito por Vassileva y otros y se llama *modelado activo del estudiante* (Mccalla et al. (2000)). Es una aproximación que consiste en una multitud de modelos del estudiante desarrollados y mantenidos por una variedad de agentes software en el contexto de entornos multiagente distribuidos en el que los estudiantes forman una comunidad de aprendizaje y enfoca su atención sobre los procesos y contexto del modelado, más que en su descripción global. Además, no se trata de un ME monolítico asociado a cada estudiante, sino que los MEs están fragmentados y distribuidos a través del sistema (la aproximación se enfoca precisamente en los procesos de modelado, como recuperación, integración, o interpretación de los modelos fragmentados, más que en la representación tradicional del conocimiento). Esta aproximación descentralizada se centra en el proceso de recoger e integrar información, virtualmente distribuida, acerca del usuario/estudiante en momentos particulares y con propósitos específicos, no en recoger el mayor número de datos posibles acerca del usuario/estudiante como sucede en la aproximación centralizada.

  Esta aproximación es la propuesta por Niu. Su objetivo fue crear una representación (de procedimiento) genérica de *propósitos* para modelado del estudiante (denominado *modelado del estudiante basado en propósitos*). Con este fin, se creó una biblioteca de propósitos para el modelado (parecido a las bibliotecas de procedimiento en BUGGY). Los propósitos se organizan en diferentes niveles de generalidad y estarán asociados a rutinas para recuperar, integrar y usar los datos en el modelado del estudiante. Estos procedimientos serán recuperados y ejecutados mediante componentes distribuidas (agentes) cuando sean necesarios.

#### 2.2.5.1 Taxonomía en el modelado del estudiante basado en propósitos

Basado en propósitos significa que los modelos distribuidos y fragmentados pueden ser computados a la vez, para un propósito particular, usando sólo los datos requeridos para este propósito. Las ventajas principales son dos: Primero, el enfoque sobre propósitos puede acelerar la computación. Debido a que los agentes en el sistema sólo mantienen modelos parciales distribuidos, una integración completa de la información podría ser costosa y algunas veces imposible. Segundo, la recuperación e integración de información en un momento determinado, para un propósito particular, tendría en cuenta el contexto local y podría ser vinculada directamente a la adaptación necesitada. El modelo, por tanto, sólo tiene sentido en el contexto en el que es creado, tal como tiempo, propósito, el agente que lo creó, las fuentes disponibles, etc., para computar modelos de usuarios en el instante en que el propósito es reconocido.

El sistema en el que se aplica esta aproximación es I-Help (Bull et al. (2001), Vassileva et al. (2003)), un sistema multiagente distribuido que permite a los usuarios pedir, recibir y proporcionar ayuda síncrona y asíncronamente sin tutoría inteligente (Figura 2.10).

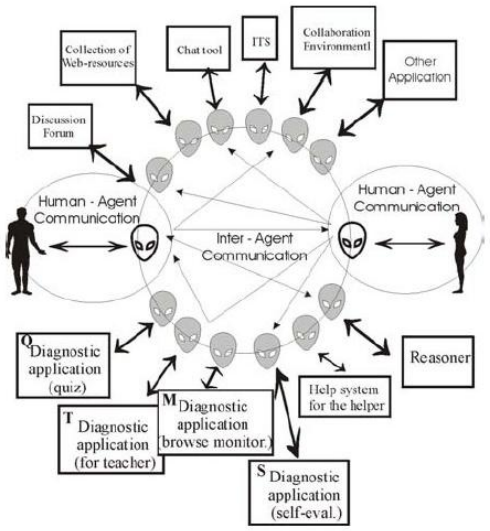


**Figura 2.10:** Arquitectura multi-agente de I-Help (extraído de Vassileva et al. (2003))

En la figura, las caras grises representan agentes personales. Las aplicaciones se representan como cajas rectangulares. Las cuatro cajas que incluyen letras mayúsculas en la esquina superior izquierda (Q, T, M, y S) son diferentes aplicaciones de diagnóstico encargadas de crear y actualizar modelos de usuario en dominios particulares.

Cada estudiante tiene su propio agente personal en el que no sólo están almacenadas las características e información del propietario sino también información fragmentada de otros estudiantes que han contactado con el propietario del agente antes. Hay dos componentes en I-Help: un componente privado uno-a-uno que proporciona ayuda, consulta y asistencia en interacciones no preestablecidas entre dos individuos, tal como tutoría; y un componente de ayuda y foros de discusión pública que permite a los estudiantes preguntar y responder a cuestiones sobre una variedad de tópicos. El modelado del estudiante en este entorno puede servir para varios propósitos: selección de un sistema de ayuda para propósitos de equiparación, evaluación de los estudiantes por los profesores, valoración propia por el estudiante, reflexión por el estudiante, gestión del conocimiento por el diseñador de un curso, etc.

Siguiendo el enfoque descentralizado, los MEs en I-Help están representados y almacenados de forma distribuida y computados cuando son requeridos.

En cuanto a la taxonomía del conocimiento en el ME (Bull et al. (2001)), varios tipos de información puede ser modelados en el sistema I-Help: conocimiento del usuario en diversos tópicos, interés, estilo cognitivo (clasificación basada en la de Riding y Scheema, Riding and Cheema (1991)), entusiasmo en participar (medido en función de la frecuencia de conexión en I-Help, número de correos leídos y re-leídos en las discusiones públicas, velocidad en el tiempo de respuesta a peticiones de ayuda en discusiones públicas y privadas, etc.), preferencias de interacción en discusiones privadas (indicadas a través de características que los usuarios buscan en un ayudante, máximo número de sesiones de ayuda concurrentes en las que puede participar, etc.), preferencias de personas para la interacción así como personas prohibidas (con las que no se desea interactuar), datos sobre la amabilidad del usuario (los usuarios pueden registrar opiniones sobre la amabilidad del ayudante en las discusiones privadas, por ejemplo, cuando

un ayudante potencial rechaza ofrecer ayuda o, el tipo de participación en las discusiones públicas como, por ejemplo, si ofrecen respuestas y si estas respuestas son leídas frecuentemente y confirmadas. En las discusiones públicas, los usuarios tienen mecanismos para votar también la utilidad de los correos).

Para representar los contextos del sistema I-Help, así como los propósitos se usa la *jerarquía de propósitos*. Un propósito puede verse como un paquete de datos y procesos. Cuando se propone un propósito, sus paquetes son añadidos en el entorno del programa actual para que sus procesos tengan acceso directo a lo que necesiten conocer, sin requerir el acceso a todo el conocimiento del sistema completo. Además, los propósitos se pueden organizar en jerarquías, de tal modo que el sistema puede verse en muchos niveles, desde muy general a muy específico (figura 2.11). Los propósitos en la jerarquía son unidos mediante relaciones *abstracción-refinamiento* para el movimiento de propósitos de grano más fino a más grueso. Los propósitos específicos heredan información y procedimientos de los propósitos más generales anteriores a ellos en la jerarquía. De manera similar a las clases en Programación Orientada a Objetos, un propósito específico puede desactivar o proporcionar excepciones (redefinir) la información/procedimientos definidos en un propósito más general. La estructura de la jerarquía de propósitos es similar a la de la jerarquía de granularidad dej McCalla y Greer (McCalla and Greer (1994)).

En cada propósito, se almacenan tres clases de información: entradas (denotan el tipo de datos del dominio que son relevantes para el propósito dado), funciones (conjuntos de algoritmos computacionales usados por un agente computacional particular para, a partir de las entradas y dentro de las restricciones de los recursos disponibles, generar las salidas deseadas) y salidas (MEs parciales que son útiles, fiables y apropiados para el propósito).

Los propósitos más generales se denominan *propósitos de más alto nivel*, y los más específicos *propósitos de más bajo nivel*. Ambos podrían computar información procedente de los datos puros (en la figura 2.11, la línea discontinúa especifica que hay más niveles entre ellos). Los datos puros denotados por $R_1 \ldots R_n$ en la figura 2.11 pueden ser recuperados desde el entorno. Una fuente de datos puros pueden los propios usuarios que proporcionan alguna información de perfil antes de usar el sistema o los datos de usuario almacenados por ciertas aplicaciones, tales como información de conexión almacenada en una base de datos, etc. Los datos puros pueden ser vistos como MEs sencillos creados por varios agentes o aplicaciones y almacenados de forma distribuida. Esta información del usuario puede ser reutilizada para propósitos diversos cuando alguna tarea específica de modelado del estudiante surja.

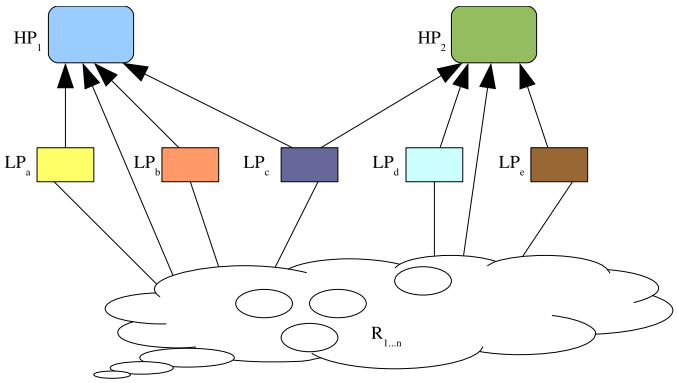


**Figura 2.11:** Jerarquía de propósitos en el sistema I-Help

En la figura 2.11 $LP_a \ldots LP_z$, representan propósitos de más bajo nivel. Se han desarrollado

algunos propósitos de más bajo nivel para I-Help relacionados con ciertos aspectos de un estudiante. Por ejemplo, un LP podría ser cómo se valora la reputación de un estudiante A. Las entradas de este LP incluirían datos acerca de cómo muchas personas han dado a A un premio y/o descuentos y cómo ha sido amonestado por personas. Las funciones en LP deberían ser reglas de producción usadas para calcular la reputación de A de acuerdo a tales entradas. La salida es el grado de reputación de A valorado como bajo, medio o alto. Los propósitos de más alto nivel (denotados por $HP_1 \ldots HP_n$) desplazan el enfoque de grano más fino a grano más grueso con menores restricciones. Pueden servir para categorías estadísticas, para propósitos de modelado abierto del estudiante, etc. Por ejemplo, uno de estos propósitos puede ser que el profesor quiera comparar el nivel de actividad del estudiante con el de su grupo para evaluar al estudiante, o que un estudiante pueda preguntar al sistema cómo le perciben otras personas como ayudante, o cómo su actividad a través de las preguntas/respuestas se relaciona con el nivel de actividad de un grupo dado para reflejar su desarrollo.

Los MEs parciales, como salidas de los diversos propósitos, capturan particulares puntos de vista de los estudiantes en varios tamaños de grano. Son mantenidos por cada agente que invocó este propósito pero son distribuidos y fragmentados a través del sistema. Estos modelos parciales pueden ser usados como datos puros o como entradas para otros propósitos.

A pesar de que el modelado basado en propósitos es útil para sistemas como I-Help y que tiene como ventaja fundamental el permitir modelado del estudiante distribuido, hay muchos temas al respecto abiertos a investigación en este momento, tales como, decidir claramente qué información es relevante para un propósito particular o como dar sentido a datos posiblemente inconsistentes e incluso contradictorios entre distintos agentes (cada agente puede tener distintas visiones de un mismo estudiante en distintas situaciones), cómo interpretar modelos creados por otros agentes, o si el modelo creado para un propósito específico debería ser mantenido por cada agente o no. Parece que la creación de una taxonomía de propósitos y la construcción de una jerarquía de propósitos genérica, concebida para muchos dominios, son trabajos abiertos a investigaciones futuras. Actualmente, se ha realizado una taxonomía de propósitos para soportar modelado del usuario activo en un dominio concreto; un sistema de gestión de cartera de inversiones (Xiaolin et al. (2004)).

### 2.2.6 Taxonomía del modelo GET-BITS

GET-BITS (*G*eneris *T*ools for *B*uilding *ITS*s) (Jerinic and Devedzic (2000)) es un modelo de SITs que usa una aproximación orientada a objetos. Es una extensión específica del modelo más general de sistemas inteligentes denominado OBOA (Devedzic et al. (1999)). Este modelo es una abstracción unificada de diferentes formalismos de representación del conocimiento (reglas, marcos, lógica, redes semánticas, etc.) y diferentes modelos de conocimiento humano. En términos de análisis y diseño orientado a objetos, OBOA define una clase abstracta *K-element* para representar un elemento de conocimiento abstracto, sin importar la complejidad del elemento o su propósito. Tanto las técnicas de representación del conocimiento como también las estrategias de resolución de problemas (tales como tareas genéricas, clasificación heurística, etc.) o conceptos de más alto nivel y agentes (por ejemplo, planificadores, scripts, multiagentes, pizarras, etc.) pueden también definirse como elementos de conocimiento más específicos, simples o agregados. De este modo, se puede establecer una jerarquía de tipos de conocimiento que se necesiten en un sistema particular y diseñar un diagrama de clases que parta de K-element, como la clase más abstracta en la raíz (esto es análogo a la clase *Object* en la raíz de las jerarquías de clases de *Smalltalk* y Java y el resto de clases derivadas directamente o indirectamente de ésa

clase).

Las clases para representar el conocimiento no son las únicas herramientas requeridas para construir SIT orientado a objetos. También se precisan algunos objetos de control que conecten funcionalmente los módulos del sistema, manejen mensajes, controlen cada sesión en el sistema, monitoricen las reacciones del estudiante, etc. GET-BITS también especifica clases de estos objetos de control (ver figura x.9). Por ejemplo, muchas clases se derivan de la clase *Frame* como *Lesson* y *Topic* (lecciones y tópicos, respectivamente, del usuario que aprende), *TQ* (una clase abstracta usada para derivar las clases que representan las preguntas que el estudiante tiene que responder, problemas que el sistema genera para él/ella, etc.) o *Explanation* (explicaciones que el sistema genera en respuesta a varias clases de usuarios (usuarios finales y desarrolladores del sistema), así como tópicos y explicaciones orientadas a conceptos). Los nombres del resto de clases pueden ser fácilmente interpretados. Además de estas clases, para diseñar e implementar SITs orientados a objetos incluye las clases para representación de conocimiento, para especificar objetos de control, y para representar conceptos específicos de SITs (por ejemplo, modelos del estudiante y estrategias pedagógicas). Todas las clases se implementan en una biblioteca de clases de forma genérica para usarlas en el diseño e implementación de shells de desarrollo de SITs o herramientas de autoría. El modelo GET-BITS se detalla en [DEVE00].

En OBOA, se diferencian entre cuatro tipos de conocimiento: conocimiento de dominio (clase *D_conocimiento*), conocimiento de control (clase *C_conocimiento*), conocimiento explicatorio (clase *E_conocimiento*) y conocimiento de sistema (clase *S_conocimiento*). El conocimiento temporal y conocimiento de mantenimiento de la verdad se tratan como parte de conocimiento de control en OBOA.

Otros aspectos interesantes en GET-BITS es que se definen 5 niveles de abstracción para diseñar SITs y si fuera necesario se podrían definir subniveles de grano fino en cada nivel de abstracción. Cada nivel tiene asociado conceptos, operaciones, técnicas de representación del conocimiento, métodos de inferencia, herramientas y técnicas de adquisición del conocimiento y herramientas de desarrollo que se pueden considerar como dimensiones en las que pueden ser analizadas cada nivel [DEVE00]. Al igual que la idea anterior -y tomado también del campo de la ingeniería del software, GET-BITS usa patrones de diseño (descripciones de soluciones satisfactorias para problemas comunes que suceden reiteradamente en el diseño software al producir aplicaciones en un contexto particular). Los patrones de diseño incrementan de este modo la eficiencia en el proceso de diseño al construir sistemas de tutoría inteligentes haciendo además los SITs más reusables, flexibles y robustos. Estos patrones se usan en GET-BITS al desarrollar las bibliotecas de clases que soportan la construcción de SITs basados en GET-BITS. El uso de uno de estos patrones puede verse en [DEVE00].

El modelo GET-BITS se ha usado en el desarrollo de FLUTE, SIT en el dominio de lenguajes formales y autómatas cuya arquitectura se muestra en la figura x.10. Este sistema permite la introducción sistemática de los estudiantes en el dominio del sistema, de acuerdo con la estructura lógica del dominio, el conocimiento previo individual y capacidades de aprendizaje de cada estudiante [DEVE98].

### 2.2.7 Taxonomía del conocimiento en la ontología OMNIBUS

En el marco del proyecto OMNIBUS[1], Mizoguchi et al. han propuesto la ontología del mismo nombre basada en la consideración filosófica de todos los conceptos necesarios para la comprensión del aprendizaje, formación y diseño formativo. Además, basado en esta ontología, han

[1] http://edont.qee.jp/omnibus/doku.php

desarrollado un prototipo de sistema de autor (SMARTIES) que pretende facilitar una mejor relación entre el comportamiento del sistema y la teoría de aprendizaje/formación de fondo. Estos sistemas son denominados sistemas de autor *theory-aware*.

Uno de los objetivos principales con la construcción de la ontología OMNIBUS es cubrir diferentes teorías y paradigmas sobre diseño de aprendizaje y diseño formativo (Mizoguchi et al. (2007)). Actualmente, la ontología sólo está enfocada, como se verá a continuación, en el diseño abstracto de contenidos de aprendizaje sin relacionarse aún con conocimiento de dominio u objetos de aprendizaje que concreten el diseño abstracto, aunque es uno de los planes futuros del proyecto al que pertenece.

Uno de los soportes esenciales de la aproximación ontológica de OMNIBUS es la relación entre teorías y diseño formativo acorde a la estructura anidada de la figura 2.12. La parte inferior de la estructura es el mundo del aprendizaje. Las teorías de aprendizaje explican los procesos y eventos en el mundo. Encima de él está el mundo formativo. El proceso formativo influye o facilita el proceso de aprendizaje. Las teorías formativas recomiendan el proceso formativo efectivo para que el proceso de aprendizaje obtenga los resultados deseados. El proceso formativo sucede en paralelo con el proceso de aprendizaje. Asimismo, el mundo del diseño formativo esta encima del mundo formativo. Una teoría de diseño formativo recomienda el proceso racional para diseñar el proceso formativo. Los dos procesos inferiores son procesos del mundo real y el superior es una planificación o proceso de diseño de los procesos/eventos del mundo real. Una característica esencial de los tres procesos es que dependen del proceso de aprendizaje que puede ser modelado como un cambio de estado en un estudiante.

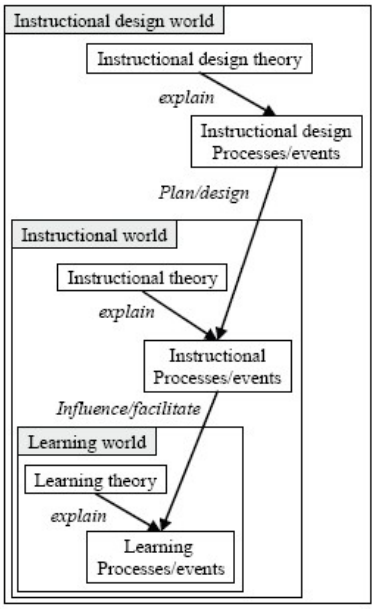


**Figura 2.12:** Estructura anidada de aprendizaje, instrucción y diseño formativo. Extraído de Hayashi et al. (2006)

Aunque todas las teorías educativas tienen cierta bases comunes para explicar el aprendizaje, para cada paradigma, el mecanismo en el desarrollo de conocimiento es distinto pero la idea de *estados* en el proceso de aprendizaje es común. Basándose en ello, la aproximación de ingeniería propuesta por los autores adopta como hipótesis de trabajo que debe estar basada en los estados; se trata de conceptualizar .Aprendizaje.en términos de cambio de estado de los

estudiantes (Mizoguchi and Bourdeau (2000)).

#### 2.2.7.1 Conceptualización en la ontología OMNIBUS

Una de las características de la ontología OMNIBUS es la conceptualización de los procesos de aprendizaje/formación desde dos puntos de vista: *qué alcanzar* y *cómo alcanzarlo* definidos como los conceptos denominados *I_L event* y *WAY* descritos a continuación (Mizoguchi et al. (2007)):

- *Conceptualización de la interacción entre aprendizaje y formación.* Los autores han definido un concepto denominado *I_L event* (figura 2.13) para unir *eventos formativos* con *eventos de aprendizaje*. Un *evento de aprendizaje* es considerado compuesto de *cambio de estado* y *acción de aprendizaje* de tal modo que, las acciones de aprendizaje causan el cambio de estado del estudiante. Por otro lado, un *evento formativo* está compuesto de una *acción formativa* que influye en los eventos de aprendizaje.

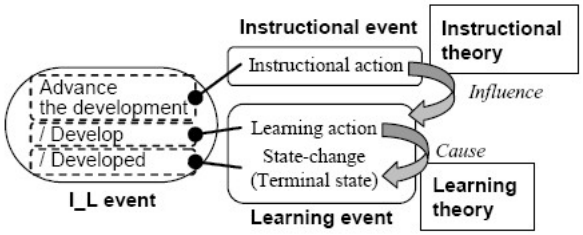


**Figura 2.13:** *I_L event.* Extraído de Mizoguchi et al. (2007)

- *Conceptualización de la estructura abstracta de escenario formativo/aprendizaje.* Un escenario es modelado en la ontología como una estructura de I_L eventos para alcanzar un cierto cambio de estado de un estudiante y que es denominado por los autores Ï_L event decomposition tree". La idea básica es relacionar con él un macro *I_L event* con uno inferior (micro) que colectivamente alcance el *I_L event* superior (macro) como una forma de logro del cambio de estado de un estudiante (referido como *WAY*). En la figura 2.14 se presenta un ejemplo de esta conceptualización; hay dos *WAY*s (WAY1 y WAY2) para alcanzar el macro *I_L event* consistente en introducir un contenido para hacer que un estudiante lo reconozca. WAY1 (basado en la teoría de Gagne y Brigg Gagné et al. (1989)) presenta lo que hay que aprender y luego proporciona guías. WAY2 está basado en la teoría de Collins (Collins et al. (1989)) y proporciona sólo demostraciones, no explicaciones. En este último caso, el macro *I_L event* no se descompone, es concretizado. Podría decirse que ambos caminos tienen la misma meta pero alcanzada mediante diferentes estrategias.

- *Conceptualización de estrategias sugeridas por teorías formativas/aprendizaje.* Las teorías recomiendan estrategias para planificar el proceso formativo y de aprendizaje de acuerdo a situaciones supuestas. En el marco de modelado de OMNIBUS, estas estrategias son modeladas como un *WAY* denominado *WAY-knowledge*. Los autores han organizado cerca de 100 elementos de *WAY-knowledge* basándose en algunas teorías (Gagne, Merrill, Collins, Jonassen, etc.) y son definidos como conceptos relacionales como se describe en la sección siguiente 2.2.7.2. Mediante esta organización los autores han pretendido clarificar la estructura conceptual de cada teoría y sus diversas guías de diseño.

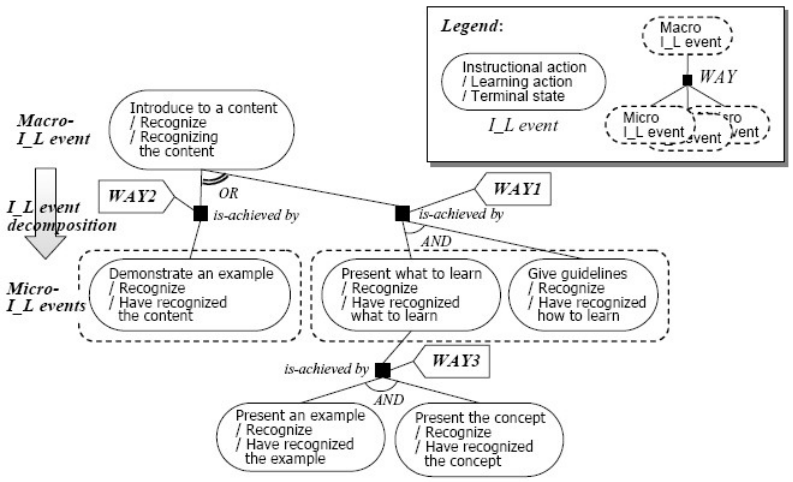


**Figura 2.14:** Ejemplo de una descomposición de *I_L event*s mediante *WAY*s. Extraído de Mizoguchi et al. (2007)

La tabla 2.5 muestra la clasificación de propiedades que son comunes a los escenarios de aprendizaje/formación y modelos, a los elementos *WAY-knowledge* y a los conceptos de teorías en la ontología OMNIBUS. De este modo, usando estas propiedades se pretende que una herramienta de autor pueda sugerir a los autores teorías que tienen las mismas propiedades que un escenario o proporcionar también a los autores la concordancia de propiedades entre un escenario y una teoría como justificación de un escenario.

**Tabla 2.5:** Clasificación no exhaustiva de propiedades. Extraído de Mizoguchi et al. (2007)

| Categories | Properties | values |
|---|---|---|
| **Learner characteristics** | - Age (type)<br>- Language<br>- Prior knowledge: fact, concept, rule | - child, adult<br>- Japanese, English, French<br>- learned, not learned |
| **Domain/topic characteristics (what to learn, content)** | - Concreteness<br>- Complexity<br>- Causality<br>- Prerequisiteness | - concrete, abstract<br>- simple, complex<br>- causal, not causal<br>- prerequisite, not prerequisite |
| **Context characteristics** | - Context of learning<br>- Testing<br>- Instruction mode<br>- Delivery mode | - School, workplace, university<br>- summative, formative, Assessment, certification<br>- individual, group, community<br>-classroom, distance, distributed |
| **I_L event characteristics (scenario)** | - Event kind<br>- Authenticity<br>- Interaction kind | - I_L, assessment<br>- authentic, artificial, virtual-<br>- action, interaction, social interaction |
| **Learning object characteristics** | - Language<br>- Language level<br>- Representation mode | - Japanese, English, French<br>- child, adult<br>- text, graphics, image, video, simulation, game |

#### 2.2.7.2 Conceptos definidos en la ontología OMNIBUS

La estructura de nivel superior de la ontología OMNIBUS se muestra en la figura 2.15. La ontología está compuesta principalmente de conceptos relacionados con los mundos *Common, Cognition, Learning, Instructional, Instructional design/Instructional system design (ID-ISD)* y *Event.*

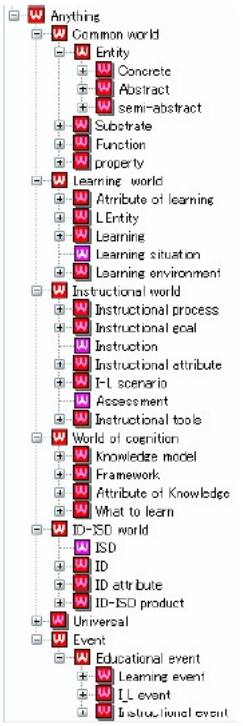


**Figura 2.15:** Estructura de nivel superior de la ontología OMNIBUS. Extraído de Mizoguchi et al. (2007)

### *Wholeness concepts*

A continuación, se describen tres conceptos fundamentales de la ontología OMNIBUS denominados *wholeness concepts*[2] definidos en el mundo común: *State* y *Action*, compartidos con el mundo de aprendizaje (*Learning world*) y con el mundo Formativo (*Instructional world*). Estos conceptos se usan para describir los procesos de aprendizaje y formativos. Otro concepto de este tipo es *Educational event*, una descripción contextualizada de proceso.

***State***: Como ya hemos mencionado, en OMNIBUS los estados (*states*) en el proceso de aprendizaje son el factor más importante para su construcción. De acuerdo a ello, se han recogido los tipos de estados a partir de diversas teorías y se han clasificado en la ontología OMNIBUS bajo una jerarquía *es-un* desde el punto de vista del significado conceptual (ver figura 2.16).

Como se puede observar, los estados en la ontología OMNIBUS se clasifican principalmente en dos tipos:

- *Internal state*: Es un estado interno de los agentes. Incluye: *Cognitive process state*, *Attitudinal state*, *Progression state* y *Developmental state*.
- *External state*: Es un aspecto del compromiso/participación del agente en una acción.

En la ontología OMNIBUS, los estados anteriores son comunes a cualquier teoría y el aprendizaje se describe mediante cambios en el estado del estudiante. La diferencia entre teorías se

[2]En el editor Hozo en el que se representa esta ontología un *wholeness concept* es un concepto de un objeto considerado como un todo (por ejemplo, coche), que está compuesto de múltiples conceptos (por ejemplo, rueda, volante, etc.).

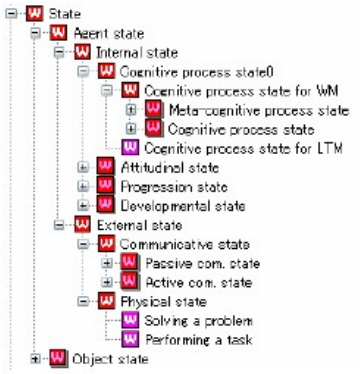


**Figura 2.16:** Jerarquía de *State*. Extraído de Mizoguchi et al. (2007)

describe como la diferencia de estados usados o no en las teorías, el proceso de cambios en el estado soportado en la teoría y la relación entre cambios de estados y acciones de aprendizaje.

***Action***: Las acciones se definen en común con aprendizaje e instrucción (figura 2.17). Todas las acciones se descomponen en subacciones y la descomposición puede repetirse casi indefinidamente aunque debería detenerse en un cierto nivel de granularidad por debajo del que las acciones de grano más fino no tienen sentido en el contexto. Las acciones de grano más fino se denominan acciones primitivas.

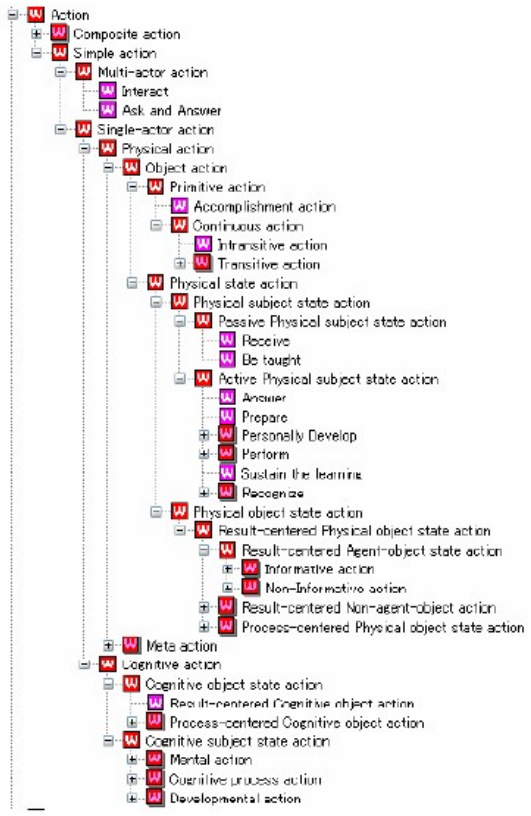


**Figura 2.17:** Jerarquía de *Action*. Extraído de Mizoguchi et al. (2007)

Hay dos clases de acciones: las acciones que tienen una única descomposición en subacciones como *andar* y otras que tienen múltiples caminos de descomposición de acuerdo al contexto/meta bajo el que las acciones están siendo ejecutadas como, por ejemplo, la acción *enseñar*. En el primer caso, las subacciones se definen usando *slots* de tipo relación *part-of*. Para representar el segundo tipo de acciones se usa *Event* y las distintas secuencias de subacciones que se pueden ejecutar para estas acciones se distinguen usando en vez de *part-of*, *WAY-knowledge* (forma de descomposición). Por lo tanto, los eventos se refieren a *Action*s en situaciones particulares que requieren a su vez acciones particulares para alcanzar sus metas. Tipos de eventos son *learning event* e *instructional event*.

- *Primitive action*: Esta acción cambia el *Comunicative state* de la persona que la realiza o del objeto. En la ontología OMNIBUS, estas acciones no pueden descomponerse en subacciones; son acciones primitivas.
- *Physical state action*: Estas acciones también cambian el *Communicative state* de la persona que la realiza o del objeto destino, como *Primitive action*, pero a diferencia de éstas últimas, pueden descomponerse en subacciones para alcanzar el cambio de estado esperado en la acción. Esta descomposición se define como un *WAY* descrito posteriormente (sección 2.2.7.2). Ejemplo: La clase *Inform* define la acción *Informar* como el cambio de estado del destinatario de esta acción, que es un Agente, al estado de *Informed* que es una subclase de *Communicative state* (figura 2.18).

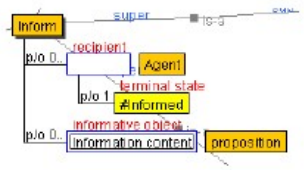


**Figura 2.18:** Definición de la acción *Inform* en el editor Hozo. Extraído de Mizoguchi et al. (2007)

- *Cognitive (state) action*: Esta acción cambia el estado interno de la persona que la realiza o del objeto. Esta acción puede descomponerse en subacciones para alcanzar el cambio de estado esperado en la acción. Ejemplo: *Recall* define la acción recordar que cambia el estado de la persona que realiza esta acción al estado de *Have recalled*, subclase de *Internal state*.

***Educational event***: Su jerarquía *es-un* se muestra en la figura 2.19. Este concepto representa: eventos de aprendizaje (concepto *Learning event*), eventos de formación (concepto *Instructional event*) y relaciones entre los dos anteriores (concepto *I_L event*).

En la ontología OMNIBUS, el proceso formativo y de aprendizaje se definen como procesos con una meta relacionada con la situación. El concepto para describir este proceso es *Event* compuesto de *process* y su información contextual: *participant*, *time* y *location*. En *Educational Event*, *process* se especifica mediante *Action* que tiene también *participant*, *time* y *location*. La distinción conceptual entre *Action* y *Event* en la ontología se debe a que *Event* se define dependiente de un contexto y *Action* se define independientemente del contexto.

- *Learning event*: está compuesto de un agente como un estudiante, una acción de aprendizaje, sus objetos, efectos y condiciones de aprendizaje, y atributos espaciales/temporales. La relación entre ellos es sugerida por las teorías de aprendizaje.

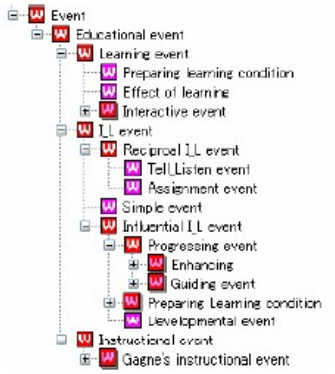


**Figura 2.19:** Jerarquía de *Educational event*. Extraído de Mizoguchi et al. (2007)

- *Effect of learning*: significa lo que puede decirnos una teoría de aprendizaje sobre el efecto esperado después de esta acción de aprendizaje. Este significado es descrito por la relación *Action result* entre *Learning action*, *Learning effect* y *Learning theory* (figura 2.20).

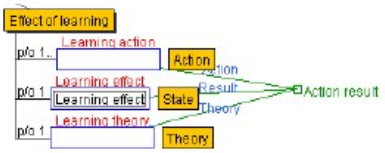


**Figura 2.20:** Jerarquía de *Educational event*. Extraído de Mizoguchi et al. (2007)

- *Preparing learning condition*: significa que cuando las condiciones de aprendizaje se satisfacen, la teoría de aprendizaje asegura que la acción de aprendizaje debería tener éxito. Este significado es descrito por la relación *Guarantee* entre *Learning action*, *Learning condition* y *Learning theory* (figura 2.21).

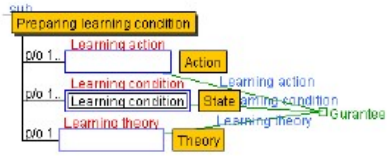


**Figura 2.21:** Jerarquía de *Educational event*. Extraído de Mizoguchi et al. (2007)

- *Instructional event*: está compuesto de un agente como un instructor, una acción formativa, sus objetos y atributos espaciales/temporales. La definición de este evento se especifica independientemente de la definición de *Learning event*. Por lo tanto, un *instructional event* no incluye cómo la formación influye en el aprendizaje. El efecto de la instrucción sobre el aprendizaje se describe dentro de *I_L event*, descrito a continuación.
- *I_L event*: define la relación entre un *Learning event* y un *Instructional event*, es decir, describe como un evento formativo contribuye a un evento de aprendizaje. Esta relación es

definida desde dos puntos de vista: la contribución de los *Instructional event*s en el cambio del estado de un estudiante y la preparación para el siguiente *Learning event*.

***Relational Concepts***

De acuerdo a la terminología del editor Hozo, en la ontología OMNIBUS se definen los siguientes tipos de conceptos relacionales:

- *Pure relational concepts*: Es un concepto para definir una relación entre slots de un concepto completo (*wholeness concept*). En la ontología OMNIBUS, los conceptos relacionales[3] son *pure relational concepts* a excepción de las subclases de WAY . Ejemplos de estos conceptos puros en la ontología son: *less-than*, *same as*, *Influence*, etc.

- *WAY* y *WAY-knowledge*: *WAY* es un concepto relacional que describe una forma de obtención del cambio de estado de un estudiante. Esta relación no se usa para definir la relación entre slots de un *wholeness concept*, sino que es una relación entre un *I_L event* superior (macro) y uno o varios *I_L event* inferiores (micro) que alcanzan el superior. Es decir, *WAY* es una descripción de una estrategia educativa (de aprendizaje o formativa). Esta estrategia puede adaptarse a situaciones de aplicación concreta mediante *WAY-knowledge* . Por lo tanto, *WAY-knowledge* es un concepto específico de *WAY* basado en teorías, aunque se está planificando extender esta definición también a conocimiento empírico. El estudio de las teorías/modelos de aprendizaje/formativos considerados, las categorías de estrategias educativas consideradas así como la relación entre ambas, teorías y estrategias, y su conceptualización se sale del ámbito del modelado del estudiante en el que se enfoca este trabajo pero se detalla ampliamente en Hayashi et al. (2008), Hayashi et al. (2009). Asimismo, ejemplos de *WAY-knowledge* de la ontología OMNIBUS en el editor Hozo pueden verse en Mizoguchi et al. (2007).

Basado en la ontología OMNIBUS, se ha desarrollado SMARTIES (*SMART Instructional Engineering System*), un prototipo de sistema de autor *theory-aware* y conforme a los estándares. Este sistema comprende teorías de aprendizaje/formativas y da soporte a los autores para crear escenarios que se ajustan a teorías desde el nivel abstracto al nivel concreto, es decir, desde las metas establecidas de un escenario hasta la asignación de objetos de aprendizaje a ellas. A la vez, es un sistema conforme a los estándares que puede exportar el escenario o modelo resultante en formato IMS LD (4).

SMARTIES ayuda a dos tipos de usuarios: a) autores de escenarios, que incluye diseñadores formativos, profesionales de la educación y menos frecuentemente estudiantes y b)autores de conocimiento, que incluyen investigadores y teorizadores. El ámbito del soporte se limita a la fase de diseño del proceso de diseño formativo más que a las fases de análisis y desarrollo.

La figura 2.22 muestra un diagrama de bloques de SMARTIES. Un autor de escenario crea un modelo de proceso formativo y de aprendizaje particular usando la interfaz de autor. El *Model manager* gestiona el modelo que crean los autores de escenarios y les proporciona unas guías para crear el modelo del proceso formativo y de aprendizaje basándose en la ontología, incluyendo los conceptos y el vocabulario que los representa y su estructura básica. De este modo, se genera un modelo de escenario declarativo de estrategias formativas/aprendizaje como *WAY-knowledge*s de la ontología OMNIBUS. Esto permite a SMARTIES proporcionar las guías teóricas

[3] *Relational concept* en el editor de ontologías Hozo es la relación conceptualizada entre múltiples conceptos (normalmente dos). Por ejemplo, la relación de fraternidad entre dos personas.

para la construcción de escenarios basándose en el conocimiento declarativo definido en la ontología OMNIBUS instalada fuera de SMARTIES a diferencia de otros sistemas de autor en los que las teorías están embebidas de una manera procedimental. Por lo tanto, SMARTIES puede automáticamente cambiar su comportamiento cuando se actualiza el conocimiento de diseño de aprendizaje/formativo incluido en la ontología OMNIBUS.

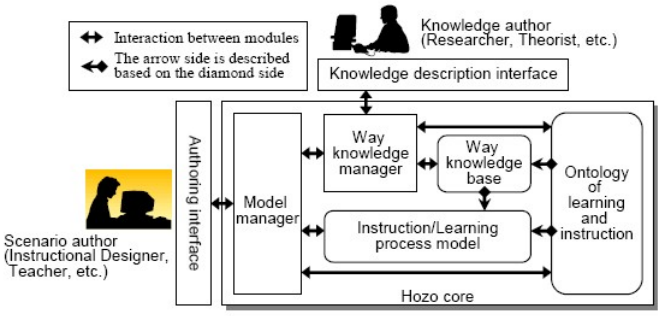


**Figura 2.22:** Diagrama de bloques del sistema SMARTIES. Extraído de Mizoguchi et al. (2007)

La figura 2.23 muestra la interfaz de usuario de SMARTIES y cómo un autor crea un escenario formativo y de aprendizaje usando *WAY-knowledge*s. El editor de escenario proporciona (figura 2.23, (A)) a los autores un entorno para describir un Ï_L event descomposition treeçomo un modelo de escenario formativo y de aprendizaje. Un *I_L event* es representado como un nodo y la descomposición de cada nodo es representada como un árbol cuya raíz es el *macro-I_L event* con unos pocos nodos como *micro-I_L event*s. En esta ventana, un autor descompone las metas de aprendizaje del escenario paso a paso eligiendo *WAY-knowledge*s aplicables. El visor WAY-knowledge (figura 2.23, (E)) visualiza partes de *WAY-knowledge*s aplicables apropiados para el *I_L event* que el autor quiere descomponer. Esto se realiza mediante *pattern matching* basado en la ontología OMNIBUS. Cuando el autor elige uno de ellos, la descomposición propuesta se visualiza en el visor y se muestra la explicación (figura 2.23, (F)). Si el autor decide adoptar el *WAY* seleccionado, la propuesta se aplica en la ventana principal. Para que el autor cree el modelo de proceso formativo y de aprendizaje, el proceso mencionado anteriormente se repite. De este modo, el autor se mueve de los niveles abstractos a los niveles concretos.

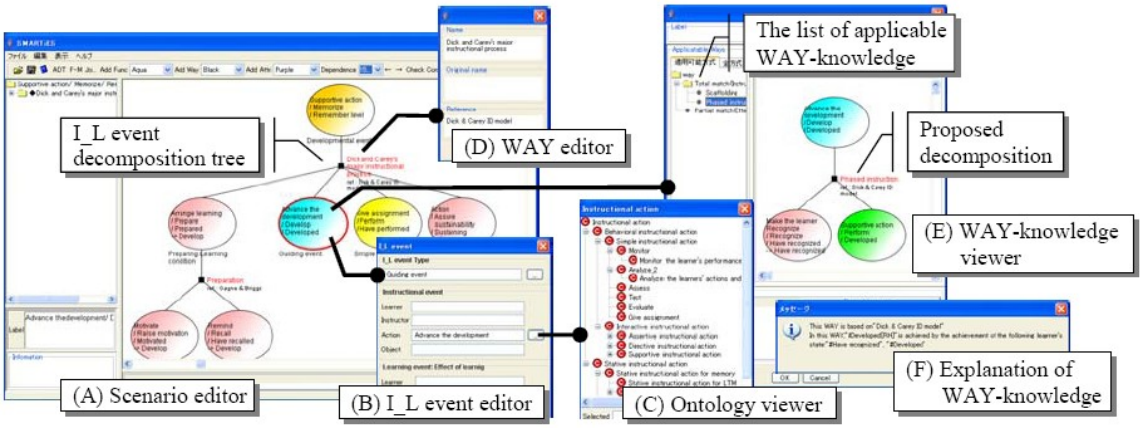


**Figura 2.23:** Interfaz de usuario de SMARTIES. Extraído de Mizoguchi et al. (2007)

Una explicación detallada de las características y de la interfaz de SMARTIES se presenta en Mizoguchi et al. (2007) y Hayashi et al. (2008).

## 2.3 Diagnóstico Cognitivo

Hay un gran número de métodos de DC que se han desarrollado para SIT (Clancey (1986), Wenger (1987), Dillenbourg and Self (1992)) pero que no han sido encajados en un marco teórico general. Otros trabajos (**?**) han intentado aplicar la teoría de diagnóstico en Inteligencia Artificial (IA) de Reiter (1987) para mantener la consistencia en un ME. Sin embargo, no hay en estos métodos un intento general de relacionar los métodos de DC con los desarrollados en el campo del diagnóstico en IA (Reiter (1987), de Kleer and Williams (1987), Struss and Dressler (1989), etc). Self (Self (1992), Self (1993)) ayuda a definir mejor la naturaleza del problema del DC estudiándolo como un caso particular del diagnóstico general en IA. Su estudio es el origen de una serie de métodos que surgieron para abordar la problemática del DC, tal y como se verá a continuación. Por otro lado, hay escasas aproximaciones (Burton (1982)), Ikeda et al. (1993)) que tengan como una característica fundamental el intentar formular la no monotonía del proceso de modelización del estudiante y que, por lo tanto, guíen realmente en la tutoría a los estudiantes hacia una mejor comprensión de la materia que se enseña.

En los siguientes apartados se describen las aproximaciones existentes para DC. En primer lugar, las aproximaciones para diagnóstico en IA, comparándolas con las aproximaciones desarrolladas para DC (Self (1993)).

### 2.3.1 Diagnóstico en IA

Los métodos de diagnóstico en IA se pueden clasificar en dos tipos (Self (1992)):

- *Métodos basados en reglas*. Estos métodos utilizan asociaciones empíricas entre síntomas y diagnósticos, como las usadas en sistemas expertos. Aunque son efectivos en casos específicos, han sido muy criticados, por ejemplo, por su excesiva especialización (no se pueden aplicar a otros sistemas de forma fácil y económica), por estar restringidos a fallos específicos (por no tener bases analíticas) y por tener un nivel de competencia desconocido.

- *Metodos basados en modelos*. Estos métodos usan descripciones de componentes y descripciones de la estructura de un dispositivo. Son independientes de las características del dispositivo e intentan solventar las dificultades de los métodos basados en reglas.

Si se compara el diagnóstico en IA con el DC podemos ya reconocer una similitud entre ambos (Self (1993)). Muchos métodos de DC usan catálogos de errores que son listas o bases de reglas que asocian errores del estudiante con conceptos erróneos específicos, por ejemplo, el método de modelo de traza de Anderson, Boyle, Corbett y Lewis (Anderson et al. (1990)). En este método la principal dificultad es el enorme tiempo que se requiere para construir un segundo catálogo de errores al pasar desde el dominio inicial (por ejemplo, programación en Lisp) a otro dominio (por ejemplo, álgebra).

Los tratamientos formales basados en modelos del diagnóstico en IA o *diagnóstico basado en modelos* (DBM) se realizan mediante mecanismos de mantenimiento de la verdad (de Kleer and Williams (1987)) o lógica por defecto (de Kleer and Williams (1987)). En los sistemas de mantenimiento de la verdad el sistema tiene constancia de todas las hipótesis de las que se pueden

derivar los resultados esperados; si se produce un resultado inesperado, deben retraerse sistemáticamente las hipótesis para que la salida producida por el sistema no sea inconsistente. En los sistemas que aplican la lógica por defecto, se especifica el funcionamiento de un dispositivo mediante hipótesis por defecto. Nuevamente, el problema de diagnóstico consistirá en retraer las hipótesis necesarias para explicar las observaciones obtenidas de la realidad.

Dado que el objetivo de la presente tesis es aplicar los mecanismos de mantenimiento de la verdad al Diagnóstico Cognitivo, a continuación se presenta como ejemplo el diagnóstico basado en el mantenimiento de la verdad de De Kleer y Williams que se ha convertido en un paradigma para el DBM.

### 2.3.2 Motor de Diagnóstico General y mantenimiento de la verdad

J. De Kleer y B. C. Williams (de Kleer and Williams (1987)) presentan un procedimiento de diagnóstico independiente del dominio de aplicación. El método desarrollado se denomina *Motor de Diagnóstico General* (MDG) ya que el objetivo es desarrollar un método general eficiente para diagnosticar fallos debidos a cualquier número de fallos simultáneos. Para alcanzar la necesidad de eficiencia, el MDG explota las características del ATMS.

El procedimiento de diagnóstico funciona acoplado con un mecanismo de inferencia y maneja prácticamente los mismos elementos que la lógica basada en los primeros principios de Reiter. Estos elementos son:

- Un modelo del sistema objeto de diagnóstico en el que se describe la estructura física del sistema en función de los distintos elementos que lo componen. Cada tipo de elemento constitutivo se rige por ciertas normas de funcionamiento. Por ejemplo, en el dominio de los circuitos eléctricos se cumplen las leyes de Kirchhoff para las mallas de cableado, etc. De igual forma a como ocurre en la lógica basada en los primeros principios, la necesidad de diagnóstico surge al comprobar que el comportamiento del sistema difiere de su modelo. El modelo del sistema lo forman:

  - Una descripción de su estructura física.
  - Modelos para cada uno de sus componentes, ya sean dispositivos, procesos o incluso pasos de una inferencia lógica.

El diagnóstico toma la estructura física, los modelos de los componentes, un conjunto de diferencias entre el sistema y su modelo, y un conjunto de medidas realizadas y produce un conjunto de candidatos que explican las observaciones.

Esta aproximación basada en el mantenimiento de la verdad está caracterizada por presentar las diferencias entre el modelo y el sistema mismo como violaciones de hipótesis realizadas. Esta característica dota de gran flexibilidad al procedimiento de diagnóstico puesto que, las hipótesis pueden formularse dentro de cualquier dominio: circuitos electrónicos, domino del razonamiento de sentido común, etc.

#### 2.3.2.1 Síntomas, conflictos y candidatos

- *Un síntoma, descrito intuitivamente, es cualquier diferencia existente entre la predicción realizada por el mecanismo de inferencia y una observación de la realidad.*

El procedimiento de diagnóstico se guía por los síntomas. Cada uno de estos síntomas nos indica una o más hipótesis que resultan contradictorias, es decir, componentes que pueden estar fallando.

- *Un conflicto es un conjunto de hipótesis que dan soporte a un síntoma y que nos están indicando por tanto una inconsistencia.*

Ejemplo:

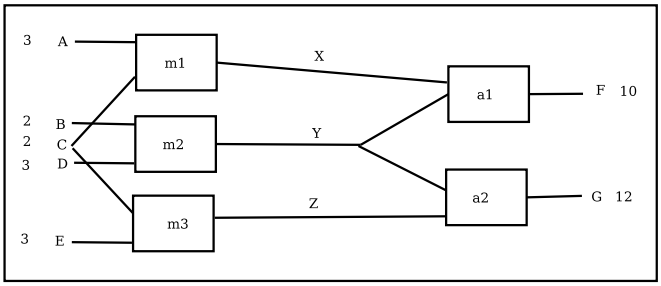


**Figura 2.24:** Circuito multiplicador-sumador

Si consideramos el circuito de la figura 2.24, en el que los elementos m1, m2 y m3 son multiplicadores y a1 y a2 son sumadores y tomamos como entradas A =3, B = 2, C = 2, D = 3 y E = 3, el resultado que obtendríamos en F debería ser 12 (A x C + B x D). Sin embargo, se constata mediante una observación que F está dando el valor 10. *"Observar que F resulta 10 cuando debería ser 12"* es un síntoma.

Siguiendo con el mismo caso, se comprueba que el resultado sobre F depende del buen funcionamiento de a1, m1 y m2. Dado que F no da el resultado esperado, al menos uno de los tres elementos debe estar funcionando mal. El conjunto <a1, m1, m2>constituye un conjunto conflicto para el síntoma anterior. Si <a1, m1, m2>es un conjunto conflicto para el síntoma, cualquier superconjunto de él lo será también. Este punto es de gran importancia y afecta directamente al rendimiento del procedimiento de diagnóstico ya que, un solo síntoma puede dar lugar a un gran número de conjuntos conflicto. El objetivo del reconocimiento de conjuntos conflicto es identificar los conjuntos conflicto minimales para un síntoma dado.

De los conceptos anteriores surge el término de candidato:

- *Un candidato es una hipótesis concreta de porqué el comportamiento del sistema difiere del comportamiento del modelo.*

En el ejemplo expuesto anteriormente, *"m2 está estropeado"* es un candidato para el síntoma observado. El objetivo último del diagnóstico es refinar el conjunto de candidatos utilizando las observaciones que hay disponibles hasta ese momento.

Un candidato se representa por un conjunto de hipótesis entre corchetes (de Kleer and Williams (1987)). Las hipótesis incluidas son falsas mientras que las que no aparecen en el conjunto son ciertas.

Como ocurría con los conjuntos conflicto, los candidatos deben ser minimales ya que, cualquier superconjunto de un candidato también es un candidato. El objetivo de la generación de candidatos es obtener el conjunto completo minimal de candidatos. El espacio de los posibles candidatos se puede representar como un retículo con la relación sub-superconjunto, en el que

los candidatos minimales definen un límite tal que todo candidato por encima de ellos es válido mientras que, cualquiera por debajo no lo es. Sin observaciones iniciales, se supone que todos los elementos del sistema funcionan correctamente, por lo que el conjunto vacío ([ ]) constituye la raíz del retículo.

En definitiva, el conjunto de candidatos se construye en dos fases:

1. Reconocimiento de conjuntos conflicto. Se utilizan las observaciones y el modelo del sistema para construir el conjunto conflicto minimal.

2. Generación de candidatos. Se utiliza el conjunto conflicto minimal para calcular el conjunto minimal de candidatos.

#### 2.3.2.2 Generación de candidatos

El diagnóstico es un proceso incremental. A medida que el diagnosticador hace observaciones o mediciones sobre el sistema se produce un refinamiento del conjunto de candidatos. La información obtenida así sirve de guía para la realización de nuevas mediciones. En una sesión de diagnóstico, el conjunto total de candidatos debe crecer de forma monótona, lo que correspondería a movernos en el espacio del retículo que forman todos los posibles candidatos, de la raíz hacia el candidato que engloba todos los componentes del sistema.

El conjunto minimal de candidatos se ve incrementado de la siguiente forma:

- Cada vez que se localiza un nuevo conflicto, todo candidato anterior que no explique el nuevo conflicto se sustituye por uno o más superconjuntos de candidatos que resultan minimales respecto de la nueva información de la que se dispone.

- Para conseguir lo anterior, se sustituye el candidato antiguo por un nuevo conjunto de posibles candidatos mínimos que contengan, cada uno, el antiguo candidato más una hipótesis del nuevo conflicto.

- Cualquier posible candidato que resulte duplicado por otro se elimina. El conjunto resultante de candidatos se añade al nuevo conjunto minimal de candidatos.

El proceso de generación de candidatos con el ejemplo expuesto anteriormente aparece detallado en de Kleer and Williams (1987).

#### 2.3.2.3 Estrategia de reconocimiento de conflictos

Podemos identificar un conflicto seleccionando un conjunto de hipótesis que conforman un entorno y comprobar si son inconsistentes con las observaciones. Si lo son, el entorno inconsistente constituye un conflicto.

No hay que confundir entorno con candidato o conflicto. *Un entorno es un conjunto de hipótesis en el que se asumimos que todos sus elementos son verdad*. Por ejemplo, asumimos que *m1* y *m2* funcionan correctamente. Un candidato es un conjunto de hipótesis en el que asumimos que todos sus elementos son falsos. Por ejemplo, m1 y m2 no están funcionando correctamente. Y un conflicto es un conjunto de hipótesis en el que al menos una de ellas es falsa.

La identificación de conflictos requiere una estrategia de inferencia C(OBS, ENV), tal que, dado el conjunto de observaciones realizadas hasta el momento (OB S), y el entorno (ENV), determine si esta combinación es consistente.

Los autores (de Kleer and Williams (1987)), aplican a esta aproximación 5 refinamientos. A continuación, como ejemplo, se muestra uno de ellos:

**Refinamiento 2:** *Monotonía y observaciones.* Sea P(OBS, ENV) el conjunto de todas las predicciones de comportamiento que podemos hacer a partir de una observación OBS dadas las hipótesis de ENV. Por ejemplo, en el caso que venimos tratando:
P ({a = 3, b = 2, c = 2, d = 3} , {a1, m1, m2}), genera {a = 3, b = 2, c = 2, d = 3, X = 6, Y = 6, F = 12}.

Dada una nueva observación M, P (OBS ∪ {M} , ENV)es siempre un superconjunto de P(OBS, ENV). Por este motivo, si somos capaces de registrar los resultados de cada P mediante un mecanismo que funcione a mode de "cache", cuando se produzca una nueva observación, únicamente tenemos que inferir el incremento que se añade al conjunto de predicciones que nos da P(OBS, ENV).

#### 2.3.2.4 Arquitectura del procedimiento de inferencia

Para aprovechar al máximo las ideas expuestas en el reconocimiento de conflictos, se supone que P cumple con los criterios básicos para utilizar mantenimiento de la verdad: puede construir una justificación por cada inferencia, y la admisión o no de una creencia depende por completo de esas justificaciones. Además, se supone que, durante el procesamiento, cada vez que se puede realizar más de una inferencia al mismo tiempo, el orden en que se realicen es irrelevante. Por último, se supone que el procedimiento de inferencia es monótono. La mayoría de los procedimientos de inferencia de la Inteligencia Artificial cumplen estos criterios. Sistemas basados en reglas, propagación de restricciones, demonios, sistemas de deducción natural y muchas formas del teorema de resolución encajan en este marco de referencia.

A cada predicción V le asociamos un conjunto de entornos ENVS(V), es decir:
ENVS (V) ≡ {env | V ∈ P (OBS, env)}

Al conjunto anterior, se le llama *entornos que dan soporte a la predicción.* Aprovechando la monotonía de la definición, sólo es necesario representar los entornos soporte minimales.

Veamos un caso concreto con el ejemplo de la figura 2.24. Si se considera la observación F = 10 y G = 12, se puede calcular Y = 6 de dos formas: Y = B x D = 6, suponiendo que m2 funciona correctamente, lo que dará lugar al entorno de soporte {m2 } o Y = G - Z = G - (C x E) = 6, suponiendo que a2 y m3 funcionan correctamente.

Por tanto, los entornos soporte para Y = 6 son { {m2}, {a2, m3}}. Cualquier conjunto de hipótesis utilizado para derivar Y = 6 es un superconjunto de estos dos entornos.

Aplicando las ventajas de la representación mediante justificaciones, se consigue que una inferencia no se realice dos veces. Si los entornos soporte de una predicción cambian, los entornos soporte de sus consecuentes se actualizan automáticamente siguiendo las dependencias marcadas por las justificaciones que se crearon cuando la regla en cuestión se ejecutó por primera vez. No es necesario, por lo tanto, volver a ejecutarla.

### 2.3.3 Estudio de John Self: aplicación del diagnóstico en IA al diagnóstico cognitivo

Las teorías de Reiter (Reiter (1987)) y de de Kleer y Williams (de Kleer and Williams (1987)) intentan proporcionar un marco general de trabajo para el diagnóstico. Por este motivo, si el DC es realmente un tipo de diagnóstico, debería también estar contemplado en este marco de trabajo (Self (1993)) donde casi todo el trabajo considerado está relacionado con el diagnóstico de dispositivos (DD), es decir, la determinación de componentes defectuosos en el diseño de un dispositivo. Como se verá a continuación, Self trata de aplicar las técnicas basadas en modelos del DD al problema del diagnóstico del conocimiento de un estudiante en la resolución de un

problema. Aunque existe una analogía superficial con el problema de identificar conocimiento defectuoso del DC (errores y conceptos equivocados) en el conocimiento de un estudiante, hay diferencias entre el DD y el DC que hacen difícil la aplicación de las técnicas del DD al DC.

Antes de continuar, veamos una serie de definiciones que se utilizan habitualmente (Self (1993)):

- *Descripción de un sistema*: se especifica mediante la estructura de sus componentes, es decir, cómo están conectadas sus componentes, y por el comportamiento de las mismas.
- *Modo o estado de un elemento*: cada componente del sistema tiene asignado un estado de entre los que constituyen el conjunto de modos o estados del elemento (por ejemplo: defectuoso y no-defectuoso).
- *Observación*: Conjunto de literales que describen medidas.
- *Evidencia*: Conjunto de observaciones.
- *Candidato*: Asignación de un estado (o modo) a cada componente de un sistema.
- *Diagnóstico para una evidencia*: Conjunto de candidatos tal que, para cada observación y cada candidato, la unión de la descripción del sistema, el candidato y la observación es consistente.

**Ejemplo.** Dado el circuito de la figura 2.25 y utilizando lógica por defecto en PROLOG:

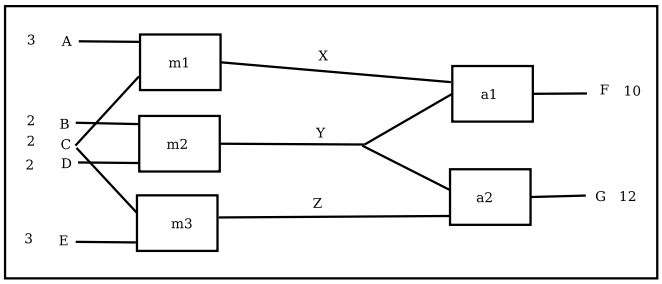


**Figura 2.25:** Circuito multiplicador-sumador

La cláusula circuito siguiente define la estructura del circuito:

```
circuito([A, B, C, D, E], [F, G]) :- multiplicador(m1, [A, C], X),
                                     multiplicador(m2, [B, D], Y),
                                     multiplicador(m3, [C, E], Z),
                                     sumador(a1, [X, Y], F),
                                     sumador(a2, [Y, Z], G).
```

Las cláusulas `multiplicador` y `sumador` definen el comportamiento de los elementos:

```
multiplicador(C, [A, B], X) :- not(defectuoso(C)), times(A, B, X).
multiplicador(C, [A, B], X):- defectuoso(C).
sumador(C, [A, B], X):- not(defectuoso(C)), plus(A, B, X).
sumador(C, [A, B], X):- defectuoso(C).
```

Si se ejecuta la cláusula circuito, se obtiene el resultado esperado ya que, bajo la hipótesis de mundo cerrado de Prolog, por defecto se asume que todos los elementos funcionan correctamente:

```
?- circuito([3, 2, 2, 3, 3], [F, G]).
F = 12, G = 12
```

No obstante, si modificamos la base de hechos:

```
?- assert(defectuoso(a1)).
```

se puede comprobar que las salidas observadas son consistentes con la hipótesis de que a1 es defectuoso:

```
?- circuito([3, 2, 2, 3, 3], [10, 12]).
yes
```

De este modo, podemos establecer la base para la cláusula `diagnostico` realizando sistemáticamente hipótesis de conjuntos de elementos que pueden estar defectuosos y comprobando cuáles son consistentes con las observaciones:

```
?- diagnostico(circuito, [3, 2, 2, 3, 3], [10, 12], X).
X = [[a1], [m1], [m2, a2], [m2, m3]]
```

Como ejemplo de la aplicación de las técnicas de DD, Self parte de la codificación en Prolog de un problema de resta de números de tres cifras: `ABC-DEF = GHI`.

El circuito de resolución del problema de la resta se muestra en la figura 2.26.

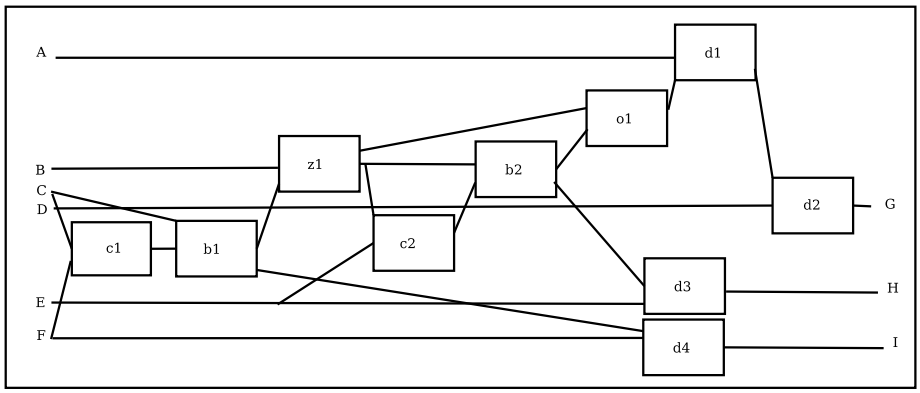


**Figura 2.26:** Circuito de resta de 3 columnas (c1, c2 son comparadores, b1 y b2 son propagadores de acarreo, z1 es un propagador de cero, o1 es una puerta OR y d1, d2, d3 y d4 son restadores)

La descripción del sistema podría ser la siguiente:

```
sub3([A, B, C, D, E, F], [G, H, I]) :-
    comparer(c1, [C, F], X1), borrower(b1, [X1,C], [X2, C1]),
    differencer(d4, [C1, F], I), zeroer(z1, [X2, B], [X3, B1]),
    comparer(c2, [B1, E], X4), borrower(b2, [X4,B1], [X5, B2]),
    differencer(d3, [B2, E], H), or(o1, [X3,X5], X6),
    differencer(d1, [A, X6], A1),differencer(d2, [A1,D], G).
```

```
comparer(C, [A, B], X):- not(faulty(C)), gre(A, B, X).
comparer(C, [A, B], X):- faulty(C).
borrower(C, [1, A], [0, A]):- not(faulty(C)).
borrower(C, [0, A], [1, X]):- not(faulty(C)), plus(10, A, X).
borrower(C, [N, A], [X, Y]):- faulty(C).
differencer(C, [A, B], X):- not(faulty(C)), minus(A, B, X).
differencer(C, [A, B], X):- faulty(C).
zeroer(C, [1, 0], [1, 9]):- not(faulty(C)).
zeroer(C, [1, A], [0, X]):- not(faulty(C)), gr(A, 0, 1), minus(A, 1, X).
zeroer(C, [0, A], [0, A]):- not(faulty(C)).
zeroer(C, [N, A], [X, Y]):- faulty(C).
```

La cláusula `sub3` define la estructura del circuito y las cláusulas `comparer`, `borrower` (propagador de acarreo), etc., definen el comportamiento de los componentes (por ejemplo, un comparador no defectuoso pone su salida a 1 si su entrada primera es mayor o igual que la segunda entrada y un 0 en caso contrario, y la salida de un comparador defectuoso no está restringida).

Si se ejecuta la cláusula `sub3`, se obtiene el resultado esperado asumiendo por defecto que todos los elementos funcionan correctamente bajo la hipótesis de mundo cerrado de Prolog:

```
? sub3([2, 7, 4, 1, 2, 9], [G, H, I]).
G = 1, H = 4, I = 5
```

Si modificamos la base de hechos:

```
?- assert(defectuoso(b1)).
```

se confirma que la salida observada, `155`, es consistente con la hipótesis de que `b1` es defectuoso:

```
? sub3([2, 7, 4, 1, 2, 9], [1, 5, 5]).
yes
```

De este modo, se pueden localizar los candidatos defectuosos en el diagnóstico como vimos en el ejemplo anterior. Por ejemplo, si un estudiante responde `155` al problema de `274-129`, hay cinco candidatos en el diagnóstico:

```
?- diagnóstico(sub3,[2, 7, 4, 1, 2, 9], [1, 5, 5], X).
X = [[b1], [z1], [b2], [d3], [c1, d4]]
```

Cada uno de los candidatos es minimal, es decir, es suficiente para explicar la observación y, por lo tanto, no significa que otros componentes estén necesariamente no defectuosos.

#### 2.3.3.1 Extensiones requeridas para DC

#### 2.3.3.2 *Modelos Jerárquicos*

Una de las dificultades que aparecen al describir circuitos (como el de la figura 2.26) consiste en decidir el nivel de detalle (la finura del grano del conocimiento) de la descripción del mismo. En el caso del DD, hay una hipótesis implícita acerca del nivel en el que se llevará a cabo la reparación ante un posible fallo. Se asume que se reparará o sustituirá un multiplicador o un sumador completo, no un simple transistor. Sin embargo, en el caso del DC, no se puede asumir

a menudo esta hipótesis. Por ejemplo, no podemos suponer que un estudiante tendrá dificultades, por ejemplo, con la propagación de acarreos, pero no con algún componente específico de la propagación de acarreos, como los acarreos cuando el minuendo es menor que el sustraendo. Por lo tanto, para el caso del DC, se podría, en lugar de adivinar el tamaño del grano de detalle adecuado, empezar con una descripción abstracta y, progresivamente, expandir componentes sospechosos.

Por ejemplo, para la resta de tres columnas se podría comenzar con el siguiente circuito (Figura 2.27):

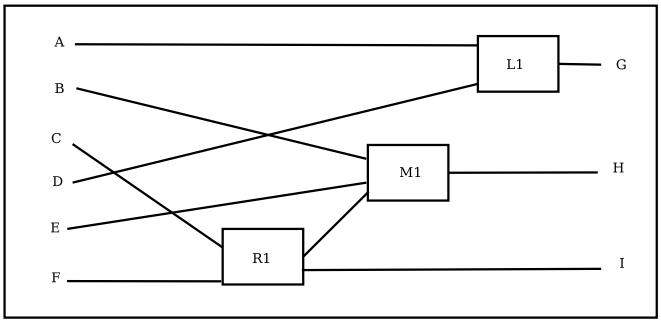


**Figura 2.27:** Circuito de resta de 3 columnas abstracto

La descripción del sistema podría ser la siguiente, donde la estructura del sistema es más sencilla que la anterior pero el comportamiento del componente es más complejo:

```
sub3([A, B, C, D, E, F]), [G, H, I]):-
    righter(r1, [C, F], [X1, I]), middler(m1, [X1, B, E], [X2, H]),
    lefter(l1, [X2, A, D], G).

righter(C, [A, B], [X, Y]):-
    not(faulty(C)),
    gre(A, B, 1), X=0, minus(A, B, Y);
    gr(B, A, 1), X=1, plus(10, A, Z), minus(Z, B, Y)).

righter(C, [A, B], [X, Y]):- faulty(C).
```

Y de forma similar para las cláusulas de `middler` y `lefter` (Self (1992)). Un diagnóstico para esta versión abstracta de `sub3` sería de la forma siguiente:

```
?-diagnóstico(sub3,[8, 0, 3, 2, 6, 7], [6, 2, 6], X). X = [[m1]]
```

Es decir, un fallo en el elemento de la columna central es suficiente para explicar la observación.

Partiendo de esta descripción, se puede crear una nueva descripción menos abstracta sustituyendo el componente `m1` por su expansión. Por ejemplo, se pueden definir cláusulas de expansión como ésta:

```
expansion:-
    middler(m1, [X1, B, E], [X2, H]), zeroer(z1, [X1, B], [X3, B1]),
    comparer(c2, [B1, E], X4), borrower(b2, [X4, B1], [X5, B2]),
    differencer(d3, [B2, E], H), or(o1, [X3, X5], X2).
```

de modo que el componente `m1` se reemplaza dinámicamente en la cláusula `sub3` por los elementos que lo forman. Con estas modificaciones queda una descripción del sistema con siete componentes (`righter`, `lefter` y los cinco componentes que reemplazan a `midler`) y el diagnóstico queda de la siguiente forma:

```
?-diagnóstico(sub3, [8, 0, 3, 2, 6, 7], [6, 2, 6], X).
X = [[z1], [b2], [d3, l1]]
```

en el que podemos tener candidatos de distintos niveles de detalle, como en el último elemento de la lista.

#### 2.3.3.3 *Modelos de fallo*

Hasta el momento se ha considerado en las descripciones de un sistema que el elemento defectuoso no tiene restricciones que afecten a su salida. Sin embargo, en DC se sabe que ciertos componentes fallan de una manera específica. Por ejemplo, hay una serie de errores estándar que afectan al tratamiento de los acarreos en la resta. Especificando estos modelos de fallos conseguimos reducir el número de candidatos del diagnóstico. En DC el estudio de modelos de fallo ha sido un área de investigación muy rica. Así, se pueden representar en Prolog los diez fallos más frecuentes en las restas (Self (1993)). Por ejemplo, la cláusula `borrower` quedaría de la forma siguiente:

```
borrower(C, [1, A], [0, A]):- mood(C, ok),
borrower(C, [0, A], [1, X]):- mood(C, ok), plus(10, A, X).
borrower(C, [0, A], [0, X]):- mood(C, borrow_no_decrement), plus(10, A, X).
borrower(C, [0, 1], [1, 10]):- mood(C, borrow_into_one).
```

Con esta descripción, cada componente tiene ahora un modo *ok* y una serie de modos de fallo. Por ejemplo, el fallo `borrow_no_decrement` pone su primera salida a 0 (cuando debería ser 1) y provoca que no se decremente la siguiente columna.

El procedimiento de diagnóstico se debería modificar para que trabajase sistemáticamente con los distintos modos de fallo y encontrase una asignación de éstos consistente con la observación:

```
?-diagnóstico(sub3, [6, 0, 5, 4, 5, 0], [2, 5, 5], X).
X = [[[b2, borrow_no_decrement]],
    [[c2, does_not_compare], [d3, smaller_from_larger]]
    [[c2, does_not_compare], [d3, diff_0_minus_N_is_N]]]
```

es decir, dados los diez modelos de fallo, se tienen tres candidatos para la observación `605-450=255`. El primero, por ejemplo, presenta el propagador de acarreo `b2` con el modo de fallo `borrow_no_decrement` (los otros componentes, por defecto, están en modo *ok*).

#### 2.3.3.4 *DC a partir de varias observaciones*

Hasta el momento se ha asumido que el diagnóstico se lleva a cabo a partir de una única observación. No obstante, en general, el diagnóstico está basado en una serie de observaciones, y en el DC se deben considerar estas observaciones como parte de una interacción en constante evolución en la que, en ciertos pasos, disponemos de un diagnóstico provisional. Por lo tanto,

es importante modificar el procedimiento de diagnóstico general para que pueda manejar estas observaciones.

Suponiendo que disponemos del diagnóstico expuesto anteriormente y de éste nuevo:

```
?-diagnóstico(sub3, [6, 0, 5, 4, 5, 0], [2, 5, 5], X).
X = [[[c2, does_not_compare], [d3, smaller_from_larger]]
    [[c2, does_not_compare], [d3, diff_0_minus_N_is_N]]]
```

está claro que el primer candidato del diagnóstico anterior ( `[[b2, borrow_-no_decrement]]` ) se ha eliminado por la observación segunda. Para combinar los dos diagnósticos podríamos hacer la intersección de los candidatos o, más exactamente, deberíamos tomar la menos general de las especializaciones de candidatos, ya que cada candidato es una hipótesis mínima.

De este modo, teóricamente, el diagnóstico a partir de múltiples observaciones no presenta problemas, pero, en la práctica, es muy ineficiente realizar cada diagnóstico por separado para después combinarlos. En lugar de esto, podríamos reunir el grupo de candidatos del primer diagnóstico y utilizarlo como restricciones para el espacio de búsqueda del segundo diagnóstico. Así, el procedimiento de diagnóstico requiere un argumento más que se utilizará para limitar el espacio de búsqueda. Así, sí es factible que el DC se pudiera realizar dinámicamente a lo largo de una interacción.

#### 2.3.3.5 *Fallos sistemáticos*

En el apartado anterior se asume que el estudiante es consistente en su comportamiento de una observación a la siguiente, pero, en la práctica, es bastante incierto. Consideremos un diagnóstico de una sola observación:

```
D1 = [[[b2, borrow_no_decrement]]]
```

En DD podríamos esperar que el fallo de un elemento sea más o menos independiente del fallo de otros componentes. Pero, para el DC, podemos esperar que todos los elementos del mismo tipo se encuentren en el mismo modo. Por ejemplo, si un acarreo está en un determinado modo de fallo, entonces todos los componentes de acarreo estarían en el mismo modo de fallo. Así, aunque `D1` indica que sólo `b2` produce fallo, podríamos extender el diagnóstico para incluir todos los acarreos, aunque no haya una observación directa de que están fallando:

```
D1 = [[[b2, borrow_no_decrement], [b1, borrow_no_decrement]]]
```

De manera similar, el diagnóstico:

```
D2 = [[[b2, borrow_no_decrement], [b1, borrow_into_one]]]
```

debe rechazarse, ya que no debe permitirse que dos componentes del mismo tipo estén en dos modos de fallo diferentes. El incluir o no estos refinamientos en el procedimiento de diagnóstico no es un problema teórico sino una cuestión que debe decidir el diseñador del DC.

#### 2.3.3.6 *Descuidos*

Otro problema que aparece en el DC pero menos con el DD (aunque se podría asemejar al problema de los fallos intermitentes de un dispositivo) es el de controlar los errores que puede cometer el estudiante por descuidos. Un descuido es el resultado de un fallo de ejecución más que en la intención. Por ejemplo, el estudiante puede realizar la operación `271-138` con éxito pero a la hora de transcribir el resultado equivocarse y escribir `132` en lugar de `133`. Evidentemente, la existencia de estos descuidos supone una dificultad considerable para el DC, ya que no queremos embarcarnos en un diagnóstico complicado y costoso cuando la explicación de algún error es un simple descuido del estudiante.

Una posibilidad es considerar el circuito de resta mostrado en la 2.27, excepto que antes de cada salida haya un componente extra (un *generador de descuidos*) que puede corromper la salida. De este modo, la descripción del sistema es:

```
sub3([A, B, C, D, E, F]), [X, Y, Z]):-
    slipper(s1, G, X), slipper(s2, H, Y), slipper(s3, I, Z).
    slipper(C, A, A):- mood(C, ok).
    slipper(C, A, X):- mood(C, slip).
```

Y por lo tanto, podríamos obtener:

```
?- diagnóstico(sub3, [2, 7, 1, 1, 3, 8], [1, 3, 2], X).
X = [ [[s3, slip]], [[b1, borrow_into_one]] ]
```

#### 2.3.3.7 *Generar el siguiente problema*

Ya que se ve el diagnóstico como parte de una interacción en desarrollo, se puede considerar el usar nuestros modelos para generar el siguiente problema adecuado, un importante paso en sistemas de tutoría (y equivalente a determinar qué entradas se tienen que proporcionar para conseguir la siguiente observación en el DD). Por ejemplo, dados tres candidatos podríamos determinar un buen problema para limitar a continuación los candidatos. De Kleer y Williams (De Kleer and Williams (1989)) sugieren utilizar el problema con mínima entropía, es decir, aquél que proporcione la máxima información para discriminar entre los candidatos. Sin embargo, esto no nos dice directamente qué problemas considerar. Para resolver este problema, se puede desarrollar un método heurístico (en el sentido de que ordene el espacio de búsqueda pero no elimine posibles soluciones). Se puede ver un ejemplo en Self (1993).

#### 2.3.3.8 Problemas por resolver en DC

Con este estudio, Self (Self (1993)) llegó a la conclusión de que la mayoría del trabajo desarrollado sobre DC en el campo de los sistemas de tutoría se puede expresar en los términos utilizados en el marco teórico del DD, con la consecuente mejora en claridad y rigor. Sin embargo, quedan por resolver algunos problemas importantes como:

- *Diagnóstico interactivo*: El DC es interactivo pero el DD no, es decir, en el DC el estudiante es un agente activo en el proceso de diagnóstico. El hecho de que el objeto de diagnóstico (el estudiante) pueda participar por sí mismo en el diagnóstico y autocorregirse complica altamente el tratamiento formal del DC. Al menos, esto indica que cualquier teoría de DC debe integrarse con las teorías de aprendizaje e interacción cuando existan.

- *El problema del diseño del circuito*: El DC no dispone de un diseño de un circuito desde el que comenzar, como ocurre en el DD. No podemos asumir, como diagnosticadores, qué circuitos describen el conocimiento del estudiante (Clancey). Para solucionar este problema es conveniente manejar todas las formas de representación del conocimiento proporcionadas por la Inteligencia Artificial.

- *El propósito del DC*: El propósito del DD no es resolver un problema - reparar o sustituir componentes defectuosos. El DC puede usarse para soportar la corrección, pero hemos visto que, más generalmente, se usa para ayudar a dirigir alguna interacción en la enseñanza. Ohlsson (Ohlsson (1986)) considera que el propósito del DC es comprobar si los objetivos de aprendizaje planificados por el tutor son consistentes con el modelo del estudiante. La salida del DC debería ser una función de las opciones de enseñanza disponibles en ese momento. Sin embargo, el DD va "hacia atrás" con respecto a los propósitos del DC: no deberíamos obtener un diagnóstico completo y luego decidir qué hacer con él sino, más bien, considerar qué opciones tenemos y luego obtener el mínimo diagnóstico suficiente para elegir entre esas opciones. En el DC, a diferencia del DD, debido a que el propósito del mismo puede cambiar, el proceso de diagnóstico necesita ser explícito para que pueda ser razonado con respecto a los objetivos.

- *El contenido del diagnóstico*: La comparación con el DD desafortunadamente refuerza el punto de vista de que el DC debería centrarse en las deficiencias de conocimiento. Las ideas actuales sobre el diseño de sistemas de tutoría se mueven, más allá de los sistemas que corrigen a estudiantes ignorantes, hacia entornos que sirven de apoyo a estudiantes en la dirección de sus actividades de aprendizaje. Para estos entornos más abiertos, el diagnóstico debería estar más relacionado con las metas del estudiante y las capacidades metacognitivas (es decir, con el control y reflexión sobre el proceso de resolución del problema), aspectos que no necesitan considerarse en el DD.

### 2.3.4 Extensión del Motor de Diagnóstico General para DC

El siguiente método de diagnóstico presentado por De Koning et al. (de Koning et al. (1995), de Koning and Bredeweg (1998), de Koning et al. (2000)), es una versión adaptada del *Motor de Diagnóstico General* -MDG (de Kleer and Williams (1987), Reiter (1987)) visto en el apartado 2.3.2. El DBM determina los fallos mediante la comparación del comportamiento observado de un dispositivo y el comportamiento predicho por un modelo. La idea fundamental subyacente en el paradigma (ya vista) puede resumirse así: el usuario proporciona una descripción estructural del sistema a diagnosticar, algunos valores observados (entradas y salidas del sistema) y un conjunto de posibles modelos de comportamiento para los componentes que constituyen el sistema. El diagnóstico comienza reuniendo los modelos para cada uno de los componentes en la descripción estructural. El modelo de comportamiento completo que resulta es la entrada para la predicción de valores, es decir, el modelo se usa para inferir: a) qué valores de salida deberían haberse generado para ciertos valores de entrada y, b) qué valores de entrada se necesitan para generar ciertos valores de salida. Todos los valores obtenidos de esta forma son los valores predichos o esperados para el comportamiento del dispositivo. Los valores esperados se comparan luego con los valores que pueden obtenerse del sistema real. Las diferencias entre estos valores se llaman discrepancias. Las discrepancias se usan para localizar el conjunto de componentes que se cree que son causa de dichas discrepancias. Cada candidato explica todas las discrepancias y refleja un diagnóstico potencial del mal funcionamiento del dispositivo. Se

pueden formular criterios para decidir si un cierto candidato puede verse como un diagnóstico suficiente.

Frecuentemente sucede que ninguno de los candidatos satisface completamente los criterios de ser un diagnóstico suficiente. Se requieren observaciones adicionales que midan el valor de algún parámetro para reducir el conjunto de posibles candidatos. Estas medidas se denominan pruebas. A partir de las publicaciones iniciales del paradigma del MDG, se han implementado un conjunto de programas que verifican y/o extienden de alguna manera la aproximación original en el marco del diagnóstico en IA (por ejemplo, Struss and Dressler (1989)).

A continuación, se describe en primer lugar el fundamento teórico de este método de DC y finalmente, se muestra la arquitectura y los pasos concretos de la aproximación.

#### 2.3.4.1 Conocimiento en múltiples niveles

Self, como ya hemos descrito en el apartado previo, mostró con éxito cómo los intentos de diagnosticar al estudiante en la resolución de problemas de resta podían ser representados en el marco del DBM pero concluyó que "la mayoría de los estudios realizados sobre DC pueden expresarse de nuevo en el marco del MDG con incremento de claridad y rigor". Esta hipótesis necesita más investigaciones y validaciones para lograr representar los estudios de DC en el paradigma del DBM. Además, Self señaló cuatro problemas fundamentales pendientes que deben resolverse para que el DBM cubra completamente al DC. Uno de ellos (el tercero) es que el propósito del DC no está siempre claro, a diferencia del DBM, donde se requiere la identificación del componente defectuoso para arreglarlo o sustituirlo. El DC debería tener en cuenta explícitamente el plan en el que el diagnóstico está inmerso (las expectativas del plan y el ámbito de cambio). Según Self, se debería definir un procedimiento de meta-diagnóstico explícito que razonase sobre (los errores en) expectativas, objetivos, etc. dado que, los errores o puntos muertos en la ejecución de un estudiante pueden ser debidos no sólo a defectos o confusiones en su conocimiento de dominio, sino también en su conocimiento de razonamiento.

Es importante recordar que una característica fundamental de este nueva propuesta es que usa el marco KADS de modelado de resolución de problemas de Wielinga y otros ya descrito en el apartado de taxonomías en el conocimiento del ME (sección 2.2.2). Este marco mutiestratificado permite a los autores distinguir varios tipos de conocimiento (conocimiento de dominio, conocimiento de inferencia, conocimiento de tarea -metas, y estructura específica de los pasos de inferencia y, en especial, conocimiento estratégico -estrategia de resolución del problema) proporcionando un soporte para manejar las nociones de metaconocimiento y metadiagnóstico. Dentro del paradigma del MDG, De Koning y Bredeweg reformulan cada entidad de conocimiento del marco KADS en términos de un componente, con sus entradas y salidas específicas, y una restricción que especifique cómo puede obtenerse la salida a partir de la entrada. De este modo, se elimina el "tercer problema pendiente" según Self.

#### 2.3.4.2 Enfoque del diagnóstico

De Koning y Bredeweg definen el problema del diagnóstico cognitivo a resolver de la siguiente forma:

- *Dado el modelo de la tarea de resolución del problema en términos de pasos de razonamiento individual y conexiones de datos entre estos pasos de razonamiento, más un conjunto de observaciones sobre el comportamiento del estudiante en la resolución del problema, el diagnóstico se define como los conjuntos minimales de pasos de razonamiento que pueden*

*no haber sido aplicados correctamente por el estudiante dadas las observaciones (de Koning and Bredeweg (1998)).*

Esta definición difiere de la comúnmente aceptada (Ohlsson (1986)). Los autores no intentan determinar el estado cognitivo interno del estudiante sino sólo diagnosticar el comportamiento del razonamiento externo del estudiante. El diagnóstico, por lo tanto, consiste en errores en el proceso de diagnóstico del razonamiento del estudiante más que en conceptos erróneos en el conocimiento del estudiante. El diagnóstico se define así a nivel de comportamiento y no a nivel conceptual.

El paradigma del MDG tiene en cuenta jerarquías de modelos de componentes. Esta característica es fundamental para, en el modelo de experiencia, poder representar las interdependencias entre los componentes de conocimiento de las capas de dicho modelo. Diagnosticar el comportamiento del estudiante es ahora más complejo que reunir automáticamente los diferentes modelos de comportamiento en la descripción estructural, ya que se necesita control explícito. En el paradigma del MDG esto se denomina *foco* del proceso de diagnóstico. En el caso del marco KADS descrito anteriormente, este foco se fija mediante las metas de resolución del problema que deben ser alcanzadas y por la estructura de tarea específica que se usa para este propósito. La estructura de tarea proporciona el foco para determinar las Fuentes de Conocimiento (FC) o pasos de inferencia que deben ser objeto del razonamiento de diagnóstico. Estas inferencias proporcionan un segundo foco y mediante sus metaclases se refieren a un subconjunto específico de todo el conocimiento de dominio disponible. Por lo tanto, la clase de inferencia que está realizándose en un cierto punto en el proceso de razonamiento, limita el número de componentes en la capa de dominio que puede tener que considerarse.

#### 2.3.4.3 Definición del modelo de la tarea de resolución del problema

Para resolver el problema del diagnóstico tal y como se define en esta aproximación, el primer paso consiste en definir un modelo de la tarea de resolución del problema. Este modelo debería considerar los requerimientos de representación del razonamiento basado en modelos, es decir, debería consistir en componentes independientes del contexto y conexiones entre éstos. Ya que la ejecución de la tarea de resolución de un problema puede verse como la realización de un conjunto de operaciones o pasos de inferencia sobre un conjunto de datos, en esta aproximación cada componente representa un paso de razonamiento instanciado en el proceso de resolución del problema particular (de Koning et al. (2000)). Estos componentes se usan para definir el modelo de diagnóstico como una traza del razonamiento con los pasos de razonamiento individuales que deben ser dominados por el estudiante para obtener la solución correcta. De acuerdo con esta definición del modelo, un diagnóstico es un conjunto de pasos de razonamiento que el estudiante puede no haber aplicado correctamente dadas las observaciones.

Aunque las ventajas de usar diagnóstico basado en modelos en la educación son considerables en términos de generalidad, reusabilidad, y transferabilidad, un obstáculo importante es el hecho, como ya se vio anteriormente, de que los modelos de comportamiento en la resolución correcta de un problema no están disponibles inmediatamente (Self (1993)). Para resolver este problema se propone en esta aproximación generar los modelos de dominio de diagnóstico automáticamente mediante un simulador cualitativo pero, como veremos a continuación, esto requiere que la salida del simulador pueda ser transformada para alcanzar los fuertes requerimientos del diagnóstico basado en modelos.

De acuerdo a lo anterior, en esta aproximación, la definición del conocimiento de resolución de un problema en términos de modelos de conexión de componentes o modelo de dispositivo

se denomina *modelo base*. Para una predicción de comportamiento específica, el modelo base representa el conjunto de todos los pasos de razonamiento o inferencias que se requieren para esta predicción. Aunque los tipos de componentes son genéricos para la tarea, cada predicción para un sistema específico requiere generar un nuevo modelo. Esto se realiza sobre las bases de la salida del simulador cualitativo denominado GARP (Bredeweg and Breuker (1993)). Este simulador genera una predicción del comportamiento sobre las bases de la descripción de algún sistema y un post-procesador transforma esta salida en un modelo que representa todos los pasos de razonamiento individuales que un estudiante debería dominar para resolver el problema de predicción.

En el modelo base, un componente (inferencia) puede definirse mediante una entrada, una salida y algún conocimiento de soporte (genérico) para derivar la salida desde la entrada. Cada componente tiene un conjunto no vacío de puertos de entrada, un conjunto, que puede ser vacío, de puertos de conocimiento soporte genérico y un puerto de salida. Cada puerto del componente está conectado a exactamente un punto de medida, pero un punto de medida puede estar conectado a más de un puerto de componente. Si un punto de medida está sólo conectado a puertos de entrada (o sólo de salida), el punto es un modelo de entrada (o salida). El flujo de datos a través de estas conexiones está formado por expresiones instanciadas. Un fragmento de modelo base se muestra en de Koning and Bredeweg (1998) y de Koning et al. (2000).

El modelo base, al contener todos los pasos de razonamiento que son necesarios para una correcta predicción de comportamiento, tiende a ser bastante grande. Como resultado, aplicar el MDG directamente sobre el modelo base no es factible en un entorno educativo en tiempo real. Aunque en teoría el número no es prohibitivo para diagnóstico en tiempo real, el modelo base tiene un número de características específicas que son desventajosas con respecto al diagnóstico basado en modelo. En primer lugar, la conectividad entre componentes es relativamente baja, comparada con circuitos digitales. En segundo lugar, el número de observaciones, y particularmente salidas observadas, es bajo y, finalmente, debido a que los cálculos del razonamiento cualitativo subyacente son relativamente débiles, la definición de reglas de comportamiento para la propagación hacia atrás no es posible para todos los tipos de componente. La combinación de estas características junto con el gran número de componentes proporciona un escenario en cualquier caso peor para diagnóstico basado en modelos.

Para solventar los problemas anteriores, se aplican técnicas de modelización jerárquica. La diferencia fundamental entre los modelos de conocimiento y los modelos de dispositivos físicos es que, para los primeros, la estructura jerárquica no esta disponible inmediatamente, es decir, los componentes de inferencia que forman parte del modelo base no se proyectan necesariamente en componentes en el sistema real, sino que tienen que ser generados para cada modelo base en tiempo de ejecución. Para ello, se ha desarrollado también en esta aproximación algoritmos que automáticamente añaden estructura jerárquica al modelo base. De este modo, se aplican a continuación tres tipos de abstracciones diferentes al modelo: en primer lugar, el modelo se simplifica ocultando todos los pasos de razonamiento (componentes) que no son esenciales (aunque son necesarios técnicamente hablando) para el comportamiento básico del sistema. En segundo lugar, se sustituyen las secuencias (fragmentos) de pasos de razonamiento por un componente de inferencia abstracto combinando una cadena de pasos de razonamiento en uno. Finalmente, se realiza el tercer nivel de mayor abstracción, en el que se agrupan los diferentes componentes de inferencia de acuerdo al estado de comportamiento o transición a la que pertenecen.

Por lo tanto, el proceso de generar un modelo jerárquico de la tarea de resolución de problemas a aprender por el estudiante o *tarea de predicción cualitativa de comportamiento* está

completamente automatizado y los algoritmos correspondientes que implementan los pasos anteriores se detallan en de Koning et al. (2000).

t

#### 2.3.4.4 Arquitectura para el DC

Los modelos jerárquicos así añadidos a las restricciones de representación del DBM permiten que se pueda aplicar la técnica de MDG para valorar el comportamiento del estudiante en la resolución de problemas.

El método de DC de K. de Koning et al. (de Koning and Bredeweg (1998), de Koning et al. (2000)) derivado del MDG (de Kleer and Williams (1987)) usa una arquitectura general para sistemas de enseñanza basada en la arquitectura para sistemas inteligentes de ayuda propuesta por Breuker (Breuker (1990)). En la figura 2.29 se muestra un esquema de la parte relevante para DC de esta arquitectura. Como se puede observar, el motor de diagnóstico usa información del *Modelo de Dominio Instanciado* (modelo de dispositivo) generado, como ya se ha explicado anteriormente, por el *Simulador Cualitativo GARP* y las *Observaciones* obtenidas por un *Intérprete de Funcionamiento*, que se encarga de analizar la entrada del estudiante y chequear si las respuestas son correctas. Las *Observaciones* se utilizan para el *Reconocimiento de Conflicto*, igual que en el MDG. En el MDG estándar, la *Discriminación de Candidatos* se hace estimando el número de tests (pruebas) necesarios para establecer un diagnóstico. En el contexto de un sistema de enseñanza, sin embargo, el conocimiento sobre el dominio y la información del ME pueden estar disponibles para guiar el proceso de discriminación según se verá a continuación. El resultado del proceso de diagnóstico completo se pasa al tutor.

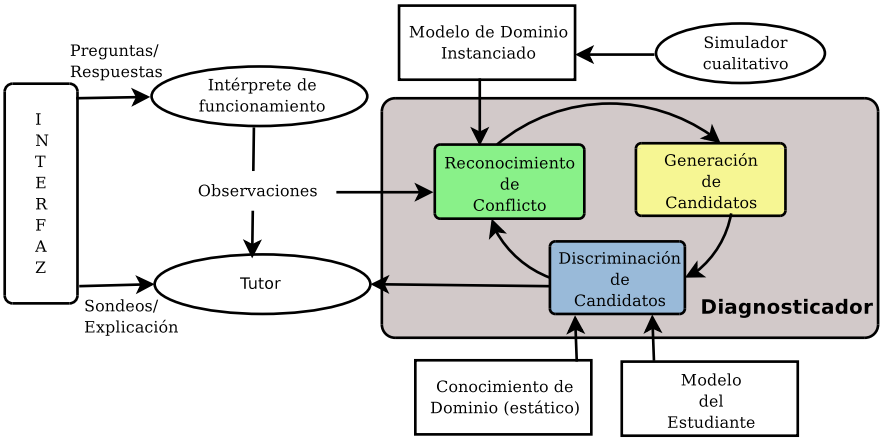


**Figura 2.28:** Arquitectura parcial para Sistemas de Enseñanza

El método de diagnóstico es el siguiente:

- ***Reconocimiento de Conflicto***: Un conflicto es un conjunto de componentes en el que, al menos, uno de ellos es incorrecto. Este reconocimiento se hace siguiendo, sin diferencias significativas, el proceso de diagnóstico del MDG y consiste básicamente en comparar las observaciones proporcionadas por el *Intérprete de Funcionamiento* con el comportamiento esperado generado por el *Simulador Cualitativo*, obteniéndose así el conjunto de conflictos.

- ***Generación de candidato***: Un candidato es un conjunto de componentes tal que, cuando todos los elementos de éste fallan, produce una predicción consistente con las observaciones y constituye un diagnóstico. Este paso, al igual que el anterior, se hace siguiendo, sin diferencias significativas, el proceso de diagnóstico del MDG. Los conjuntos de candidatos que cubren cada conflicto se generan usando recubrimiento incremental, es decir, cada candidato que no cubra ninguno de los nuevos conflictos, se sustituye por un nuevo conjunto de candidatos, cada uno de los cuales contiene el candidato antiguo más un elemento del nuevo conflicto. A continuación, se minimaliza el conjunto resultante eliminando cada candidato subsumido o duplicado por otro candidato.

- ***Discriminación del candidato***: Este tercer paso ya difiere del proceso de diagnóstico del MDG estándar. Esto es debido a que la naturaleza de los componentes es significativamente distinta en los modelos de conocimiento y en los circuitos digitales. En un circuito digital, dos componentes del mismo tipo pueden comportarse de acuerdo a las mismas reglas, pero son instancias distintas físicamente. En los modelos de razonamiento, esto no es necesariamente así, diferentes componentes (pasos de razonamiento) del mismo tipo probablemente fallen colectivamente. La única excepción son los componentes de tipo recuperación. Este componente se introduce en esta aproximación y tiene una entrada y una salida. Esta salida es igual a la entrada si el componente está funcionando correctamente y, si es defectuoso, representa la situación en que el estudiante no conoce, no puede recuperar o no puede reproducir una determinada expresión. En este caso, diferentes instanciaciones del componente son realmente operaciones independientes, ya que se refieren a la recuperación de hechos de conocimiento diferentes (por ejemplo, el error de no recuperar correctamente la relación entre *nivel* y *presión* no está relacionado normalmente con una recuperación incorrecta de la influencia negativa de la velocidad de flujo sobre el volumen). Partiendo de estas premisas, el algoritmo de este paso del diagnóstico considera que la probabilidad de fallo de un conjunto de instancias del mismo tipo de componente es igual que la probabilidad de un componente sencillo en la discriminación del candidato. Para componentes de tipo recuperación, se puede emplear una heurística adicional en la discriminación de candidato. Teniendo en cuenta que la mayoría de los errores que se cometen son debidos a conocimiento de dominio perdido o confundido, puede asumirse que los componentes de recuperación tienen una más alta probabilidad de fallo a priori que los componentes de inferencia y, de forma similar, los componentes que se pueden descomponer tienen una probabilidad de fallo más alta que los componentes del modelo base individual, porque incorporan un número de pasos de razonamiento en el modelo base. Se puede observar que el motor de diagnóstico explota aquí sólo la estructura del modelo y los tipos de componentes, y no hace referencia a las semánticas de las inferencias o predicción de comportamiento específico que se ha modelado.

  Los tres algoritmos utilizados se detallan en de Koning et al. (2000).

Como aplicación de este enfoque, se ha implementado en un sistema prototipo llamado START$^{light}$ cuya apariencia puede verse en de Koning and Bredeweg (1998) y de Koning et al. (2000) así como un ejemplo detallado de funcionamiento del prototipo. Este prototipo implementa completamente el modelo de generación y el modelo jerárquico así como el motor de diagnóstico descritos anteriormente. El ejemplo de diagnóstico se basa en una pregunta típica sobre el comportamiento de un sistema físico llamado *sistema de balanzas*: dos contenedores llenos de agua se colocan en una balanza, y cada contenedor tiene una salida en la parte inferior. El peso de los contenedores vacíos se supone que es igual y su tamaño es distinto. La tarea

del estudiante en el ejemplo es predecir qué sucede a las dos columnas de agua y a la balanza una vez que se abren las salidas de ambas.

#### 2.3.4.5 Ventajas e inconvenientes del método

Con esta aproximación se ha hecho posible que los métodos de DBM, y en particular el MDG, pueda reutilizarse para DC aplicándolo a los dos niveles de resolución de problemas: nivel de dominio y (meta-)nivel de razonamiento. Al menos esta adaptación es posible para dominios de enseñanza y aprendizaje que permiten la modelización estructural/de comportamiento, tales como (razonamiento sobre) dispositivos físicos. En concreto, se han empleado este método usando simulación cualitativa y diagnóstico basado en modelos para representar y analizar el comportamiento del razonamiento de un estudiante en la predicción del comportamiento de un dispositivo.

La principal ventaja de esta aproximación es que proporciona un método genérico para la tarea de diagnóstico cognitivo que hasta el momento había estado basada en catálogos de errores predefinidos y específicos del dominio.

Además, estas investigaciones sobre la estructura del razonamiento en tareas de "predicción de comportamiento" ha proporcionado un "circuito de diseño" detallado y validado para este tipo de tareas. Sin embargo, no quiere decir que se haya resuelto el segundo "problema pendiente" de Self - no se da el "circuito" a diagnosticarse en DC. De hecho, se muestra que requiere gran adquisición de conocimiento y experimentos de validación.

Otro tema pendiente en esta aproximación es que sólo se considera la localización de deficiencias en el comportamiento del estudiante como causas de errores pero no las confusiones en el conocimiento del estudiante. La falta de conocimiento puede ser la más frecuente fuente de error en las técnicas de adquisición cognitivas, pero las confusiones (conceptos erróneos) son probablemente las más difíciles de detectar y remediar. Una confusión es análoga a una conexión incorrecta en un dispositivo (error de configuración). Los errores de configuración son una complicación extra en el diagnóstico, y no hay métodos de DBM que traten directamente con ellos. En este sentido, el marco STAR parece particularmente útil para el desarrollo de entornos educativos que estimulen las capacidades de autocorrección del estudiante.

Una característica fundamental del algoritmo de discriminación es que los costes computacionales son bajos porque no se supone propagación de comportamiento. Sin embargo, el diagnóstico final puede no siempre ser obtenido en un número mínimo de pruebas posibles, ya que el algoritmo no usa estimaciones sobre el número total de pruebas que se necesitan para establecer el diagnóstico final. Una razón importante para no usar estimaciones generales es que, los modelos jerárquicos aseguran que los modelos de diagnóstico son pequeños y, por lo tanto, el número de posibles puntos de prueba es generalmente bajo. De este modo, el impacto de la selección de un punto de prueba poco óptimo es limitado.

El algoritmo de discriminación de candidato no obtiene un punto de prueba, sino una lista de posibles puntos de prueba ordenados por su poder de discriminación (factor de división). Aunque el proceso de diagnóstico puede razonar acerca de los resultados esperados de un cierto punto de prueba, dicha maquinaria no puede determinar el coste de una prueba específica dentro del contexto educativo actual. Como consecuencia, la prueba más efectiva propuesta puede ser muy cara en el sentido de que no es adecuada en el diálogo actual. En este caso, el sistema educativo puede seleccionar otro punto de prueba de la lista.

### 2.3.5 Método de descomposición-P de Tsybenko

Los intentos de aplicar técnicas basadas en modelos al diagnóstico cognitivo de J. Self y K. de Koning et al., descritas anteriormente, muestran la utilidad del paradigma basado en modelos para DC. Sin embargo, las dificultades en construir y actualizar los MEs se mantienen. Estas aproximaciones suponen que la forma en que un estudiante aborda un problema se conoce a priori: Self construye un circuito de resolución del problema de la resta y Koning et al. aplican "modelos de dispositivos" para DC en tareas de "predicción de equilibrio en sistemas de balanzas". No obstante, a menudo no hay "modelos de dispositivos" que puedan utilizarse en DC. Un estudiante normalmente aplica su propio método de resolución del problema, y hay muchos dominios con múltiples formas posibles de abordar un problema. En estos casos, ¿cuál de los circuitos posibles debería usarse para DC cuando el estudiante falla al resolver este problema?. Yury Tsybenko (**?**) presenta una aproximación en la que el estudiante construye el circuito necesario (*modelo asociado*) durante la resolución del problema. Este circuito se usa además para el diagnóstico cognitivo.

Además, un análisis formal de los dispositivos que modelan al estudiante muestra que hay algunas características de tales dispositivos que hacen que la aplicación de las técnicas de MDG tradicional a su diagnóstico sea ineficiente. El paso de generación de un diagnóstico inicial proporciona, en general, muchos diagnósticos mínimos que compiten. Por este motivo, son necesarias medidas adicionales para seleccionar un diagnóstico. La cuestión es qué medida tomar a continuación. La aproximación de MDG aplica un análisis probabilístico para la medida de la siguiente selección. Sin embargo, la fiabilidad de los componentes del sistema cognitivo no se conoce a priori y, aunque se conocieran, son bajas comparadas con la fiabilidad de los componentes de un sistema técnico. Este hecho disminuye considerablemente el poder discriminatorio de la técnica MDG basada en la entropía para la selección de la siguiente medida. Es más, un dispositivo que modela a un estudiante no es tan complejo como un dispositivo técnico compuesto de cientos y cientos de componentes. Para los dispositivos que modelan al estudiante es posible aislar los componentes defectuosos directamente.

El método de descomposición-P de Tsybenko no cuenta con información probabilística sino que está guiado por información de primeros principios tales como la estructura del dispositivo (*modelo asociado*) y la observación actual. Por este motivo, es una aproximación alternativa a la aproximación adaptada del MDG (de Koning and Bredeweg (1998)) vista anteriormente que, además, permite aislar los fallos directamente.

#### 2.3.5.1 Modelo del dominio del problema

El método de Tsybenko (**?**) caracteriza el dominio del problema en términos de entidades de conocimiento elementales C que son aplicables en ese dominio, y un conjunto de variables X que describen el estado actual del dominio. Con cada entidad de conocimiento C se asocia el estado inicial del dominio *In(C)* bajo el cuál el conocimiento es aplicable y el estado final del dominio *Out(C)*. Del mismo modo que en el método de K. de Koning et al. se representa el dominio del problema mediante una estructura basada en modelos para el diagnóstico del conocimiento del estudiante, en este método la descripción del dominio del problema puede ser reformulada en términos de componentes C con sus entradas *In(C)* y salidas *Out(C)*.

El proceso de resolución de un problema en concreto se representa en términos de circuitos de resolución de problemas (CRP). La solución de cualquier problema se puede representar como una secuencia de problemas primitivos o acciones que alcanzan las metas perseguidas. Cada acción primitiva es una aplicación de una porción de conocimiento C. En este formalismo,

la solución del problema completo corresponde al CRP P, que transforma el estado de dominio dado *In(P)* en el estado de dominio alcanzado *Out(P)*. Cada problema P puede representarse por el estado inicial del dominio *In(P)* y la meta buscada *Out(P)*. Resolver el problema P significa encontrar una secuencia de porciones de conocimiento primitivo C1,...,Ck (CRP) que transforma *In(P)* en *Out(P)*.

**Ejemplo 1**. Sea el dominio de la Geometría. Un objeto en geometría consiste en elementos primitivos tales como los lados, ángulos, y otros elementos que caracterizan al objeto. Los elementos primitivos pueden representarse por un conjunto de variables Var = (x1,...,xn). Estas variables de dominio tienen dos estados: el estado conocido, cuando los valores se han asignado a las variables, y el estado desconocido. El conocimiento de dominio consiste en leyes, fórmulas y teoremas que describen las dependencias entre los objetos y elementos primitivos. Aplicando el conocimiento de dominio a las variables en estado conocido, se pueden inferir los valores de las variables en estado desconocido. Un problema de tutoría típico se describe en términos de ciertos datos dados y un estado al que llegar. Por ejemplo, considérese la resolución del problema del triángulo representado en la figura 2.29. Se proporciona al estudiante los valores de dos ángulos y la longitud de uno de los lados y se le propone averiguar las longitudes de los otros dos lados.

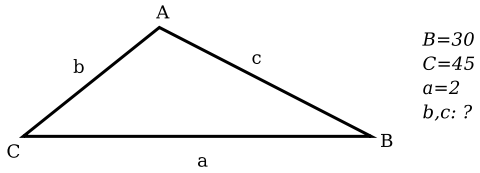


**Figura 2.29:** Triángulo ABC

#### 2.3.5.2 Generación de modelos asociados

La primera fase para modelar al estudiante consiste en construir un modelo para él. Este método genera el modelo del estudiante para dominios de resolución de problemas. El modelo debe representar conocimientos del estudiante y técnicas de resolución de problemas.

La mayoría de las aproximaciones al ME suponen que la forma en que un estudiante aborda un problema viene determinada por los expertos. Sin embargo, pueden existir muchas soluciones a un problema y los estudiantes pueden aplicar sus propios planes de resolución de problemas. Por este motivo, durante la etapa de generación del modelo, este método permite al estudiante preparar su propio plan de resolución. El sistema plantea un problema y proporciona un menú con partes de conocimiento necesario: reglas, fórmulas, etc. Se propone al estudiante construir su propia solución. El motor de inferencia guía al estudiante durante la resolución del problema. Si esta solución no es óptima, el estudiante debe realizar las mejoras necesarias. La versión resultante del CRP del estudiante, llamada *modelo asociado al problema P*, se añade al ME y es una característica individual de un estudiante que refleja su propia experiencia (los conocimientos que el estudiante ha aprendido) en la resolución de un problema P. El modelo tiene una estructura jerárquica explícita, es decir, el conocimiento del estudiante se puede representar en varios niveles de detalle, de tal modo que los problemas complejos pueden descomponerse en subproblemas. Este modelo es la base del DC.

#### 2.3.5.3 Generación del CRP que modela al estudiante

De acuerdo a los pasos anteriores, el método genera de forma automatizada circuitos de resolución del problema (dispositivos que modelan al estudiante). Para ello, se asume que se satisfacen las siguientes condiciones:

1. Un *dispositivo* no contiene bucles ni componentes innecesarias y

2. Un *dispositivo* consta de los componentes más apropiados para la resolución de un problema P.

El CRP compuesto del mínimo número de componentes puede parecer que constituye la solución óptima al problema P. Sin embargo, puede haber dos CRPs igual de cortos en longitud y de diferente complejidad. Para discriminar entre estos dos CRPs se asigna un peso numérico *w(C)* a cada conocimiento elemental C, cuyo valor corresponderá al nivel de complejidad de C. El peso P del CRP es la suma de los pesos de sus componentes. Un peso *w(C)* grande tiende a prohibir la aparición del conocimiento C en el CRP óptimo para el problema P, mientras que un peso pequeño indica que el conocimiento C es apropiado para la solución del problema P. Un CRP de peso mínimo representa la solución más simple al problema P.

Para obtener el CRP, el método se basa en el Modelo de Cálculo de Tyugu (Tyugu (1984)). El conjunto de trozos de conocimiento se combina con el problema P para producir el grafo solución del problema *G(P)*. Los nodos de este grafo se etiquetan con subconjuntos de variables de dominio y sus arcos con porciones de conocimiento elemental. El arco C une dos nodos x1,...,xr y x0. x1,...,xr si y sólo si los valores de las variables *x1*, ..., *xr* se conocen, y el valor de *x0* puede derivarse de los valores de *x1*, ..., *xr* bajo la aplicación del conocimiento C. El CRP para el problema P se corresponde con cada camino en el grafo *G(P)* que une el nodo etiquetado por el conjunto inicial de variables con el nodo cuya etiqueta contiene las variables de la meta buscada. El método propuesto permite generar el CRP apropiado para metas de tutoría, tales como CRP de peso mínimo, etc.

#### 2.3.5.4 Representación del ME mediante un modelo basado en suposiciones

El ME está basado en un modelo de la estructura y comportamiento del sistema a ser diagnosticado. Las discrepancias entre el comportamiento esperado y las observaciones actuales constituyen las bases para la obtención del diagnóstico.

Para cada porción de conocimiento primitivo o componente C, el ME contiene la suposición *bel(C)* de que un estudiante conoce esa porción de conocimiento C. Se asume que un estudiante ha aprendido la porción de conocimiento primitivo *C* si y sólo si el estado de dominio del problema contiene *In(C)*, y se alcanza *Out(C)* como resultado de aplicar *C*. Para controlar el comportamiento del componente *C* se tiene que chequear el estado inicial *In(C)* y el estado resultante *Out(C)* de las variables de dominio.

Para diagnosticar el estado cognitivo del estudiante se aplica el modelo asociado. Si un estudiante ha resuelto satisfactoriamente el problema P se asume que ha aprendido todos los arcos primitivos almacenados en el modelo asociado a P. Por ejemplo, siguiendo con el ejemplo anterior, si un estudiante ha resuelto satisfactoriamente el problema del triángulo (figura 2.29), el sistema asume que ha aprendido cómo calcular $\sin(\mathtt{X})$ para el ángulo X, las fórmulas: $\sin(\mathtt{A})/\mathtt{A} = \sin(\mathtt{B})/\mathtt{b}$ y $\mathtt{A}+\mathtt{B}+\mathtt{C} = 180$, etc. Esta información se representa en términos de creencias observadas por el sistema. Basándose en las creencias observadas, el sistema puede inferir

nuevas creencias (llamadas *creencias inferidas*). El conjunto de creencias observadas e inferidas forman el ME.

**Ejemplo 2**. La figura 2.30 presenta el CRP de peso mínimo para el problema del ejemplo 1. La descripción del circuito es un conjunto de sentencias de primer orden, que definen el comportamiento de los componentes del circuito y su conexión. Se adoptará la convención de Reiter (Reiter (1987)) de que AB(x) es un literal que establece cuándo el componente x es anormal.

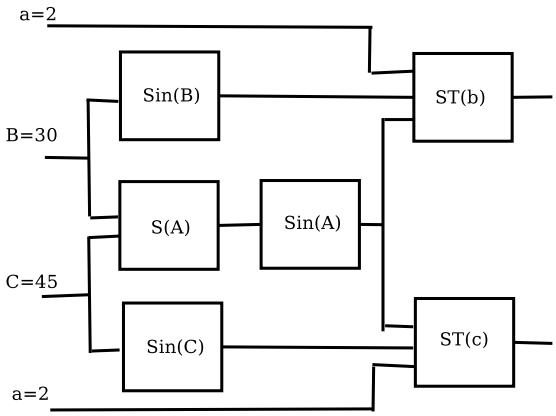


**Figura 2.30:** CRP para el problema del ejemplo

*Triángulo ABC*

La descripción del circuito de la figura anterior, viene dada por:

$$\sin(x) \Rightarrow (\neg AB(x) \Rightarrow out(x) = \sin(in(x)))$$
$$S(x) \Rightarrow (\neg AB(x) \Rightarrow out(x) = 180 - in1(x) - in2(x))$$
$$ST(x) \Rightarrow (\neg AB(x) \Rightarrow out(x) = in1(x) \star \sin(in2(x)) / \sin(in3(x)))$$

El diagnóstico comienza si un estudiante falla al resolver el problema P, es decir, cuando el comportamiento observado del estudiante contradice el comportamiento previsto por el modelo asociado al problema P. El diagnóstico cognitivo consiste en localizar los componentes (entidades de conocimiento primitivas) cuyo comportamiento anormal explican el comportamiento anormal del CRP de P .

**Ejemplo 3**. Suponiendo que la respuesta del estudiante es: b = 2, c = 2.8 para el problema P del Ejemplo 1, esta respuesta viola la predicción b = 1.05, c = 1.47 y nos indica que se ha producido un fallo del estudiante durante la resolución del problema. Para diagnosticar los errores del estudiante se aplica el modelo asociado al problema P representado en la Figura 4.8. Con Out(ST(b)) = 2, Out(ST(c)) = 2.8 en el CRP dado, hay seis posibles diagnósticos que cubren todos los componentes de dispositivo: [A], [Sin(A)], [Sin(B), Sin(C)], [Sin(B), ST(c)], [ST(b), Sin(C)], [ST(b), ST(c)], donde [C1,...,Ck] indican que las componentes C1,...,Ck han fallado. Por lo tanto, son necesarios pasos de diagnóstico adicionales para discriminar entre los diagnósticos que compiten.

#### 2.3.5.5 Método de descomposición-P para DC

Los métodos basados en modelos en IA aplican información adicional para guiar el proceso de diagnóstico. Entre estos métodos están la aproximación de MDG de Kleer y Williams

(De Kleer and Williams (1989)) que, como ya vimos anteriormente, aplica información probabilística para decidir qué medida tomar a continuación, las aproximaciones jerárquicas de Davis (Davis (1984)), Hamscher (Hamscher (1990)) y Mozetic (Mozetic (1991)), que aplican un diseño jerárquico multinivel para reducir el número de componentes consideradas en cada nivel; los modelos de teoría de fallos de Struss y Dressler (Struss and Dressler (1989)), que aplican el conocimiento de posibles fallos y sus consecuencias en el comportamiento. En general, en el diagnóstico técnico se da a priori una descripción precisa del sistema que se va a diagnosticar. A diferencia de este tipo de diagnóstico, en el DC los modelos asociados se generan dinámicamente durante la interacción con el estudiante de tal modo que, informaciones como probabilidades de fallo o estructuras jerárquicas para modelar al estudiante no están disponibles. Por esta razón, se aplica el método de descomposición-P de Tsybenko, que permite aislar componentes defectuosas en ausencia de otra información.

La idea básica de esta aproximación es reducir el diagnóstico de un sistema al diagnóstico independiente de sus subsistemas. Se realiza la descomposición secuencial hasta que los componentes defectuosos han sido aislados. Normalmente, se necesitan medidas adicionales para la descomposición-P, ya que el paso de generación de diagnóstico inicial probablemente no proporcione un único diagnóstico.

Los modelos asociados se generan de forma gráfica, donde los nodos principales del grafo representan los componentes del modelo y los arcos del grafo representan las conexiones entre los componentes. El comportamiento de un componente se describe mediante un conjunto de restricciones. Estos conjuntos de restricciones se conectan entre sí mediante variables compartidas. La observación actual se modela como la asignación de valor a la variable relacionada. De este modo, un grafo G para un modelo asociado tiene dos tipos de nodos: el conjunto de nodos principales *Comps*, que representa el conjunto de componentes del modelo y el conjunto de nodos auxiliares *Var*, que representa el conjunto de variables compartidas. El conjunto *Sd* son los arcos del grafo que representa las conexiones componente-variable. Un grafo conflicto para un modelo dado es un subgrafo mínimo *G(Obs)* de G que contiene aquellos caminos de G a través de los cuales se obtienen los conflictos mínimos para el modelo. Se supone que un camino conflicto comienza y finaliza en un nodo principal.

Un grafo conflicto permite representar tanto la topología de un modelo como un conjunto de observaciones realizadas en un modelo.

Las medidas se incorporan al formalismo del grafo conflicto: medir la salida del componente *C* implica quitar todos los arcos del grafo estructural conectados a la salida de la componente *C*, y viceversa, quitar los arcos conectados a la salida del componente *C* indica que la salida del componente *C* tiene que ser medida. Tsybenko demuestra que la selección de medida para la descomposición-P puede reducirse a la descomposición del grafo conflicto del sistema.

- *Teorema* (informal): Suponiendo que se quitan todos los arcos conectados a la salida del componente C, el problema de diagnóstico del grafo G conflicto supone la descomposición del grafo G en subgrafos desconectados. Luego la medida de la salida del componente C supone una descomposición del problema dado en subproblemas de diagnóstico independiente.

Si no está disponible ninguna información discriminatoria, entonces puede aplicarse una estrategia de selección de medidas semejante a una división por la mitad. La mejor medida siguiente es aquélla que proporciona una descomposición de un grafo conflicto para el problema de diagnóstico dado en partes iguales.

El método de descomposición-P está guiado por pura información de primeros principios, tal como la descripción del modelo y el comportamiento observado del modelo. La eficiencia de la aproximación propuesta puede incrementarse si se tiene en cuenta alguna clase de criterio de preferencia (probabilidades de fallo de componentes, número de componentes sospechosos en un diagnóstico, etc.). La idea básica del método es la siguiente:

1. Aplicar un criterio de preferencia para determinar aquellas partes del dispositivo donde se localizan los fallos preferidos.

2. Realizar la descomposición-P para separar aquéllas partes del dispositivo completo.

Para aislar los componentes defectuosos se aplica el siguiente algoritmo, que es una descomposición-P secuencial de las partes sospechosas de un modelo:

**Algoritmo 2.1:** Algoritmo de descomposición-P de Y. Tsybenko

```
ListSis = Problema de diagnóstico;
while ListSis ≠ ∅ do
    foreach subproblema S ∈ ListSist do
        Seleccionar los puntos que llevan a la descomposición de S en subproblemas de
        diagnóstico independientes, y medir estos puntos;
        foreach nuevo subproblema do
            if subproblema no muestra mal comportamiento then
                Excluir subproblema de la consideración;
            else if subproblema sospechoso consta de componentes simples then
                Los componentes defectuosos se aislan;
            else
                Añadir subproblema a ListSis;
```

Las definiciones y teoremas que resumen brevemente la aproximación formal de la descomposición-P de Tsybenko se detallan en **?**.

**Ejemplo 4**. La Figura 2.31a representa el grafo G estructural para el CRP del Ejemplo 1. Quitar los arcos conectados al componente *Sin(C)* supone la descomposición del grafo *G* en tres subgrafos desconectados. Medir la salida del componente *C* supone la descomposición-P del problema de diagnóstico dado en tres subproblemas independientes: *Sub(Sin(B), ST(b))*, *Sub(S(A), Sin(A))*, y *Sub(Sin(C), ST(c))*, donde el término Sub (C1,...,Ck) indica que el subproblema esta compuestos de los componentes C1,...,Ck (Figura 2.31b).

En el ejemplo 2 se ha visto que el paso de generación de diagnóstico inicial proporciona seis diagnósticos mínimos que compiten. ¿Cuál de estos candidatos es un diagnóstico actual? Para determinar un diagnóstico actual se aplica el método de descomposición-P. El ejemplo 3 muestra que la medida *Out(Sin(A))* es la mejor porque proporciona una descomposición en tres partes iguales. Para determinar la salida del componente *Sin(A)*, el sistema pide al estudiante que averigüe *sin(A)*, si B=30 y C=45. Suponiendo que el estudiante responde que *sin(A)* es 0.5, los subsistemas *Sub(Sin(B))*, *ST(b))* y *Sub(Sin(C), ST(c))* no muestran el comportamiento erróneo, por lo tanto pueden ser excluidos del diagnóstico. Sólo el subsistema *Sub(S(A), Sin(A))* necesita ser considerado. Hay dos diagnósticos mínimos: *[S(A)]* y *[Sin(A)]* para el subsistema anterior. Considerar el grafo estructural para el subsistema *Sub(Sin(A), S(A))* aparece en negrita en la Figura

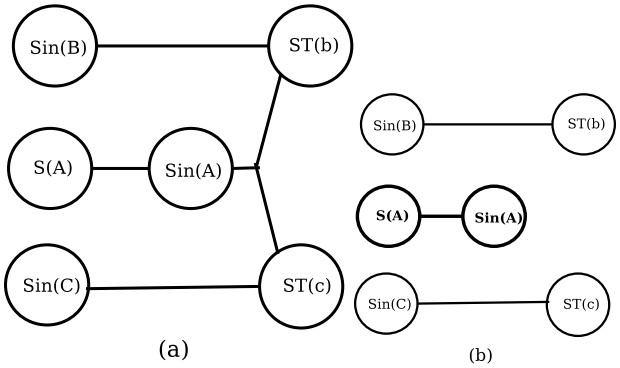


**Figura 2.31:** Grafo estructural para el CRP

del ejemplo 2 (a), y grafo estructural después de la descomposición-P (b)

2.31b. Quitar el arco entre los nodos *S(A)* y *Sin(A)* lleva a la descomposición del grafo en subgrafos de un elemento. De acuerdo a uno de los teoremas definidos en esta aproximación, medir la salida del componente *S(A)* necesariamente aisla los componentes defectuosos. Para discriminar entre diagnósticos que compiten *[S(A)]* y *[Sin(A)]* se propone al estudiante encontrar el ángulo A si B = 30 y C = 45. Suponiendo que el estudiante responde que A es 105, el componente defectuoso es *Sin(A)*, los otros componentes son normales. Sobre la base del diagnóstico del conocimiento del estudiante, un tutor deriva que: el estudiante ha aprendido las fórmulas *sin(A)/a = sin(B)/b* y *A+B+C = 180*, el estudiante sabe cómo obtener *sin(X)* para los ángulos B=30 y C=45, y el estudiante falla al averiguar *sin(A)* cuando A=105.

El método de Tsybenko se ha aplicado al diagnóstico cognitivo dentro del SIT llamado ARQUIMED (Tsybenko (1993)).

### 2.3.6 Sistema de diagnóstico SMDS

M. Ikeda et al. (Ikeda et al. (1993)) presentan un método de modelización denominado HSMIS (*H*ipothetical *S*tudent *M*odel *I*nference *S*ystem). Es un método no monótono cuyas principales características principales son: (a) modelo de inferencia basado en lógica de 4 valores de verdad, y (b) tratamiento de la no monotonía de las creencias del estudiante y del propio proceso de inferencia utilizando el ATMS de de Kleer (de Kleer (1986)).

Esta aproximación al ME es básicamente una inferencia inductiva que induce un modelo a partir de los datos observados, en este caso, las respuestas del estudiante a los problemas que se les plantean. El método propuesto se apoya en un lenguaje de descripción del ME denominado SMDL (*S*tudent *M*odel *D*escription *L*anguage) y el sistema de diagnóstico incluido en HSMIS se denomina SMDS (*S*tudent *M*odel *D*iagnosis *S*ystem).

La mayoría de los métodos de modelización del estudiante convencionales ya citados presentan algunas limitaciones y no consiguen un procedimiento de inferencia completo y aceptable. M. Ikeda et al. presentan una formalización del ME que satisface los siguientes requisitos:

- *Relación precisión-coste*: en general, cuanto mayor es la precisión del ME, más efectivo es el comportamiento del sistema. Sin embargo, existe una relación inversa entre la precisión del modelo y el coste de construirlo. Por lo tanto, esta relación inversa debe tenerse en cuenta para establecer un esquema de representación apropiado para el ME.

- *No monotonía*: el proceso de aprendizaje se logra fundamentalmente con el cambio de las mentes de los estudiantes. La consistencia de sus respuestas se puede perder fácilmente. Por esta razón, los métodos de modelización deberían ser capaces de controlar automáticamente la consistencia de dichas respuestas de igual forma que las mentes de los estudiantes. Hay muy pocos intentos de formular la no monotonía del proceso de modelización del estudiante.

- *Afirmaciones desconocidas*: si el estudiante falla al deducir su propia solución a un problema significa que no ha podido resolver el problema, pero no implica que no tenga ningún conocimiento. El módulo del ME debería usar esta afirmación como dato informativo acerca de su conocimiento, y construir un modelo que explique porqué no puede deducir la respuesta a partir de su propio conocimiento. Esto requiere un ME que deduzca afirmaciones desconocidas (*unknown*).

- *Fundamento teórico*: debería definirse un mecanismo de modelización del estudiante independiente del dominio y con un fundamento teórico que contribuya a la clarificación del ME y a la articulación de la escalabilidad y reusabilidad del mecanismo propuesto.

Para poder entender SMDS, el sistema de diagnóstico del ME incluido en HSMIS, es necesario ver previamente los fundamentos tanto de SMDL como del método de modelización presentado.

#### 2.3.6.1 SMDL: lenguaje de descripción del ME

SMDL es un lenguaje para modelar al estudiante ejecutable y que puede simular el comportamiento del estudiante en la resolución de un problema. Tanto SMDL como HSMIS están implementados en una versión extendida de Prolog (Shapiro (1983)), ESP -*E*xtended *S*elf-contained *P*rolog), y forman parte del entorno para SITs denominado FITS.

Para conseguir los requisitos anteriores, el ME requiere representar no sólo a los estudiantes sino también el nivel de comprensión de los estudiantes que tiene el sistema. Esto implica que el modelo tiene que distinguir dos estados: el sistema puede predecir el comportamiento del estudiante o el sistema no puede predecirlo. Cuando el modelo está basado en lógica, como en este caso, tiene que tener dos valores de verdad, verdadero y falso, que representen los dos estados anteriores. Empleando la terminología de Prolog, el valor *false* es tratado como *fail* (fallo) y el valor *verdadero* es tratado como *success* (éxito). Este último valor representaría el estado lógico del estudiante que puede ser: *true, false* o *unknown*, que representan que el estudiante cree que una afirmación es cierta, el estudiante cree que una afirmación es falsa, y el estudiante desconoce el valor de verdad de una afirmación respectivamente. La discriminación entre los tres valores se hace introduciendo un argumento auxiliar interpretado por un meta-intérprete.

Los hechos se representan en SMDL como se ve en los siguientes ejemplos:

```
templada(Paris, true).
torrida(Paris, false).
fertil(Paris, unknown).
```

Estos tres hechos representan que el estudiante cree que Paris no está en la zona tórrida sino en la zona templada y no sabe si es o no fértil. Si éstos son todos los hechos en el ME, también representa que el sistema no puede decir nada acerca del conocimiento del estudiante sobre otras afirmaciones.

Los predicados de SMDL son de la forma $\mathtt{p}(\mathtt{X}_1, \mathtt{X}_2, \ldots, \mathtt{X}_m, \mathtt{T})$, donde $p$ es un nombre de predicado y $\mathtt{X}_i$ $(1 \preceq i \preceq m)$ es una variable. La secuencia de variables $\mathtt{X}_1, \mathtt{X}_2, \ldots, \mathtt{X}_m$ se abreviará a partir de ahora en la descripción de este método como $\tilde{\mathtt{X}}$. $T$ es un objeto de valor de certeza o uno de los cuatro valores de verdad (*true*, *false*, *unknown* y *fail*).

Las cláusulas están escritas de la siguiente forma:

$$\mathtt{A} :: -\mathtt{B}_1, \mathtt{B}_2, \ldots, \mathtt{B}_K$$

donde $A, B_i$ $(1 \preceq i \preceq k)$ son predicados. Si k = 0, la cláusula es un *hecho*. $A$ se llama *cabecera* de la cláusula y $B_1, B_2, \ldots, B_k$ *cuerpo de la cláusula*.

Un *programa P* de SMDL es un conjunto finito de cláusulas. Por ejemplo:

```
crece(X,T4) :: -templada(X,T1).
crece(X,T5) :: -torrida(X,T2), humeda(X,T3).
```

Estas dos cláusulas muestran que el estudiante piensa *"si el lugar X está en la zona templada o en la zona tórrida y húmeda entonces la planta crece en X"*. Las cláusulas con la misma cabecera tienen una relación disyuntiva y los predicados en el cuerpo tienen una relación conjuntiva.

Dada una meta ? – `crece(paris,T)`, el intérprete SMDL llama a las submetas en este orden:

$$\mathtt{templada}(\mathtt{paris}, \mathtt{T}_1), \mathtt{torrida}(\mathtt{paris}, \mathtt{T}_2) \text{ y } \mathtt{humeda}(\mathtt{paris}, \mathtt{T}_3)$$

El valor de verdad $T$ de `crece(paris,T)` se obtiene, por lo tanto, de acuerdo a:

$$\mathrm{T} = \mathrm{T}_4 \vee \mathrm{T}_5 = \mathrm{T}_1 \vee (\mathrm{T}_2 \wedge \mathrm{T}_3) = true \vee (false \wedge unknown).$$

El proceso de ejecución de un programa SMDL se define de dos formas: *derivación débil* y *derivación fuerte*. La derivación débil es como el proceso de ejecución de Prolog, es decir, una meta se alcanza en esta derivación si existe al menos una cláusula que derive la meta. La derivación fuerte es más compleja, ya que una meta con valor de certeza $T$ se alcanza en esta derivación si todas las cláusulas *OR* unificables con la meta se aplican y el resultado de la evaluación OR es $T$. Las definiciones formales de este proceso pueden verse en Ikeda et al. (1993).

#### 2.3.6.2 HSMIS

HSMIS está basado en la inferencia inductiva (es una versión extendida de MIS (Shapiro (1983)), y describe el conocimiento del estudiante como un programa SMDL. Los algoritmos de inferencia inductivos, en general, suponen que los datos observados (*oráculos*) son consistentes. Sin embargo, en este caso, el sistema de inferencia del ME debe afrontar a veces datos inconsistentes que, en nuestro caso, son las respuestas del estudiante, debido a que el alumno puede cambiar su forma de pensar o cometer errores de descuido. Para resolver este problema, se emplea en HSMIS el ATMS. El diagrama de bloques de HSMIS se muestra en la figura 2.32.

Las principales tareas de cada bloque son:

- **ATMS**: controlar la consistencia de un conjunto de suposiciones (entorno) usadas por SMIS.
- **SMIS**: resolver el problema planteado a los estudiantes.
- **CRS**: resolver las inconsistencias identificadas cambiando el entorno.

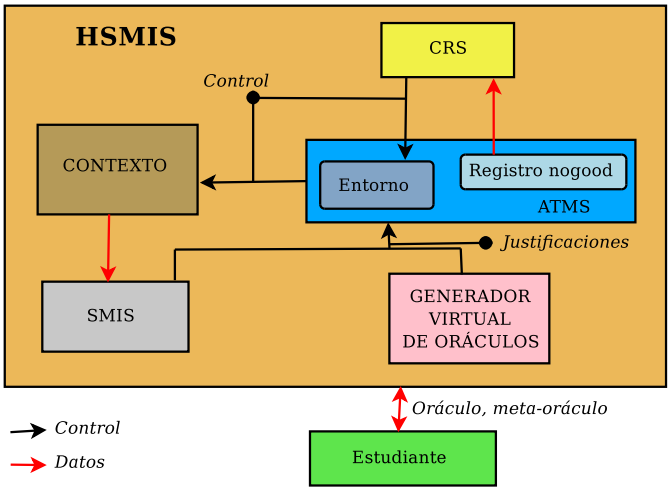


**Figura 2.32:** Diagrama de bloques de HSMIS

- **Generador virtual de oráculos**: mejorar la ejecución del modelo de inferencia. Sin hacer preguntas al estudiante, genera supuestas respuestas de éste basándose en la fiabilidad del alumno.

Observando el diagrama de bloques de la Figura 2.32, el comportamiento global del sistema es el siguiente:

El estudiante proporciona las respuestas (*oráculo real*) y el generador de oráculos virtuales genera *oráculos virtuales* si es necesario, y se los pasa a ATMS junto con los oráculos reales. Para gestionar la consistencia de los oráculos HSMIS se comporta de la siguiente forma:

1. SMIS informa a ATMS de todo el proceso de razonamiento, que lo registra en su base de datos. Cuando se detecta una contradicción, ATMS obtiene el entorno que causa la contradicción en función de la información proporcionada en este punto del proceso, y almacena este entorno en el registro no válido (*nogood record*) como registro contradicción.

2. SMIS invoca a CRS (*C*ontradiction *R*esolution *S*ystem) para que resuelva la inconsistencia y busque un entorno consistente.

3. Según la causa de la inconsistencia detectada, CRS genera candidatos del entorno consistente modificando o eliminando oráculos, y analiza la consistencia de estos candidatos invocando a ATMS para que compruebe su consistencia.

4. ATMS responde a la petición inspeccionando el registro no válido y

5. Si un candidato de un nuevo entorno es consistente, ATMS cambia el entorno actual por él, de tal modo que SMIS ya pueda trabajar con datos derivados de este entorno consistente y

6. SMIS continua la inferencia inductiva con el nuevo entorno.

Por lo tanto, básicamente HSMIS consta de SMIS y de módulos de gestión de las contradicciones. Ambos se detallan a continuación.

#### 2.3.6.3 SMIS: Motor de Inferencia Inductivo

En este sistema, un par problema-respuesta se denomina *oráculo* y se usa como dato a ser abarcado por el modelo obtenido. Formalmente, un oráculo es de la forma $< \mathtt{p}\left(\tilde{\mathtt{X}}', \mathtt{T}\right), \mathtt{T}' >$, donde $\tilde{\mathtt{X}}'$ es una secuencia de términos básicos, $\mathtt{T}' \in \{true, false, unknown\}$, y $T$ es una variable de verdad.

El término *abarcar* se define de la siguiente forma (Ikeda et al. (1993)):

***Definición 1***: la cláusula $\mathtt{C} = \mathtt{p}\left(\tilde{\mathtt{X}}, \mathtt{T}\right) :: - \mathtt{q}_1\left(\tilde{\mathtt{X}_1}, \mathtt{T}_1\right), \mathtt{q}_2\left(\tilde{X}_2, T_2\right), \ldots, q_k\left(\tilde{X}_k, T_k\right) \in \mathtt{P}$ abarca $\mathtt{p}\left(\tilde{\mathtt{X}}', \mathtt{T}'\right)$, cuando existe $\theta$ (sustitución) tal que:

$\mathtt{C}\theta = \mathtt{p}\left(\tilde{\mathtt{X}}', \mathtt{T}'\right) :: - \mathtt{q}_1\left(\tilde{\mathtt{X}_1}', \mathtt{T}_1'\right), \mathtt{q}_2\left(\tilde{\mathtt{X}_2}', \mathtt{T}_2'\right), \ldots, \mathtt{q}_k\left(\tilde{\mathtt{X}_k}', \mathtt{T}_k'\right), \mathtt{T}' = \wedge_{\mathtt{i}=1}^{\mathtt{k}} \mathtt{T}_\mathtt{i}'$ y

$\left\{< \mathtt{p}\left(\tilde{\mathtt{X}}', \mathtt{T}\right), \mathtt{T}' >\right\} \cup \left\{< \mathtt{q}_\mathtt{i}\left(\tilde{\mathtt{X}_\mathtt{i}}', \mathtt{T}_\mathtt{i}\right), \mathtt{T}_\mathtt{i}' > \mid 1 \leq \mathtt{i} \leq \mathtt{k}\right\} \subset \Omega$ (conjunto de oráculos dados al sistema).

Llamamos a $\mathtt{q}_1\left(\tilde{\mathtt{X}_1}', \mathtt{T}_1'\right), \mathtt{q}_2\left(\tilde{X}_2', T_2'\right), \ldots, q_k\left(\tilde{X}_k', T_k'\right)$ *traza de alto nivel* de la cláusula C para $\mathtt{p}\left(\tilde{\mathtt{X}}', \mathtt{T}'\right)$.

SMIS aplica el siguiente procedimiento repetitivamente al modelo, hasta que éste es capaz de abarcar todos los oráculos dados:

- Si hay una diferencia entre un oráculo y el hecho derivado del ME, se activa el sistema de diagnóstico del estudiante denominado en este sistema SMDS, que se describe a continuación. SMDS identificará la causa de la diferencia.
- De acuerdo al diagnóstico, SMIS selecciona la operación adecuada, o quitar una cláusula incorrecta o añadir una nueva cláusula, e informa de ello al ATMS.

Un ejemplo de traza de alto nivel se incluye en la Figura 2.33:

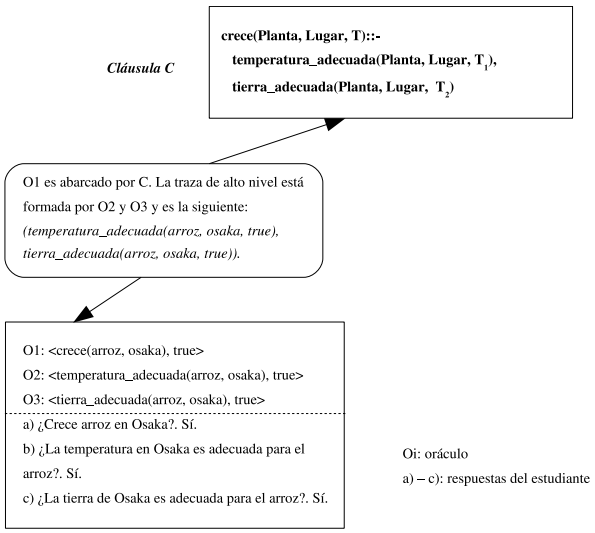


**Figura 2.33:** Ejemplo de traza de alto nivel para una cláusula

#### SMDS: Sistema de Diagnóstico del ME en SMIS

SMDS se encarga de obtener las trazas del proceso de derivación en el modelo actual y chequear los resultados con las respuestas del estudiante. SMDS identifica una cláusula incorrecta

que tiene una refutación lógica y también los oráculos no abarcados que deban serlo por el modelo. Para cubrirlos, SMIS busca una nueva cláusula a insertar en el modelo actual. Esta nueva cláusula debe tener un soporte lógico mediante oráculos. Si se tiene un oráculo no abarcado y una cláusula cuya cabecera y cuerpo se equiparan con algún oráculo entonces, la cláusula es añadida al modelo actual.

Para realizar las tareas SMDS tiene tres procedimientos: *ip*, *fp* y *failp*. SMDS, de forma selectiva, activa uno de estos procedimientos de acuerdo a la diferencia entre el valor de certeza del oráculo y el derivado del ME. El procedimiento *failp* de SMDS identifica la causa de la no existencia de completitud del modelo que es, o bien una meta no cubierta o una cláusula incorrecta, y dinámicamente decide qué procedimiento, bien *fp* o *ip*, debería activarse (en Ikeda and Mizoguchi (1992) aparece una explicación detallada de estos tres procedimientos de SMDS).

Para entender en detalle las tareas de los tres subprocedimientos de SMDS, los autores establecen formalmente un conjunto de definiciones (Ikeda et al. (1993)). A modo de ejemplo gráfico, en la Figura 2.34 se ha añadido, a la traza de alto nivel de la cláusula del ejemplo anterior, una *refutación* de *C*. La cláusula *C* incorrecta con una refutación detectada debería ser eliminada del modelo. De esta tarea se encarga el procedimiento *fp* de SMDS.

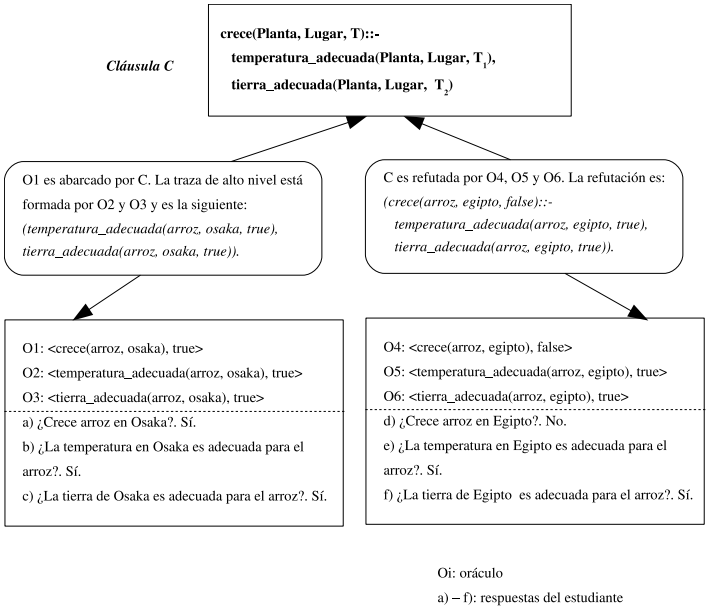


**Figura 2.34:** Ejemplo de traza de alto nivel y refutación para una cláusula

### *Búsqueda de una nueva cláusula*

Los candidatos de la cláusula a añadir se generan mediante operadores de refinamiento que se definen modificando los definidos en MIS (Shapiro (1983)). Un grafo de refinamiento es un grafo dirigido cuyos nodos son las cláusulas de SMDL y los arcos corresponden a operaciones de refinamiento. Los nodos hijos de un nodo se producen aplicando un operador de refinamiento a dicho nodo.

Para buscar una cláusula añadida al modelo se aplica una búsqueda en amplitud sobre el grafo de refinamiento, que tiene a la cláusula como uno de sus nodos. Para el arco dirigido $C \xrightarrow{\rho} C'$ en el grafo de refinamiento, se cumple la siguiente relación:

$\mathtt{C} \underset{\rightarrow}{\rho} \mathtt{C}'$: cuando la cláusula $C'$ abarca una meta $A$, la cláusula $C$ abarca $A$.

$\rho$: operador de refinamiento que obtiene $\mathtt{C}'$ a partir de la cláusula $C$.

La búsqueda en el grafo de refinamiento puede ser podada, es decir, cuando la cláusula no abarca A, las ramas que llevan a los descendientes de $C$ en el grafo pueden podarse ya que, si un nodo no está soportado lógicamente, sus nodos hijos no lo están tampoco.

Un grafo de refinamiento representa conocimiento que permite una búsqueda eficiente de una cláusula que se va a añadir al modelo pero no tiene ningún conocimiento a priori de errores. Dados algunos errores típicos específicos de la materia considerada, el procedimiento de búsqueda podría comenzar su tarea desde estos errores, haciendo que la búsqueda fuera más eficiente (esta característica puede verse como una deficiencia del método propuesto).

#### 2.3.6.4 Descripción del proceso de inferencia

La tarea de modelización realizada por HSMIS se define de la siguiente forma:
***Definición***. Cuando se proporciona un conjunto de oráculos $\Omega$, se infiere un modelo para el conjunto de oráculos que permite minimizar la distancia $\mathtt{D}(\Omega, \tilde{\Omega})$. $\tilde{\Omega}$ satisface la siguiente restricción:

$$\tilde{\Omega} = \{< \mathtt{A_i}, \tilde{\mathtt{T}}_\mathtt{i} > | \, \mathtt{1} \leq \mathtt{i} \leq \mathtt{k}\} \leq \mathtt{k} \text{ para } \Omega = \{< \mathtt{A_i}, \mathtt{T_i} \mid \mathtt{1} \leq \mathtt{i} \leq \mathtt{k}\}$$

La distancia $\mathtt{D}(\Omega, \tilde{\Omega})$ se define como el número de oráculos de $\Omega$ cuyos valores de certeza son diferentes de los correspondientes en $\tilde{\Omega}$.

Este concepto de distancia es útil cuando se detecta una contradicción. Como se verá más adelante, CRS busca un conjunto de oráculos consistente con la distancia mínima cuando se detecta una contradicción causada por un conjunto de oráculos inconsistentes.

Las condiciones de HSMIS para añadir al modelo la nueva cláusula:
$\mathtt{C} = \mathtt{p}(\tilde{\mathtt{X}}, \mathtt{T}) :: - \mathtt{q}_1(\tilde{\mathtt{X}_1}, \mathtt{T}_1), \mathtt{q}_2(\tilde{X_2}, T_2), \ldots, q_k(\tilde{X_k}, T_k)$, pueden describirse de la siguiente forma:

*if*
*cond1:* $\left(< \mathtt{p}\left(\tilde{\mathtt{X}'}, \mathtt{T}\right), \mathtt{T}' > \in \tilde{\Omega}\right)$ *y*
*cond2: (C es correcta) y*
*cond3: (C abarca* $\mathtt{p}\left(\tilde{\mathtt{X}'}, \mathtt{T}\right)$ *y*
*cond4: (C es la cláusula más general)*
*then*
*añadir C al modelo*

La existencia de las condiciones (*cond1* a *cond4*) previas son informadas por HSMIS a ATMS mediante diferentes nodos suposición o justificaciones de ATMS, que se detallan en Ikeda et al. (1993). Como ejemplo, se presenta a continuación la justificación asociada a *cond3*:

*cond3*: La condición significa que C tiene una traza de alto nivel correcta para $\mathtt{p}\left(\tilde{\mathtt{X}'}, \mathtt{T}'\right)$ en $\tilde{\Omega}$. HSMIS informa a ATMS de su existencia de este modo:

$$\begin{matrix} oracle\left(p\left(\tilde{X'}, T\right), T'\right) \\ oracle\left(q_1(\tilde{X_1}, T_1)\theta, T_1'\right) \\ \vdots \\ oracle\left(q_k(\tilde{X_k}, T_k)\theta, T_k'\right) \end{matrix} \Rightarrow cover\left(C, p\left(\tilde{X'}, T'\right)\right)$$

Del mismo modo, cuando se identifica una refutación para una cláusula $C$ por SMDS, ATMS es informado con otro tipo de justificación, y cuando se comprueba que $\mathtt{C}'$ no abarca $\mathtt{p}\left(\tilde{\mathtt{X}}', \mathtt{T}'\right)$, HSMIS informa a ATMS de ello con otro tipo suposición, que se muestra a continuación como ejemplo:

$< uncover\left(C', p\left(\tilde{X}', T'\right)\right), \{\{a_i\}\}, \{\{\mathtt{a_i}\}\} >$

donde, al igual que en un nodo ATMS, $uncover\left(C', p\left(\tilde{X}', T'\right)\right)$ es el dato usado en el sistema de razonamiento, $\{\{a_i\}\}$ es un conjunto de entornos que mantienen el dato y $\{\{a_i\}\}$ también es el conjunto de justificaciones. En este caso, se expresa el identificador de la suposición como $a_i$.

**Control del proceso de modelización**

HSMIS intenta modelar al estudiante a partir de su comportamiento planteándole automáticamente preguntas para evitar la ambigüedad en la selección de cláusulas alternativas y en el diagnóstico, y así obtener una tutoría más adecuada. Este objetivo requiere incluir ciertos mecanismos de control en el comportamiento de HSMIS que se describen a continuación:

- *Oráculos virtuales*: para el modelo inicial hay dos alternativas. Una es el modelo vacío, que significa que el profesor no sabe nada sobre el estudiante de antemano y la otra alternativa es el conocimiento completo de él (material de enseñanza). Esto último implica que el profesor asume que el estudiante comprende el material muy bien. Aunque la primera alternativa es muy razonable, el sistema tiende a realizar muchas preguntas para conseguir mucha información de cómo de bien comprende el estudiante el material. Por otro lado, la segunda alternativa no necesita muchas preguntas, al menos para estudiantes excelentes, ya que el modelo puede explicar su comportamiento correcto. En HSMIS se emplea esta última alternativa ya que la característica anterior es muy razonable, según los autores, en tutoría real.

  Sin embargo, hay un problema que hay que solventar y es que, uno puede incluir una cláusula en el ME sin ninguna justificación. La forma de solucionar este problema es mediante el generador de oráculos virtuales, que genera respuestas del estudiante plausibles basándose en la fiabilidad del ME actual, en vez de realizar preguntas. El tutor en este escenario plantea menos preguntas sustituyendo la información necesaria con respuestas correctas. A este tipo de oráculos se les denomina oráculos virtuales.

  Por cada oráculo virtual, se informa a ATMS de su generación con otra justificación:

- *Meta-Oráculos*: Los estudiantes a veces quieren expresar su conocimiento en forma no de hechos sino de conocimientos. Y el sistema algunas veces quiere preguntar al estudiante por qué sus respuestas son de esa forma. Por ejemplo:

  ```
  System: Does rice grow in Russia?
  Student: Yes, it does.
  System: Why do you thing rice grows in Russia?
  Student: It has wide flat field and river.
  ```

  En este caso, HSMIS puede obtener un oráculo y una cláusula de la siguiente forma:

  *< grow(rice, russia), T), true >*

  ```
  grow(rice, Place) ::-
      flat_field(Place),
  ```

```
river(Place).
```

La cláusula obtenida del estudiante se llama meta-oráculo. Cuando se añade la cláusula al modelo basado en el meta-oráculo, HSMIS informa a ATMS con un tipo de justificación y, de forma similar, cuando la cláusula C se elimina del modelo basado en un meta-oráculo, HSMIS informa a ATMS con otro tipo de justificación.

#### 2.3.6.5 Tipos de contradicciones del estudiante

Ya se ha visto anteriormente que la respuesta del estudiante a una pregunta se representa en HSMIS mediante un par hecho-valor de certeza y se denomina oráculo. Un conjunto de oráculos obtenidos por observación del comportamiento del estudiante en un cierto periodo tiende a ser inconsistente por varias razones. Este comportamiento inconsistente se puede clasificar de acuerdo a los siguientes criterios (Kono et al. (1994)).

- *Desde el punto de vista de la causa de la contradicción*:

1. *Contradicciones del estudiante*. Son contradicciones relacionadas con la no monotonía del comportamiento de un estudiante o de su propio conocimiento.
   - 1.1 *Contradicciones de oráculos causadas por cambios en la mente de los estudiantes*. La consistencia en las respuestas del estudiante puede perderse fácilmente debido a que, durante su periodo de aprendizaje, el estudiante adquiere nuevos conocimientos que causan cambios en sus mentes. Su comportamiento se basa en el conocimiento actual independientemente de su conocimiento previo.
   - 1.2 *Contradicciones de oráculos causadas por errores*. El conjunto de oráculos que contiene errores cometidos por descuidos del estudiante es inconsistente con su conocimiento actual.
   - 1.3 *Contradicciones del conocimiento del estudiante*. Un estudiante a veces tiene conocimiento inconsistente en su cabeza que también causa oráculos contradictorios.
2. *Contradicciones por suposiciones en el modelado*. El proceso de modelado del estudiante es básicamente hipotético. Por lo tanto, la completitud de un ME inferido no está siempre garantizada, es decir, el modelo no representa completamente el estado actual de conocimiento del estudiante. Una respuesta del estudiante deducida a partir del ME actual puede ser a menudo diferente de los nuevos oráculos. Por lo tanto, las suposiciones que fueron asumidas cuando se infirió el modelo actual llegan a ser inconsistentes con el conjunto de oráculos. Son pues contradicciones causadas por inconsistencias entre el conocimiento que tiene actualmente un estudiante y el conocimiento representado en el ME.

- *Desde el punto de vista del tratamiento de la contradicción*:

1. *Contradicciones de mundo simple*. Las contradicciones del tipo 1.1, 1.2 y 2 deberían resolverse revisando el ME.
2. *Contradicciones de múltiples mundos*. Las contradicciones del tipo 1.3 no deberían resolverse, sino que el conocimiento contradictorio del estudiante debería representarse para utilizarlo de forma efectiva en la tutoría.

Las clasificaciones anteriores quedan resumidas en la Tabla 2.6.

Tabla 2.6: Clasificación de contradicciones en el modelado del estudiante

| CRITERIO DE CLASIFICACIÓN | NOMBRE DE LA CONTRADICCIÓN | TIPOS |
|---|---|---|
| *Causa de la contradicción* | Contradicción del estudiante<br>Contradicción del modelado | 1<br>2 |
| *Tratamiento de la contradicción* | Contradicción de mundo simple<br>Contradicción de múltiples mundos | 1.1, 1.2, y 2<br>1.3 |

#### 2.3.6.6 Tratamiento de las contradicciones

Una inconsistencia entre el entendimiento actual del estudiante y un ME es causada por suposiciones defectuosas que fueron tomadas como hipótesis en un paso de inferencia previo y han sido creídas. Es necesario pues, un proceso de modelado del estudiante no monótono para eliminar estas inconsistencias (oráculos que no son consistentes con el entendimiento actual) del conjunto de suposiciones actual. El sistema de modelado deberá intentar construir un modelo consistente en un único mundo. Por este motivo, a las contradicciones de los tipos 1.1, 1.2 y 2 se les denomina *contradicciones de mundo simple*.

La metodología para resolver *contradicciones de mundo simple* es la siguiente:

1. Establecer las suposiciones necesarias para construir el ME que satisface el conjunto de oráculos de cada fase de modelado del estudiante.
2. Derivar el modelo a partir de las suposiciones y registrar el proceso de derivación.
3. Si se detecta una inconsistencia entre el conjunto de oráculos y el modelo, se debe encontrar el conjunto de suposiciones que causan la inconsistencia.
4. Resolver la contradicción revisando las suposiciones hechas por el sistema en el conjunto y continuar el modelado.

Los cuatro pasos anteriores de la revisión de creencias hacen que ATMS sea apropiado como módulo central en el mecanismo de control para el proceso inductivo de modelado del estudiante (tal y como se vio en la sección 2.3.6.2, al describir la arquitectura de HSMIS). El sistema de inferencia ejecuta la resolución del problema e informa a ATMS de su proceso de inferencia.ATMS mantiene y revisa un conjunto de suposiciones válidas que son el origen del proceso de inferencia y derivación de datos del sistema de inferencia. Cuando se informa de la derivación de una contradicción, ATMS calcula el conjunto de suposiciones que causa la contradicción mediante encadenamiento hacia atrás de los caminos de derivación informados a partir de la contradicción. Cuando una suposición es denegada, los datos que dependen de ella no pueden seguir manteniéndose y son automáticamente denegados por ATMS.

Las contradicciones de conocimiento del estudiante (tipo 1.3), que son contradicciones del propio conocimiento del estudiante, no deben ser resueltas sino que deberían ser representadas tal y como son. Estas contradicciones se representan sobre las bases de lógica de múltiples mundos y por ello se les denomina *contradicción de múltiples mundos* en contraposición

a las contradicciones de mundo individual. Basadas en la formulación de mundos múltiples, una contradicción de conocimiento del estudiante puede ser también definida como una clase de inconsistencia que aparece entre algunas suposiciones que han sido asumidas en el proceso de inferencia, de manera similar a la contradicción de mundo simple. HSMIS no es capaz de abordar este tipo de contradicciones sólo las contradicciones de mundo simple. Sin embargo, el sistema de modelado del estudiante THEMIS, basado en HSMIS, sí es capaz de tratarlas mediante una estructura de árbol de discriminación de conceptos (Kono et al. (1994)).

#### 2.3.6.7 Heurísticas para distinguir contradicciones

Es difícil no sólo para los sistemas de modelado sino también para los profesores distinguir y detectar los cuatro tipos de contradicciones vistos ya que sus indicios son muy similares. En HSMIS no se ha desarrollado en profundidad una metodología para diferenciar los tipos de contradicciones, pero se han empleado algunas heurísticas que se describen a continuación (Ikeda et al. (1993)).

La diferencia en el tratamiento entre las contradicciones de mundo simple y las de múltiples mundos sugiere la siguiente forma de discriminación entre estos dos tipos de contradicciones: asumiendo que la certeza de cada oráculo y cláusula dados pueden estar disponibles en el ME, si la certeza de una cláusula que es inconsistente con los oráculos dados o las certezas de algunos de los oráculos son menores que un cierto umbral, la inconsistencia debería ser considerada como una contradicción de mundo simple y, por lo tanto, debería resolverse. Por otro lado, si las certezas son suficientemente altas, la inconsistencias se consideran contradicciones de conocimiento del estudiante y, por lo tanto, no son revisadas sino incluidas en algunos mundos, es decir, todos los datos fiables pueden estar presentes en la formulación de múltiples mundos.

Para detectar las contradicciones de cada subcategoría de las contradicciones de mundo simple se incorporan las siguientes heurísticas:

El cambio en el conocimiento del estudiante que causa el tipo de contradicción 1.1 sucede especialmente justo después de que los errores del estudiante son corregidos. El estudiante cambia entonces su entendimiento desde un estado erróneo a un estado correcto. Es apropiado aplicar procedimientos de revisión para el tipo de contradicción 1.1 cuando los oráculos correctos se obtienen justo después de la tutoría, es decir, el sistema resuelve la contradicción o bien excluyendo los oráculos pasados inconsistentes con las cláusulas correctas, o preguntándole los valores de certeza de los oráculos. De este modo, las cláusulas erróneas son eliminadas y las cláusulas correctas son añadidas.

En el caso de que el estudiante hubiera cometido errores por descuidos que causen contradicciones del tipo 1.2, el estudiante debería haber aplicado consecuentemente las cláusulas que son inconsistentes con oráculos obtenidos durante un cierto periodo. Este tipo de contradicciones es detectado por criterios similares a los de las contradicciones de conocimiento del estudiante, es decir, los oráculos inconsistentes y cláusulas deberían ser suficientemente fiables. Hay dos formas de distinguir estas contradicciones: una es considerar una situación como un tipo 1.2 sólo cuando la situación no pudiera ser tratada como una contradicción de conocimiento del estudiante, y otra es hacer una pregunta muy similar al estudiante para obtener una confirmación. Sin embargo, estas contradicciones pueden ser distinguidas suficientemente introduciendo heurísticas dependientes del dominio, por ejemplo, el estudiante tiende a tomar un movimiento uniformemente acelerado por un movimiento uniforme si el movimiento es vertical. Estas heurísticas podrían ser añadidas a las heurísticas independientes del dominio anteriores.

Hay un punto más que, según los autores, debiera considerarse en el diseño de sistemas de modelado del estudiante. Puede asumirse que hay estudiantes que se comportan con dificultad de forma persistente, porque tienen baja capacidad o por una selección inapropiada del sistema del nivel de la tarea. En este caso no tiene sentido dejar que el estudiante complete la tarea actual. Es posible detectar este estado del estudiante diagnosticando el registro pasado de oráculos adquiridos. En estos casos, el sistema de modelado debería abandonar el modelado del estudiante e informar al tutor del fallo, así como dejar al estudiante regresar a tareas elementales.

#### 2.3.6.8 Detección de contradicciones

Las contradicciones derivadas en el proceso de inferencia de HSMIS se clasifican en siete tipos de los que, a modo de ejemplo, se extraen a continuación dos:

- *Contradicción de test de cobertura*: cuando se encuentra una traza de alto nivel correcta para la cláusula debido a la aparición de nuevos oráculos, se detecta la contradicción mediante la siguiente regla: $uncover\,(C,\,A)\,\&\,cover\,(C,\,A) \Rightarrow \perp$
- *Contradicción de oráculo*: cuando $< A,\,T_1 > \in\,\Omega, < A, \tilde{T}_2 > \in\,\tilde{\Omega}$ y $T_1 \neq T_2$, el entorno del oráculo es inconsistente. La regla de detección de la inconsistencia es la siguiente:

  $oracle\,(A,\,T_1)\,\&\,oracle\,(A,\,T_2)\,\&\,T_1 \neq T_2 \Rightarrow \perp$

Cuando cualquiera de estas contradicciones se detecta, ATMS es informada de ella y actualiza en registro no válido (*nogood record*).

Para abordar los siete tipos de contradicciones de mundo simple anteriores, en HSMIS se formulan, además, un conjunto de suposiciones en el marco de ATMS:

- La mayoría de estas suposiciones (Ω-*consistent(C)*, *uncover(C, A)*, *general(C)*, *trust(C)* y *metaOracle(C)*), se llaman *suposiciones por defecto*, que significa que "se asume que son *in* mientras no se encuentre ninguna evidencia en contra". El conjunto de estas suposiciones incluidas en el entorno se denomina *entorno por defecto* (denotado por $D_e$). A partir de éste, el entorno actual Ce puede expresarse de la siguiente forma: $C_e = D_e \cup \tilde{\Omega}$.

  Por ejemplo, para abordar una *contradicción de test de cobertura*, se formula la siguiente suposición:

  **uncover(C, O)**: representa que un oráculo *O* no puede ser abarcado por una cierta cláusula *C*. Esta suposición se mantiene a menos que *cover(C, O)* se derive (se usa para podar las ramas en el grafo de refinamiento y detectar así que la búsqueda de una cláusula para cubrir una meta presenta un fallo. Este fallo se trata también como una contradicción).

- El sistema debería ser capaz de generar el conjunto de oráculos $\tilde{\Omega}$ fiables actualmente en el modelo de inferencia a partir del conjunto completo de oráculos dados Ω. Para ello, se añade la siguiente suposición:

  **oracle(O)**: representa una pregunta y la correspondiente respuesta del estudiante a la misma, es decir, un oráculo O real. Si la suposición *oracle(O)* es *in*, el oráculo O está en $\tilde{\Omega}$. Esto significa que el oráculo es fiable en el presente.

El proceso de inferencia no monótono de HSMIS se realiza controlando el estado de las suposiciones. Estas suposiciones representan varias decisiones hipotéticas que se realizan durante el modelado del estudiante.

#### 2.3.6.9 Resolución de contradicciones

De acuerdo al tipo de suposiciones formuladas en HSMIS para abordar las contradicciones de mundo simple vistas anteriormente, las contradicciones pueden clasificarse en:

- *Contradicciones respecto al entorno por defecto*: el método de resolución estas contradicciones puede derivarse fácilmente de su definición. Se supone que una contradicción se detecta a partir de que existen ambos estados inconsistentes en el contexto actual, por ejemplo: *uncover(C, A)* y *cover(C, A)*. Cuando ya ha sido declarada la suposición *cover(C, A)*, ésta debe mantenerse hasta que el oráculo $A$ no pueda ser abarcado por la cláusula $C$, es decir, *uncover(C, A)*. Por lo tanto, la inconsistencia puede resolverse eliminando *cover(C, A)* desde el entorno por defecto.

- *Contradicciones respecto al entorno del oráculo* (correspondientes a $\tilde{\Omega}$): CRS genera $\tilde{\Omega}$ consistente. La consistencia está garantizada hasta el punto que no incluya contradicciones que han sido encontradas hasta el momento actual, es decir, puede contener una contradicción encontrada en el futuro. Las operaciones llevadas a cabo para la generación son: (a) eliminar un oráculo de $\tilde{\Omega}$ y (b) modificar el valor de certeza de un oráculo en $\tilde{\Omega}$.

CRS busca un $\tilde{\Omega}$ consistente (como ya se vio en la sección 2.3.6.3) en orden ascendente del número de modificaciones de $D\left(\Omega,\tilde{\Omega}\right)$ (distancia $D\left(\Omega,\tilde{\Omega}\right)$ mínima). Para mejorar la eficiencia y validez educativa en la selección del conjunto de oráculos consistente que tiene las modificaciones mínimas, se incorporan en el sistema algunas heurísticas independientes del dominio en el procedimiento de búsqueda: dar prioridad a las respuestas correctas o recientes, dar prioridad a los oráculos que mantienen la cláusula plausible que es soportada por relativamente muchos oráculos, etc. (2.3.6.7). Las heurísticas dependientes del dominio también podrían ser introducidas aunque no es realizado en HSMIS. Después de esto, el sistema continúa la inferencia sobre el nuevo entorno consistente que incluye $\tilde{\Omega}$.

## 2.4 Modelado del estudiante en Entornos Virtuales Inteligentes para la Formación/Entrenamiento

En este apartado se describen brevemente los Entornos Virtuales (EVs) y sus beneficios en el campo de la enseñanza para, posteriormente, analizar el estado de la cuestión del modelado del estudiante, en concreto, en Entornos Virtuales Inteligentes para la Formación/Entrenamiento (EVIEs) puesto que es uno de los tipo de sistemas a los que se aplica el modelado del estudiante propuesto en la presente tesis.

### 2.4.1 Entornos Virtuales en la Enseñanza

Las primeras aplicaciones a las que se les puede atribuir el calificativo de Realidad Virtual (RV) surgen aproximadamente a mediados del siglo XX. El término de Realidad Virtual, como muchos otros, ha sido definido de formas distintas en la literatura especializada. Cada autor hace hincapié, en mayor o menor medida, en alguna de las características principales que identifican este tipo de aplicaciones. Así, algunos autores consideran la RV, por ejemplo, sólo como una tecnología (véase, Heim (2000)). De hecho, el propio Heim define la RV de la siguiente manera:

*"A computer-generated simulation of the real or imagined environment or world"*

En esta definición, el mundo generado por ordenador en la RV puede ser, bien un modelo de un objeto del mundo real (v.gr., un laboratorio, una casa, etc.), bien un mundo abstracto, intangible pero comprendido por los humanos, como podría ser un mundo de ciencia ficción imaginario, un sistema operativo, un planeta, una molécula química, etc.

La mayoría de los autores actualmente van más allá y consideran el término RV desde una perspectiva más compleja, como una experiencia humana. De acuerdo a esta segunda visión (véase, Fizgerald y Riva (Fizgerald, 2001) se considera que la esencia de la RV es la relación de inclusión entre el participante y el entorno virtual, estableciendo además que:

> *"The basis for the Virtual Reality idea is that a computer can synthesize a three-dimensional (3D) graphical environment from numerical data"*

Desde esta segunda visión, el sentido de presencia es un aspecto clave en la RV (Slater (1999)) y depende de dos factores inmersión e interacción. El término de inmersión puede definirse (Cronin, 1999) como un factor humano:

> *"Effective immersion requires the ability on the part of the participant to control attention and focus on what is going on in the VE while simultaneously excluding all interference from the outside world"*

Según Sastry and Boyd (Sastry, 1998), en el sentido presencia, sobre todo en las aplicaciones del mundo real, influye más el nivel de interacción/interactividad que los actores experimentan en el entorno simulado. Es decir, influye más la posibilidad del usuario de "navegar, seleccionar, coger, moverse y manipular un objeto con más naturalidad", que la riqueza y fidelidad de las imágenes disponibles.

Las características ventajosas proporcionadas por los EVs, mencionadas en la sección anterior, han permitido establecer los escenarios en los que el aprendizaje en EVs puedan proporcionar beneficios sobre la educación tradicional para el aprendizaje de conceptos y habilidades. Esto está provocando un cambio tanto en los desarrolladores como en los educadores y, por lo tanto, la aparición de diversas plataformas de realidad virtual soporte para la educación y el entrenamiento en diferentes dominios de aprendizaje. Los beneficios proporcionados por el uso y adaptación de los EVs a la educación y al entrenamiento son numerosos (Mantovani et al. (2003)): *aprendizaje activo y experimental* (la interacción con los EVs fomenta la participación activa de los estudiantes), *aprendizaje en contextos imposibles o difíciles de llevar a la práctica en la vida real por otros medios* (por ejemplo, exploración de planetas, viajes dentro del cuerpo humano. También cuando el aprendizaje o entrenamiento puede resultar peligroso, como en contextos donde hay riesgo de lesiones para el estudiante, cuando las situaciones requieren el uso de equipos altamente caros o imposibles de obtener por otros medios, etc.), *aumento de la motivación* (por ejemplo, usando un formato de juego), *mejora de la colaboración* mediante EVs compartidos así como fomentar el aprendizaje de habilidades que pueden ser desarrolladas mejor a través de la experiencia de un grupo en un entorno común (por ejemplo, cuando la experiencia de crear un entorno simulado, o modelo es importante en el objetivo del aprendizaje), *adaptabilidad* (los EVs ofrecen la posibilidad de su adaptación a las características específicas de los estudiantes y a sus necesidades: diferentes velocidades y preferencias de estilos de aprendizaje, etc., así como la presentación más flexible a los estudiantes de un mayor conjunto de experiencias que las proporcionadas en entornos educativos .estándares"), *evaluación y valoración* (los EVs ofrecen un gran potencial como una herramienta de evaluación; cada sesión en estos entornos puede ser fácilmente monitorizada y registrada por los profesores, así como facilitar las tareas de valoración del aprendizaje), etc.

#### 2.4.1.1 Entornos Virtuales para la Formación/Entrenamiento

A finales de los años 90 surgió un nuevo campo de investigación denominado Entornos Virtuales Inteligentes para el Entrenamiento y/o Formación (EVIEs), que combinan dos áreas de gran complejidad: los EVs y los SITs. De este modo, los beneficios de los entornos 3D ya mencionados previamente, pueden así combinarse con los de los SITs (personalización de la presentación y contenido, adaptación de la estrategia de tutoría a los requerimientos de los estudiantes, etc.), para proporcionar soluciones educativas/de entrenamiento, con un valor añadido.

El uso de entrenadores basados en RV para el entrenamiento/formación de tareas estructuradas está siendo explotado en muchas áreas de aplicación, con especial interés, en los últimos años, en los sectores militar y médico. La mayoría de estos sistemas proporcionan un aprendizaje experimental y están enfocados principalmente al entrenamiento de los pasos específicos involucrados en el alcance de un procedimiento bastante estructurado.

Dado el enfoque de este trabajo, el análisis de estos sistemas se va a centrar en el modelado del estudiante. Como ya vimos anteriormente, el ME es uno de los módulos clave de un SIT y de especial complejidad puesto que debe representar el conocimiento del estudiante y, en la medida de lo posible, reflejar el proceso de razonamiento del estudiante. Asimismo, para poder inferir el estado del conocimiento del estudiante, es necesario, como también se vio, un mecanismo de diagnóstico cognitivo inherente al modelo. Esta complejidad aumenta cuando los SITs se aplican a EVs, como en el caso de los EVIEs, ya que proporcionan nuevas posibilidades de interacción que deben ser consideradas como nuevos elementos de información para todo el proceso educativo: el camino seguido por el estudiante durante su navegación a través de los escenarios 3D; el comportamiento no-verbal (como, por ejemplo, la dirección de la mirada); instrucciones o pistas que el módulo de tutoría puede proporcionar al estudiante directamente relacionadas con el entorno; nuevos tipos de preguntas que el estudiante puede formular en esta clase de entornos y cómo pueden influir en lo que el estudiante conoce a priori; etc. Esta nueva información es muy importante, no sólo para la evaluación del estudiante, sino también para el diagnóstico del estado de su conocimiento, al objeto de permitir al tutor software seleccionar la estrategia de tutoría más adecuada. Por esta razón, se requiere que la estructura del SIT, que está inserta en el EVIE, se enriquezca con todos los aspectos previamente citados (Imbert et al. (2007)), pero manteniendo un modelo conceptual claro, estructurado y bien definido.

La reciente aparición del campo de los EVIEs y su complejidad combinada con la de los SITs y, en concreto, con la del modelado del estudiante inmerso en este tipo de sistemas, ha dado lugar a que no hayan surgido muchas aproximaciones con un modelo conceptual bien definido, con una completa taxonomía y un método de diagnóstico asociado apropiado. Muchas de estas aproximaciones no consideran un ME explícito o completo tales como: ATEEG, un sistema de entrenamiento para controladores de tráfico aéreo (Alem and R. (1996)), STEVE (Soar Training Expert for Virtual Environments), un agente animado diseñado para ayudar a uno o varios estudiantes a realizar labores físicas que sigan un procedimiento determinado (Rickel and Lewis (1998), Lewis et al. (2000)), el entorno de aprendizaje virtual inteligente basado en web para entrenamiento de estudiantes en calidad industrial y mejora de procesos (WIVLE) propuesto por (Chi et al., 2006), el SIT para el modelo MASCARET (Buche et al. (2003), Querrec et al. (2004)) aplicado al entorno Sécurévi para entrenar a oficiales de bomberos, LAHYSTOTRAIN, un EVIE para entrenamiento de cirujanos (Los Arcos et al. (2000); Muller-Wittig et al., 2001), el sistema Ines para entrenamiento de enfermeras (Hospers et al. (2003)), el sistema PRVIR para protección radiológica en una central nuclear (Méndez et al., 2001), etc.

Centrándonos en el modelado del estudiante objeto de este trabajo, a continuación se des-

criben algunos EVIEs significativos. En primer lugar, como ejemplo de las numerosas aproximaciones que poseen un ME con las características previamente citadas, se presentan tres ejemplos: TLCTS, MASCARET y LAHYSTOTRAIN para, posteriormente, hacer hincapié en otros EVIEs con un modelado del estudiante mejor definido.

### 2.4.2 TLCTS

*Tactical Language and Culture Training System* (TLCTS) es un proyecto de la Universidad del Sur de California (USC) financiado por DARPA (Johnson (2007); Johnson and Valente (2008)). Su objetivo es la enseñanza de lenguas autóctonas a militares que participan en misiones fuera de su país donde, gran parte de los problemas se originan por el desconocimiento de la lengua y costumbres del país en el que se encuentran. Esta carencia provoca malentendidos y enfrentamientos con la población dificultando en definitiva la labor de los militares.

El objetivo principal de la aplicación es que los usuarios adquieran habilidades básicas fluidas en un tiempo mínimo, para poder interactuar adecuadamente con la población y realizar las misiones con el menor número de contratiempos. Para ello, la aplicación se ha estructurado en los siguientes tres tipos de actividades:

- *Skill Builder*: aplicación en la que los estudiantes pueden adquirir nuevas habilidades. Los usuarios aprenden poco a poco palabras y frases del nuevo idioma así como las situaciones en las que se pueden y deben utilizar. En esta aplicación pueden practicar la pronunciación y realizar ejercicios que conlleven hablar y entender el lenguaje que estudian y las habilidades aprendidas en cada una de las lecciones se pondrán a prueba en las otras dos actividades.

- *Arcade*: juego que consiste en desplazarse por un escenario siguiendo las instrucciones que da el ordenador en la lengua que se estudia. En ciertos momentos aparecen enemigos de distinto colores que se pueden matar empleando la palabra que identifica su color. El objetivo es que el estudiante adquiera fluidez en el uso de la lengua que está estudiando.

- *Misión*: ejercicio que reproduce una misión que, probablemente, tengan que realizar los militares desplegados en un país extranjero. En ella, se debe completar una tarea en un determinado escenario que supone relacionarse con la población. El alumno así, debe ir dando instrucciones de lo que quiere hacer y lograr el apoyo de la población para conseguir su objetivo. Con este fin, debe usar correctamente los conocimientos adquiridos anteriormente, usando frases correctas, con una adecuada pronunciación así como mostrando un conocimiento de las costumbres locales. En los escenarios de la misión, el usuario deberá interactuar con diversos personajes que aparecen en ellos. Estos personajes están controlados por agentes y su comportamiento viene dado por el diagrama de estados que lo modela.

Aunque el usuario puede alternar entre los tres tipos previos de actividades a voluntad, tiene a su disposición una guía que les ayuda a utilizarlos en el orden más idóneo para proporcionar mayores beneficios.

Actualmente, la aplicación soporta distintos tipos de clientes pero alguno de ellos en fase experimental. El único cliente que soporta la funcionalidad completa es el cliente monousuario en Windows denominado *Lapu*. Este cliente está construido sobre el motor del juego *Unreal Tournament* de tal modo que, el estudiante se comunica con los personajes de la misión a través


**Figura 2.35:** Ejemplo de entrenamiento de una misión en TLCTS

de un reconocedor de voz y de unos gestos que puede seleccionar en un menú de la pantalla. Hay un gestor de entradas que interpreta tanto la voz como los gestos realizados y los transforma en actos comunicativos. El motor de simulación social determina entonces la respuesta del entorno y de los personajes y un planificador de acciones envía estas respuestas al motor gráfico que las dibuja.

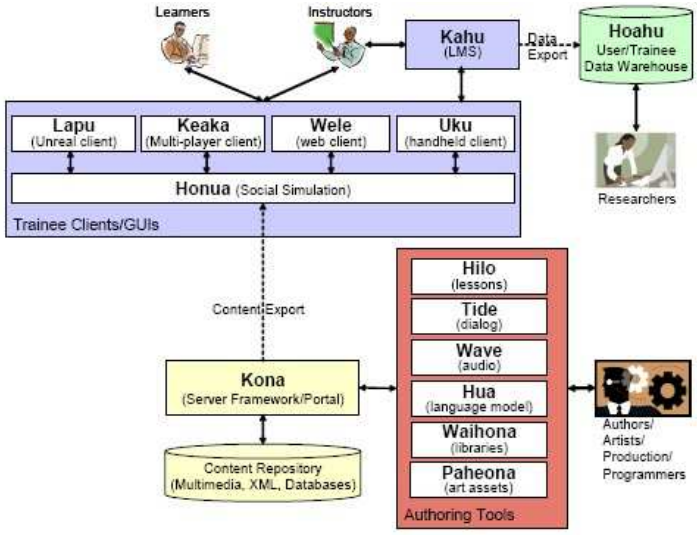


**Figura 2.36:** Arquitectura de TLCTS

Los clientes se conectan a un servidor web denominado *Kona*, en el que se almacena el contenido de los cursos en formato *xml*. Cada curso se representan como un libro formado por capítulos, que son las distintas lecciones de entrenamiento. También existe un sistema gestor de aprendizaje (LMS -*Learning Management System*) llamado *Kahu* que se comunica con los computadores cliente a través de una red local. *Kahu* proporciona y gestiona repositorios para

los perfiles de los estudiantes así como soporte para una fácil reconfiguración.

Mientras el estudiante interactúa con el sistema, un ME realiza una estimación probabilística del nivel de maestría de las capacidades del estudiante y el modelo se va actualizando. A la vez, se mantiene el *log* de acciones y se realizan las grabaciones de las frases del usuario para poder utilizarlas posteriormente.

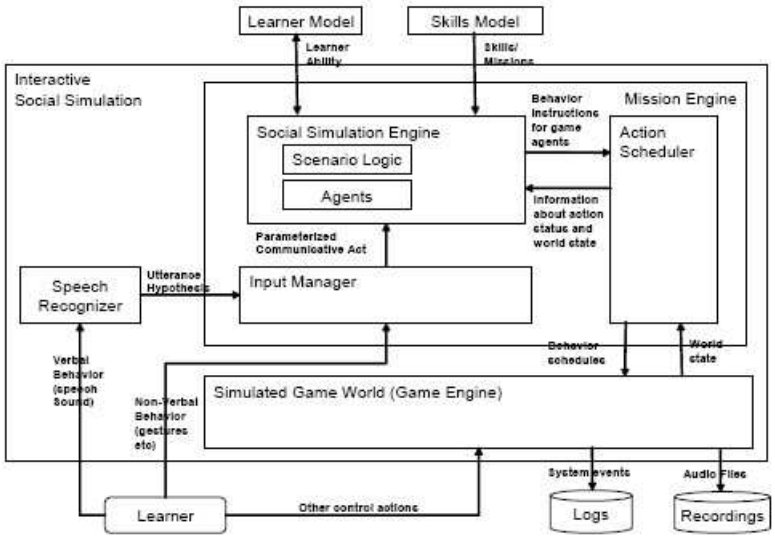


**Figura 2.37:** Arquitectura del cliente

*Lapu* para PC

El sistema se ha evaluado en diversos escenarios con resultados bastante positivos que han servido, además, para rediseñar algunas partes de la aplicación y hacerla más rápida, efectiva y usable. En concreto, se ha dotado a la aplicación de la posibilidad de recopilar archivos de log de las sesiones de entrenamiento y de las grabaciones de los usuarios permitiendo por un lado adaptar el sistema a la forma de uso de los alumnos y, por otro lado, mejorar la tasa de aciertos del reconocedor de voz en diversas situaciones.

### 2.4.3 MASCARET

MASCARET (*Multi-Agent Systems to simulate collaborative, Adaptive and Realistic Environments for Training*) es un sistema basado en agentes que integra un SIT para crear EVs de entrenamiento (Buche et al. (2003), Querrec et al. (2004)). Este sistema se ha utilizado para desarrollar Sécurévi, un entorno para entrenar a oficiales de bomberos en la dirección de operaciones.

MASCARET se ha diseñado en función de la pertenencia de los agentes a distintas organizaciones y se basa en los conceptos de agente, organización, rol y comportamiento. Su objetivo es integrar una pedagogía diferenciada basada en el contexto usando para ello los ITS. Para ello, MASCARET permite establecer los modelos necesarios para la creación de un ITS:

El *modelo de dominio* contiene el conocimiento del procedimiento a realizar y cómo realizarlo en el EV. Existe un agente encargado de mantener el conocimiento del dominio capaz de informar sobre las precondiciones y metas de una acción, el orden de las acciones y la responsabilidad de los intervinientes. Asimismo, maneja información sobre los distintos roles que intervienen en un fenómeno físico.

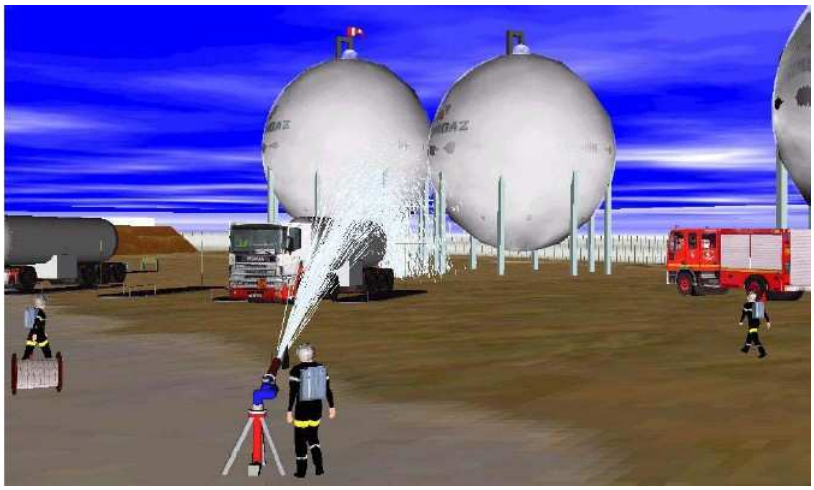

**Figura 2.38:** Entrenamiento en Sécurévi

El *modelo de errores* es un agente que maneja una base de conocimiento con errores típicos de los estudiantes. Esta información se considera crucial ya que el tutor tiene el conocimiento sobre el dominio (modelo de dominio) y el conocimiento sobre el estudiante (modelo del estudiante) y el modelo de errores se usa para identificar los errores cometidos por los estudiantes y averiguar su causa.

El *modelo del estudiante* proporciona en cada instante el estado del conocimiento del estudiante. En MASCARET hay tres componentes: *PsycologicalFeatures*, avatar y *Curriculum*. *PsycologicalFeatures* es la información relativa al nivel del estudiante -novato, experto, estado emocional -stress, etc. *Curriculum* asocia *Exercises* y *Errores*. Un *Exercise* está constituido por un *Context*, que instancia un entorno físico y social, y un *Scenario*. El avatar es un agente que maneja información acerca de las acciones del estudiante y proporciona el procedimiento colaborativo y el histórico del estudiante.

El *modelo pedagógico* define las estrategias pedagógicas y contiene información sobre cómo enseñar el conocimiento, mostrando especial interés por la estrategia cooperativa en la que los actores pedagógicos cooperan para la realización de las tareas, intercambian ideas acerca del problema concreto y comparten los mismos objetivos. El modelo pedagógico implica potencialmente a todos los demás agentes que colaboran entre sí, cada uno con su rol correspondiente (tutor, experto, compañero de equipo o fuente de errores), para dar un soporte pedagógico colaborativo.

El *modelo de interfaz* no se considera en la actualidad en MASCARET. La comunicación se realiza directamente desde los otros modelos del SIT.

En un ciclo de acción (figura 2.39), el estudiante realiza una acción observada por el agente pedagógico. El avatar del estudiante tiene el conocimiento sobre la ejecución de la acción actual y esta observación permite actualizar el modelo del estudiante. A continuación, se compara el modelo del estudiante con el experto y, si se detecta un error, se compara con el modelo de errores y se actualiza el modelo del estudiante. Finalmente, según la información sobre la causa del error, un agente pedagógico selecciona una acción a realizar y se le muestra al estudiante.

Realmente no hay un modelado explícito del estudiante en MASCARET sino que el modelo del estudiante se obtiene usando, de acuerdo a la forma previamente descrita, la detección activa de las acciones del estudiante y comparándolas con el modelo de dominio.

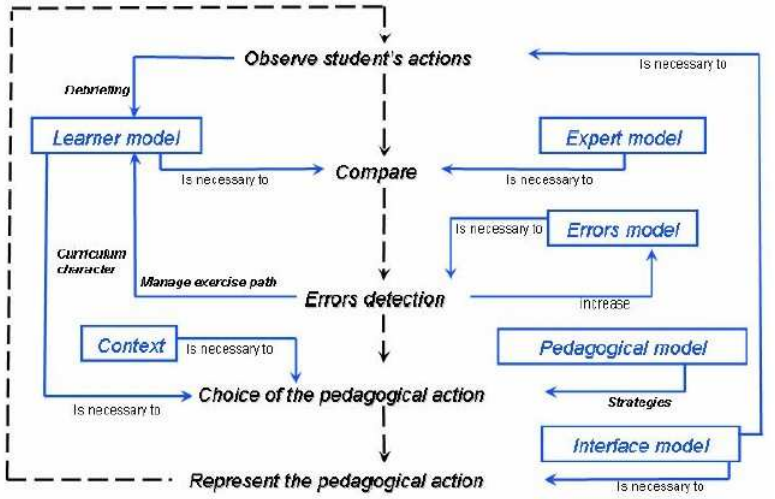


**Figura 2.39:** Comportamiento pedagógico

### 2.4.4 LAHYSTOTRAIN

Lahystotrain (*Integration of Virtual Environments and Intelligent Training Systems for Laparoscopy/Hysteroscopy Surgery Training*) es un sistema de entrenamiento de cirujanos en operaciones de laparoscopia e histeroscopia (Los Arcos et al. (2000); Müller-Wittig et al. (2001)). Teniendo en cuenta los riesgos que conlleva el entremiento en el propio quirófano y las cuestiones éticas implícitas en los experimentos con animales, el uso de un sistema de entrenamiento basado en realidad virtual tal como Lahystotrain ha supuesto una alternativa bastante atractiva para la formación de cirujanos en este área.


**Figura 2.40:** Interfaz del estudiante en Lahystotrain

El sistema admite dos tipos de usuarios: estudiantes, con cuatro niveles posibles de experiencia, e instructores, encargados de supervisar la evolución de los estudiantes y con capacidad para proponerles nuevos ejercicios y consultar información acerca de los distintos ejercicios y estudiantes.

La formación consta de dos pasos. En primer lugar, el estudiante adquiere los conocimientos teóricos relacionados con las patologías a tratar, y en una segunda fase pone a prueba esos conocimientos y adquiere habilidades prácticas mediante la realización de ciertas operaciones.

La aplicación se compone de una simulación de RV y un sistema de entrenamiento denominado ATS (*Advanced Training System*). ATS se compone de dos agentes pedagógicos, el asistente

y el tutor, tres agentes de apoyo (el cirujano auxiliar, la enfermera y el anestesista) que completan el equipo cuyo comportamiento se emula en las operaciones, y una interfaz de usuario. Esta interfaz se usa para que el estudiante solicite explicaciones, la intervención del asistente y el tutor así como la comunicación con la enfermera, el anestesista y el cirujano auxiliar. El estudiante debe aprender su papel dentro del equipo y la forma de coordinarlos.

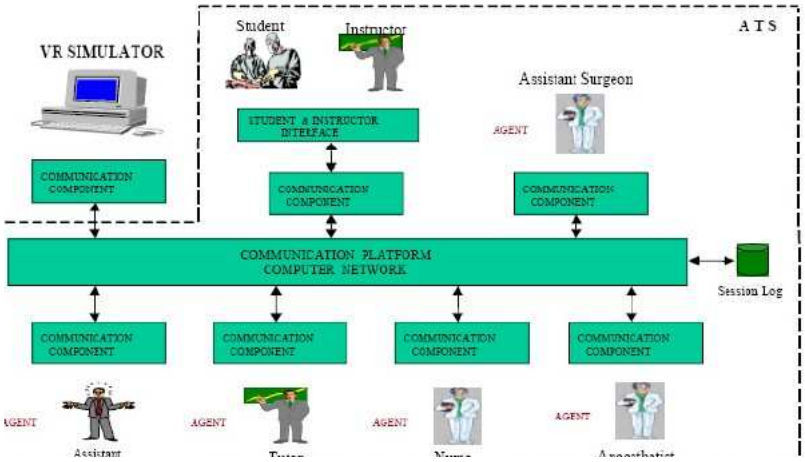


**Figura 2.41:** Arquitectura de Lahystotrain

Durante una intervención, el estudiante recibe la ayuda de un asistente que le proporciona explicaciones de forma proactiva (explicaciones proporcionadas a petición del estudiante que tratan distintos aspectos del procedimiento que se realiza como, por ejemplo, la mejor forma de acceder a un órgano) y explicaciones de forma reactiva (no se ofrecen a petición del estudiante sino cuando éste comete un error o sucede algo anómalo en la operación. Estas explicaciones pueden interrumpir y desconcentrar al estudiante por lo que se permite que el estudiante defina un nivel de interrupción).

Existe un agente tutor encargado de supervisar el entrenamiento. Este agente registra información del estudiante y datos de sus habilidades, genera el plan de instrucción que se crea y se modifica dinámicamente en función de la información obtenida del estudiante y de sus resultados durante la sesión actual.

Existe una base de datos (carpetas) que almacena el modelo del estudiante y contiene una parte estática y otra dinámica. La parte estática (no actualizada durante la sesión de entrenamiento) contiene información sobre las preferencias del estudiante (nivel de intrusión del tutor, medios preferidos para presentar explicaciones) y datos personales (nombre, edad, hospital, código de identificación, etc.). La parte dinámica incluye las operaciones realizadas por el estudiante durante su entrenamiento, ejercicios llevados a cabo, los errores cometidos y su conocimiento de las patologías y el instrumental usado.

El asistente se encarga de controlar la evolución del ejercicio: recibe del simulador las acciones del estudiante y el estado del paciente y, como consecuencia, envía las isntrucciones acerca de lo que debe suceder: acciones de los diversos agentes, complicaciones en la operación, etc. El agente asistente consta de un módulo monitor, un módulo de diagnóstico, un explicador y un avatar animado. Las acciones recibidas del simulador se pasan al monitor que las compara con los datos almacenados en la base de datos del dominio para detectar errores. El módulo de diagnóstico, implementado como un sistema basado en reglas con dos fases: generación de

hipótesis y validación, recibe los errores y trata de descubrir las causas. A continuación, el explicador genera el contenido y estructura de las explicaciones y gestiona el diálogo con el estudiante. Finalmente, el avatar animado se encarga de representar al tutor dándole la explicación correspondiente al estudiante (figura **??**).

La responsabilidad fundamental del tutor consiste en supervisar y gestionar el entrenamiento. El agente Tutor consta de un módulo supervisor de entrenamiento, un planificador pedagógico y el modelo del usuario o del estudiante (figura 2.42). El módulo supervisor de entrenamiento analiza las acciones del estudiante para evaluar sus errores y replanificar los objetivos de la sesión si es necesario (por ejemplo, si hay errores graves o si lo solicita el estudiante). El planificador pedagógico es el encargado de construir el plan de entrenamiento tanto al principio de la sesión como cuando lo solicita el supervisor de entrenamiento. Este plan se construye acorde con el conocimiento pedagógico y la información almacenada en el modelo del estudiante.

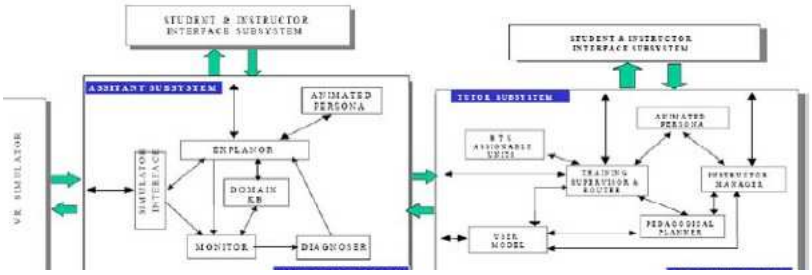


**Figura 2.42:** Arquitectura de los agentes Tutor y Asistente

Los tres agentes de apoyo tienen un comportamiento reactivo así como una descripción de su arquitectura pueden verse en (Arcos et al., 2000).

### 2.4.5 VI-MED

VI-MED (Mili et al. (2008)) es un entorno integrado basado en el juego para aprendizaje/entrenamiento virtual de estudiantes de enfermería y sin riesgo de dañar a pacientes reales. Este sistema tiene dos componentes principales: a) el juego, que simula un entorno de hospital con pacientes generados para cada sesión así como un rango de intervenciones que el jugador puede usar sobre los pacientes y b) un componente o subsistema de educación/evaluación (figura x).

figura de arquitectura general

El componente de juego está diseñado para proporcionar a los estudiantes de enfermería un entrenamiento virtual, ante o en paralelo con el entrenamiento real realizado en el hospital con pacientes reales. El ámbito de las sesiones de juego es establecido por la facultad de enfermería e individualizada de acuerdo a los logros de los estudiantes. Un juego imita una sesión de enfermería y consta de los siguientes pasos básicos:

1. *Registro de entrada*: el estudiante se registra y VI-MED lo carga en el perfil del estudiante.

2. *Generación de tareas*: VI-MED genera un conjunto de pacientes basados en el perfil del estudiante. El registro de los pacientes consiste en datos socio-demográficos (nombre, edad, peso, raza, etc.), condición médica (por qué fueron admitidos, por ejemplo, diabetes descontrolada, asma, etc.), prescripción médica del doctor especificando el tratamiento para

el paciente, informe sobre su estado fisiológico inicial con la especificación de variables tales como temperatura, nivel de glucosa, tensión, etc. Todos estos parámetros constituyen el *modelado estático del paciente* y se generan en base a distribuciones usadas por defecto almacenadas en la base de datos del juego así como en el perfil del usuario inicialmente creado por el supervisor de la facultad y actualizado por el sistema para ir midiendo los logros del estudiante. El supervisor de la facultad también actualiza el perfil del usuario para establecer el ámbito de cobertura y los objetivos de aprendizaje. La información del perfil de usuario tiene preferencia sobre la información de la base de datos (por ejemplo, si la facultad excluye el asma para la clase completa, ningún paciente de asma se generará independientemente de su frecuencia en la base de datos).

3. *Informarse*: El estudiante lee el informe de cambios acerca de cada paciente confiado a él. El turno del estudiante comienza en el momento en que el estudiante revisa todos los pacientes.

4. *Turno en progreso*: Durante el turno, el estado de los pacientes evoluciona de tal forma que refleja su condición médica y su reacción a diferentes acontecimientos que tienen lugar (acontecimientos planificados como las comidas y dormir, intervenciones del enfermero estudiante que consisten en procedimientos médicos y medicamentos. Estos parámetros son modelados creando grafos como el de la figura (figura 238) basándose en las entradas de la facultad de enfermería. Luego se representan en MATLAB para encontrar la función que mejor encaje. Las funciones generadas son después programadas para generar la progresión de los parámetros. El paciente descrito es ideal en el sentido de que no hay incertidumbre, no hay sorpresas asociadas a su cuidado; constituyen el *modelado de paciente "ideal" dinámico*. Son casos importantes para el entrenamiento de estudiantes enfermeros ya que refuerza el conocimiento aprendido y la expectativa de que un gran porcentaje de los pacientes son casos de este tipo que reaccionan a tratamientos de una forma predecible.

   Sin embargo, la competencia consiste en ser capaz de manejar todas las excepciones y reaccionar correctamente y a tiempo a casos atípicos, en ser capaz de pensar con criterio, reconocer erorres y recuperarse de ellos. En VI-MED se han planificado tres tipos de "incertidumbres" denominadas habilidades críticas del estudiante (no todas implementadas en el entorno) y son las siguientes: a) variaciones en la reacción del paciente a tratamientos, b) introducción de error y ruidos en los datos de un paciente dado así como en las prescripciones y c) eventos no planificados que conllevan cambio en el estado del paciente (Mili et al. (2008)).

5. *Final de turno*: Al final del turno, el reloj se detiene y el estado de los pacientes deja de modificarse.

6. *Registro de salida*: El estudiante visualiza un resumen del turno y tiene la opción de crear este parte de turno en su registro permanente o eliminarlo. Si el estudiante crea este parte, el log de la sesión se añade a una base de datos. Estos informes en la base de datos se usan para actualizar el perfil del estudiante, generar retroalimentación para el estudiante, proporcionar resúmenes y valoraciones para la facultad.

El componente de evaluación es un sistema basado en web que accede y modifica la base de datos de los perfiles de los estudiantes y los logs de turnos de enfermería. Este subsistema tiene

una interfaz con el componente de juego (para recoger los logs y actualizar los perfiles de usuario), una interfaz para los estudiantes (que monitoriza su progreso generando diversos informes) y una interfaz para la facultad (para generar informes sobre sesiones individuales, estudiantes, clases o conceptos). Los profesores pueden también establecer parámetros de estudiantes o clases para controlar los objetivos de aprendizaje y la dificultad de los turnos de enfermería generados para sus estudiantes.

#### 2.4.5.1 Modelo del estudiante

El subsistema de valoración analiza la ejecución del estudiante y marca el ámbito y nivel de dificultad de los turnos generados. Este análisis se usa para generar informes con diferentes niveles de detalle para el estudiante y la facultad. El análisis del log de datos del estudiante y su uso para marcar el aprendizaje del estudiante se muestra a continuación:

- *Recogida y análisis de la ejecución del estudiante*: Por cada sesión de juego en la que el estudiante se registra, se almacena un log en la base de datos del estudiante. El log de turno registra una historia de 12 horas por cada paciente del turno incluyendo los parámetros clave cada hora y un log de cada evento que sucede con una gráfica antes y después de los parámetros relevantes.

  El log de datos es analizado desde tres puntos de vista distintos: análisis basado en los resultados, análisis basado en el proceso y opiniones críticas y estados de alerta.

  *Análisis basado en los resultados*: Los indicadores de salud de los pacientes al final del turno se comparan con su estado al comienzo del turno. El estudiante recibe una puntuación que es una suma ponderada normalizada de los cambios acumulados en las distancias entre sus valores y el valor medio ideal. La suma ponderada representa la importancia relativa de los diferentes parámetros. Para medir la severidad de cada parámetro VI-MED usa una fórmula concreta y, finalmente, la evolución global de un paciente se mide mediante la siguiente fórmula:

  $$e(paciente) = \sum_{X} w(X) * e(X)$$

  donde se calcula la suma ponderada de la evolución del paciente en todos los parámetros $X$ de relevancia. El grado (grade?) basado en los resultados del estudiante es el promedio de la evolución de todos los parámetros en sus pacientes.

  *Análisis basado en el proceso*: Un estudiante puede tener un grado de éxito basado en los resultados alto incluso aunque haya tenido un gran número de errores (por ejemplo, es una decisión equivocada cambiar las prescripciones entre pacientes incluso si los resultados no son no son perjudiciales). De forma similar, un estudiante puede hacer todo correcto pero ver a sus pacientes empeorar (por ejemplo, dar penicilina a un paciente sin conocer alergias es la decisión correcta incluso si el paciente acaba con una reacción alérgica). El análisis basado en el proceso compara las acciones del estudiante y carencias con las acciones correctas esperadas. La ejecución basada en el proceso del estudiante se mide mediante el número de errores normalizado:

  $$GradoProceso = \frac{número(decisiones) - número(errores)}{número(decisiones)}$$

  La realimentación que el estudiante obtiene sobre la calidad basada en el proceso de sus decisiones va más allá de la clasificación binaria de decisiones correctas e incorrectas ya

que requiere un examen más detallado de la incertidumbre presente durante la decisión y el análisis del riesgo subyacente. En la figura siguiente se muestra un árbol de decisión con las probabilidades asociadas a varios resultados.

figura 240

*Alarmas y opiniciones críticas*: se supone que los enfermeros no hacen más que ejecutar órdenes confiando en su experiencia y pensamiento crítico para detectar errores cuando suceden y corregirlos. Estos errores incluyen errores en prescripciones, en la lectura o simplemente eventos inesperados que afectan al paciente y requieren un cambio en las intervenciones médicas. Las alarmas y pensamientos críticos se miden mediante la frecuencia con que un estudiante de enfermería es capaz de medir situaciones que requieren una acción no prescrita por el doctor, identificar cuando ellos tienen tiempo para llamar al doctor, discutir el caso con ellos, y también tomar iniciativas y actuar inmediatamente cuando una situación lo exige. El pensamiento crítico puede medirse en el juego por el porcentaje de errores introducidos en el juego que fueron detectados y corregidos por el estudiante de enfermería:

$$GradoPensamientoCrítico = \frac{número(detectado)}{número(introducido)}$$

El pensamiento crítico es a menudo también asociado con alarmas y la posibilidad de operar bajo presión. Es mucho más fácil para un estudiante enfermero reconocer un error si tiene muy pocos pacientes y se comportan como se espera. De este modo, cuando alguien se sale de lo ordinario, el estudiante de enfermería puede prestarle su completa atención. El pensamiento crítico es también importante cuando el estudiante está operando bajo presión de tiempo (muchos pacientes que necesiten ser atendidos), presión intelectual (muchos pacientes están reaccionando fuera de la media (varianza alta en modelos de pacientes) o presión emocional (cuando pacientes y padres de pacientes realizando peticiones y distrayendo). Cuando el estudiante progresa, estos tres tipos de presiones aumentan en el juego para probar su pensamiento crítico y estado de alerta.

- *Personalización de la Experiencia de Aprendizaje*: El sistema de juego usa el perfil del estudiante para controlar el nivel de dificultad del turno de enfermería. El perfil especifica parámetros como los siguientes: *número de pacientes* asignados en el turno de enfermería (para nuevos estudiantes, este número comienza en 1 y va aumentando hasta 10 cuando el estudiante progresa satisfactoriamente), la *mezcla de enfermedades* generadas para el estudiante en el turno (este parámetro y la progresión puede ser establecido para seguir el curriculum usado en la clase), la *desviación de los pacientes de la norma* (en ausencia de un perfil de estudiante, los pacientes evolucionan usando un modelo dinámico con una distribución predefinida, generalmente normal. Para estudiantes nuevos, la desviación estándar usada será cero, creando así pacientes ideales sin incertidumbre en su comportamiento y sin sorpresas en reacciones a tratamientos médicos. Sin embargo, cuando el estudiante progresa, la desviación estándar usada será aumentada paulatinamente hasta que alcance la desviación estándar actual. El perfil contendrá desviaciones estándar que serán usadas en vez de las desviaciones estándar almacenadas en la base de datos), *la frecuencia de errores en los datos del laboratorio y en las órdenes del doctor* (esta característica puede ponerse a cero por defecto) y la *frecuencia de cambios en el estado del paciente* (también es puesta a cero por defecto y aumentada en el perfil del estudiante).

El sistema actualiza el perfil del estudiante basado en un conjunto de reglas que monitorizan las tres métricas de valoración para el estudiante y aumenta o disminuye el nivel de dificultad

adecuadamente.

Actualmente, el sistema está siendo probado y refinado. Además, se está trabajando en expandir su funcionalidad para soportar no sólo la aplicación de conocimiento adquirido sino también la adquisición de nuevo conocimiento por el estudiante.

### 2.4.6 LaSiTo

LaSiTo es un laboratorio virtual simulado de tornos (Noguez and Huesca (2008)). Un laboratorio virtual permite realizar experimentos con equipos que no están físicamente presentes en el mismo lugar que el usuario. Hay dos tipos de laboratorios virtuales: remoto y simulado. En el laboratorio remoto el usuario interactúa con el equipo que esta físicamente en otro lugar vía sensores y actuadores, usando un ordenador y una red de comunicación. En el laboratorio simulado, tal como LaSiTo, hay un modelo de ordenador del equipo y su entorno y los usuarios interactúan con estos modelos para realizar experimentos. Esto permite también una fase de entrenamiento segura en este entorno virtual a diferencia del entrenamiento en entornos reales en los que el uso del torno puede ser peligroso para el estudiante o puede dañar el equipo.

En la figura siguiente se muestra la arquitectura genérica para laboratorios virtuales soportada por LaSiTo.

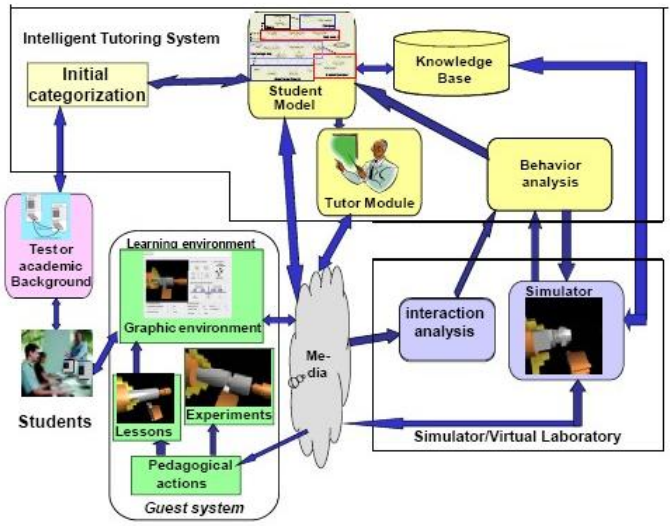


**Figura 2.43:** Arquitectura genérica para laboratorios virtuales. Extraído de Noguez and Huesca (2008)

*Learning environment*: en el entorno de aprendizaje se consideran aspectos de entornos de aprendizaje abiertos debido a que los estudiantes necesitan explorar diferentes parámetros para observar sus efectos dentro del laboratorio simulado. Sin embargo, cada experimento tiene objetivos específicos que guían el proceso de aprendizaje y que el estudiante necesita alcanzar para permitir una valoración efectiva del comportamiento en la exploración y de las metas de aprendizaje. A este nivel, se ha definido una interfaz hombre-ordenador basada en esta información (figura 2.44). Como se puede observar en la figura, la interfaz permite al estudiante observar su comportamiento durante el experimento en el simulador mediante, normalmente, una interfaz gráfica (1 en la figura), explorar y controlar el entorno simulado (2 y 3 respectivamente en la figura), ver los resultados durante un experimento; el comportamiento dinámico (área en la esquina inferior izquierda de la figura) y resultado final del experimento (5 en la figura).

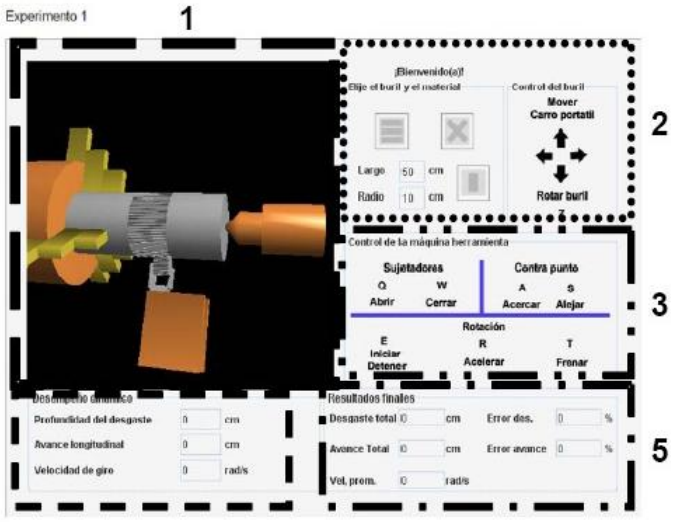


**Figura 2.44:** Entorno de aprendizaje para el laboratorio virtual LaSiTo. Extraído de Noguez and Huesca (2008)

*Simulator/Virtual Laboratory*: Es el módulo simulador que contiene un conjunto de experimentos según el dominio. En este laboratorio virtual, el estudiante puede interactuar con el equipo de torno que trabaja con metal a través de comandos directos (área 3 ) y visualizar el experimento (área 1 en la figura 2.44), tal y como se ha mencionado previamente. Para definir los experimentos, en LaSiTo se consideran algunas tareas básicas que el estudiante necesita hacer con el torno mecánico. De este modo, se define una secuencia de experimentos específicos (hasta el momento son 5 detallados en Noguez and Huesca (2008)), que permiten a los estudiantes adquirir el conocimiento y habilidades paso a paso para manejar sin peligro un torno.

*Intelligent Tutoring System*: En LaSiTo se ha asociado un SIT (figura 2.44) al laboratorio virtual de tal modo que, se extiende la libre exploración de las capacidades del laboratorio virtual con las labores de tutoría. El tutor sigue la exploración y la ejecución del estudiante en el laboratorio, actualiza su modelo, proporciona la ayuda adecuada si se requiere, infiere el estado del estudiante y decide la mejor acción pedagógica incluyendo la definición de los siguientes experimentos. Dada la incertidumbre inherente a esta tarea, en LaSiTo se ha optado por un modelo del estudiante, elemento principal de su arquitectura, que pueda representar y razonar con incertidumbre. Para ello, se usan redes bayesianas (Pearl, 1988) y modelos relacionales Probabilísticos (MRPs) (Koller, 1999). Cuando un estudiante realiza un experimento en el laboratorio virtual, el modelo del estudiante relacional (*Student Model*) propaga la evidencia desde la evaluación de los experimentos a los objetos de conocimiento en la base de conocimientos (*Knowledge Base*). Basado en esta evidencia y la evidencia acumulada de experimentos previos, el módulo de comportamiento y ejecución (*Behavior Analysis*) actualiza el modelo del estudiante. Después de cada experimento, los resultados se usan por el módulo tutor (*Tutor Module*) para decidir la mejor acción pedagógica.

#### 2.4.6.1 Student Model

La aplicación de los MRP's al modelo del estudiante en LaSiTo permite integrar el poder expresivo de las redes Bayesianas y las facilidades de los modelos relacionales (Noguez and Sucar

(2005a)). El dominio es así representado en términos de entidades con sus propiedades y relaciones. El modelado consta de los siguientes pasos:

1. *Identificar las clases del modelo*: las clases y sus relaciones proporcionan un esquema general para el modelo del estudiante.

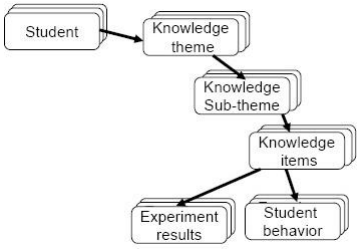


**Figura 2.45:** Esquema de MRP para el modelo del estudiante de laboratorios virtuales. Extraído de (Noguez and Huesca (2008))

2. *Obtener un esqueleto general*: para cada clase se define un numero de atributos (variables de información y variables aleatorias). Por ejemplo, la clase *Experiment results* está formada por atributos como *id, number of repetitions, success, efficiency, performance*, etc. Una vez especificado el modelo a nivel de clases, incluyendo los atributos y sus dependencias, se puede extraer un esqueleto, es decir, un modelo de red Bayesiana general para un fragmento del modelo. La figura 2.46 representa un esqueleto general obtenido del esquema de la Figura 2.45. Esta red representa las dependencias entre los objetos de conocimiento en diferentes niveles de granularidad y su relación con los resultados de los experimentos y el comportamiento del estudiante.

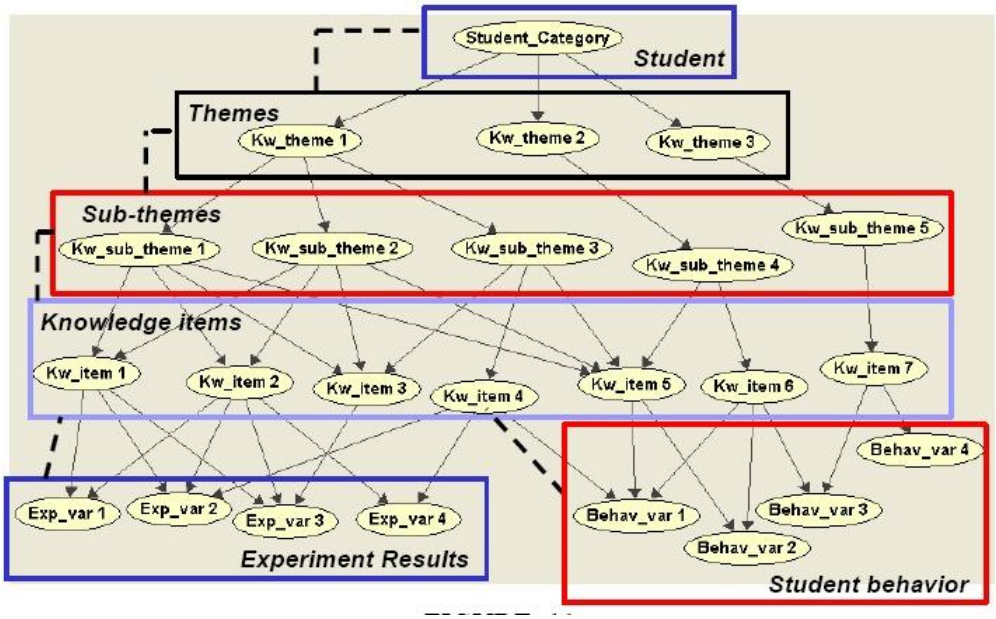


**Figura 2.46:** Esquema de MRP para el modelo del estudiante de laboratorios virtuales. Extraído de (Noguez and Huesca (2008))

Cada esqueleto general como el anterior, que especifica un modelo general para cualquier experimento, es instanciado posteriormente para un experimento particular de acuerdo a los valores de las variables específicas en el modelo. En la figura 2.47, se muestran cómo se pueden obtener varias instancias del MRP de la figura para inferir el conocimiento que el estudiante adquiere para diferentes experimentos, de acuerdo al experimento y el nivel del estudiante.

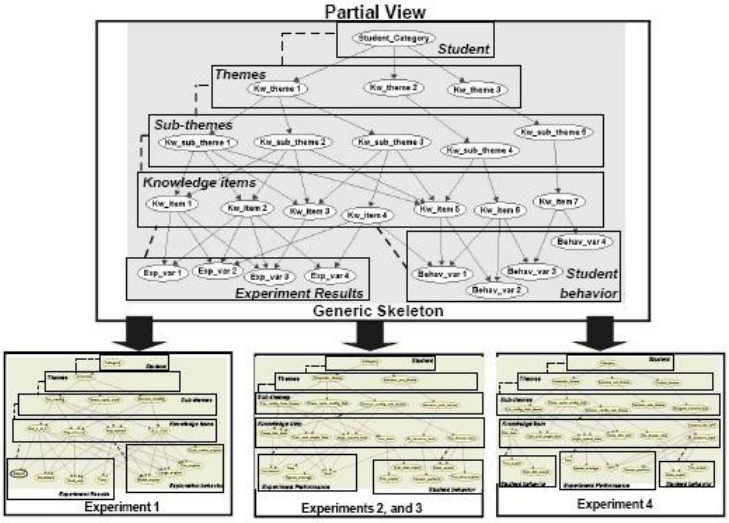


**Figura 2.47:** Instancias del ME obtenidas para distintos experimentos (1, 2 y 3, 4). Extraído de (Noguez and Huesca (2008))

3. *Obtener instancias específicas del modelo*: por ejemplo, se puede derivar la red Bayesiana particular de la figura s para el experimento 1 definido en el módulo *Simulator*, partiendo del esqueleto general de la figura 2.46. Así, la evidencia a partir de los resultados del experimento y el comportamiento en la exploración del estudiante se usan para inferir los elementos de conocimiento.

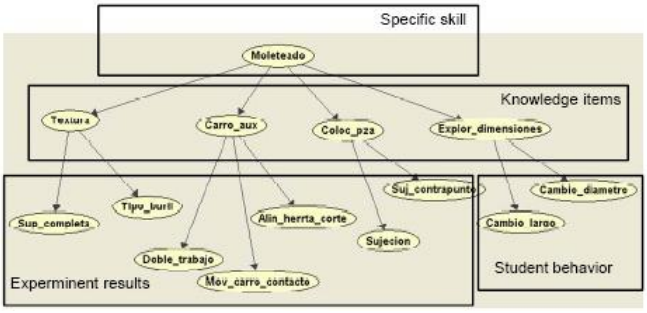


**Figura 2.48:** Fragmento de una instancia para un experimento específico (experimento1). Extraído de (Noguez and Huesca (2008))

Como en las redes Bayesianas, los parámetros del modelo consisten en una tabla de probabilidad condicional (TPC) para cada variable (atributo). Cada instancia de experimento tiene su propia TPC.

Actualmente, el modelo del estudiante relacional se ha integrado dentro de una base de datos y una herramienta de autoría que facilita la construcción del modelo para diferentes experimentos y dominios. El instructor o profesor puede especificar la exploración relevante y los parámetros de ejecución para cada elemento de conocimiento así como definir luego las relaciones entre elementos, subtemas y temas.

#### 2.4.6.2 Tutor Model

El módulo tutor mantiene la pista del conocimiento del estudiante en diferentes niveles de granularidad, combinando la ejecución y el comportamiento en la exploración en varios experimentos. De este modo, después de cada experimento, los resultados se usan por el módulo tutor para decidir la siguiente acción pedagógica mejor. Para ello, se define la utilidad de cada acción pedagógica mediante una función (Noguez and Sucar (2005b)). Si las metas del experimento están por debajo del valor esperado, el tutor aplica algunas reglas para decidir algunas acciones como visualizar ayuda o lecciones. Otras posibles acciones son las siguientes: presentar otro experimento con nivel de dificultad superior, presentar otro experimento con el mismo nivel de dificultad, aplicar un test o prueba o presentar otro experimento del tipo siguiente.

Para incrementar las posibilidades del entorno de aprendizaje y que interactúe de forma adecuada con el estudiante, siguiendo además con la filosofía no invasiva de los laboratorios virtuales, la aproximación planteada por los autores usa la información previa del estudiante como, por ejemplo, su formación académica y los datos de la interacción previa del estudiante. Una vez que esta información para cada estudiante es introducida en el sistema, el modelo relacional propaga la evidencia de los cursos aprobados para obtener una categorización inicial. Las categorías de estudiantes que usan específicamente en la aplicación son los siguientes: novato, medio y experto. Esta categoría se usa después para definir la complejidad de los ejercicios, las diferentes clases de ayuda que requiere el estudiante, etc.

El laboratorio virtual LaSiTo se ha evaluado con estudiantes de escuelas técnicas superiores satisfactoriamente. Es también importante señalar que la arquitectura y el entorno de aprendizaje semi-abierto se pueden aplicar a varios dominios. En concreto, fueron también previamente utilizados para construir laboratorios robóticos móviles, virtuales y remotos (Sucar et al. (2007)). Los resultados en este caso demostraron también que el tutor inteligente combinado así con el laboratorio virtual mejoran los conocimientos del estudiante y, en general, el proceso de aprendizaje.

Una limitación del modelo es que, actualmente, no incluye en el SIT comportamiento afectivo.

### 2.4.7 Modelo conceptual de Entornos de aprendizaje Virtual Personalizado de Xu et al.

Según los autores (Dongming Xu and Wang (2005)), los Entornos Virtuales de Aprendizaje (EVA) deberían ser capaces de identificar las necesidades de aprendizaje de los estudiantes y personalizar su formación, con o sin un instructor que complemente la enseñanza. Estos sistemas son llamados EVA personalizados -EVAP (Martinez y Bunderson, 2000).

Uno de los aspectos claves de esta propuesta, a diferencia de la mayoría de las aproximaciones examinadas en este trabajo, es la importancia que los autores exponen sobre el desarrollo de un completo modelo conceptual para EVAP, que facilite rápidamente la detección y corrección de errores en su desarrollo. Así, en (Dongming Xu and Wang (2005)) los autores presentan un modelo para EVAs basados en principios de aprendizaje objetivista y, especialmente, un modelo

conceptual para EVAPs, sistemas en los que se enfoca su trabajo. El modelo para estos sistemas puede verse, como argumentan los autores, como una aplicación de un modelo de aprendizaje constructivista. Esta característica también distingue esta aproximación de la mayoría de los EVA analizados, basados en un modelo de aprendizaje objetivista. En un modelo constructivista, los instructores son más bien intermediarios que proporcionan ayuda al estudiante durante el proceso de aprendizaje; el estudiante es que realmente debería controlar el proceso de aprendizaje, descubriendo elementos de conocimiento por sí mismo más que a través de la instrucción (Jonassen and Wilson (1993)). A diferencia de este modelo, en un modelo de aprendizaje objetivista (Leidner and Jarvenpaa (1995)), el agente de tutoría (o tutor en SITs) tiene como principal objetivo la transferencia de conocimiento al estudiante y la evaluación de su estado cognitivo usando para ello ejercicios, tests, etc.

La representación de los modelos presentados por los autores para EVAs se sustentan además en el modelado de diversas primitivas de conocimiento de estos sistemas. Dado el especial interés para este trabajo, se describen a continuación.

#### 2.4.7.1 Modelos Conceptuales de primitivas de conocimiento

El conocimiento en un EVA puede ser modelado en dos niveles: nivel de dominio relacionado con el dominio del mundo real, y el meta nivel relativo al conocimiento sobre el conocimiento del nivel de dominio.

- ***Modelo del nivel de dominio***. Su conocimiento contiene el curriculum del conocimiento de dominio, estudiantes, instructores, y la relación entre ellos. El modelo del curriculum es la estructura del curriculum. En la Figura 2.49 se muestra una parte de los modelos de los estudiantes e instructores.

  En función de los principios pedagógicos, el objetivo de aprendizaje (*Learning_Goal*) es, o bien definido por los instructores bajo el modelo de aprendizaje objetivista o bien, por los estudiantes bajo el modelo de aprendizaje constructivista.

  El modelo del estudiante está relacionado con *Learner_Profile*, *Learning_Goal*, y *Learning_Plan*. Esta última, tiene asociada la acción auto-valoración (*self-assessment*).

  La construcción de perfiles de estudiantes (*Learner_Profile*) dinámicos está basado en los patrones de comportamiento de los estudiantes y sus actividades de aprendizaje. Estos perfiles pueden ser utilizados para soportar aprendizaje colaborativo y para permitir formación personalizada a diferentes estudiantes.

  En la clase *Pedagogical_Model* se almacenan un número de estrategias educativas, la actitud docente (*teaching_attitude*) y habilidades (motora, intelectual, y en la resolución de problemas).

- ***Modelo del meta-nivel***. Las entidades del metanivel no se enfocan en la estructura del conocimiento básico, sino en la naturaleza de los contextos de aprendizaje de los estudiantes a través de los que pueden construir su propio conocimiento acerca de un dominio e interpretar sus propias experiencias. Una parte de este modelo se presenta en la Figura 2.50.

  El modelo *Situation* caracteriza un contexto abierto con el que estudiante interacciona. Como se puede observar, en esta entidad se definen atributos como *event*, que representa la situación actual; *pre_condition* representa un evento que debe suceder antes de la situación actual, y *pos_condition*, representa un evento que ocurrirá después de la situación actual.

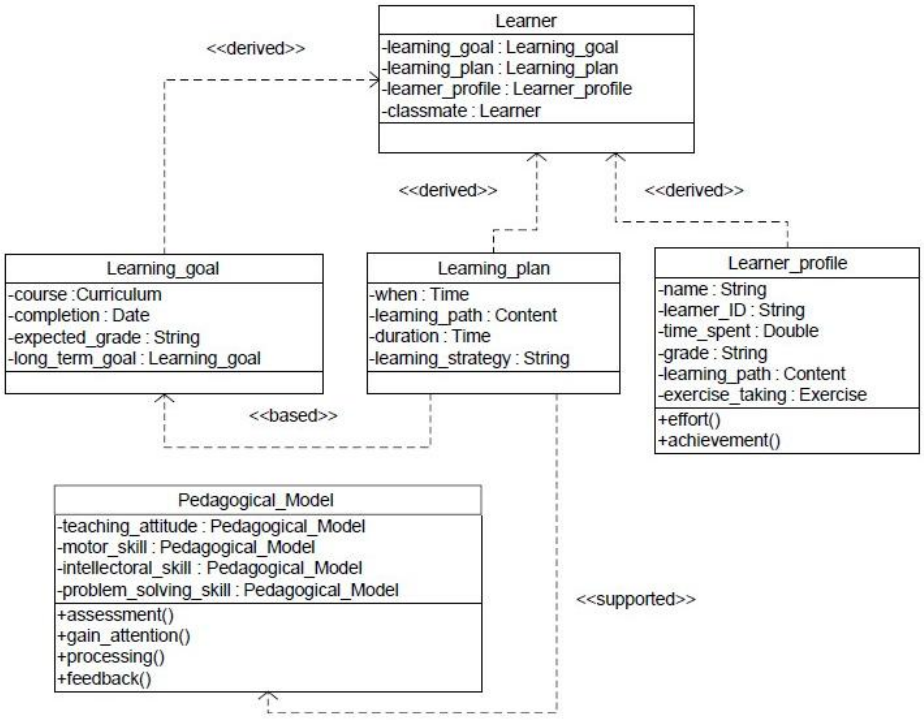


**Figura 2.49:** Diagrama de clases parcial del modelo del estudiante y el modelo pedagógico. Extraído de (Dongming Xu and Wang (2005))

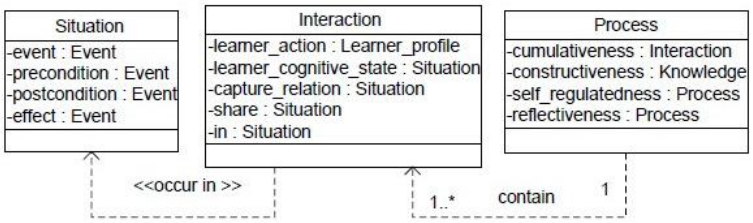


**Figura 2.50:** Diagrama parcial de clases para situación, interacción y proceso. Extraído de (Dongming Xu and Wang (2005))

El modelo *Interaction* está relacionado con la ocurrencia de eventos o las entidades que soportan situaciones y los estados cognitivos, actividades, y contextos. Las acciones del estudiante (*learner_action*) son las distintas formas de interacción (por ejemplo, utilizar, generar, o acceder a objetos). Los estados cognitivos del estudiante (*learner_cognitive_state*) permiten representar las diversas formas en que las entidades de una situación están relacionadas con la estructura cognitiva del estudiante formada previamente.

El modelo *Process* presenta cómo los patrones de interacción, en una situación, están conectados con los patrones de interacción de otra situación. Por ejemplo, *cumulativeness*

representa cómo el conocimiento aprendido en una situación puede usarse en otra situación, o *self-regulatedness*, que representa el proceso meta-cognitivo en el que aspectos de experiencias de aprendizaje previas son revisadas en una situación posterior.

#### 2.4.7.2 Modelo Conceptual de EVAPs

El modelo de aprendizaje constructivista se presenta en la Figura 2.51. Este modelo está basado en el sistema INCENSE (Akhras and Self (2000)).

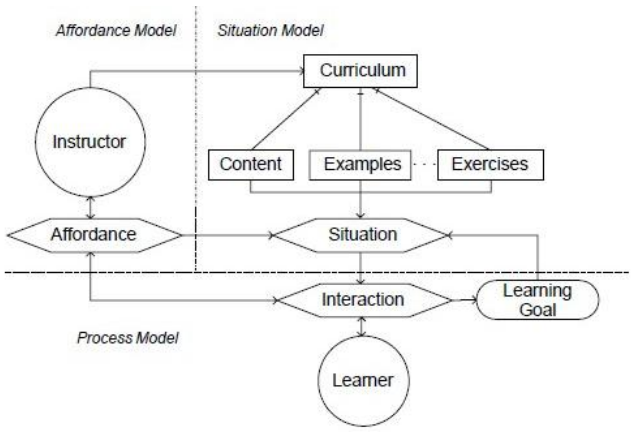


**Figura 2.51:** Modelo Conceptual de EVAs constructivistas. Extraído de (Dongming Xu and Wang (2005))

En base a los principios pedagógicos constructivistas, los EVAs se adaptan a las características del estudiante, el entorno, y la interacción entre el estudiante y el entorno. El aprendizaje desde este enfoque es un proceso interactivo entre el aprendizaje y el entorno (*learning by doing*).

En el diseño de los EVAs constructivistas, lo anterior implica resaltar que el dominio es modelado en términos de situaciones más que en la estructura de conocimiento, que la evaluación del aprendizaje se centra en el proceso de aprendizaje más que en el objetivo en sí mismo, y que las oportunidades de aprendizaje surgen a partir de situaciones alcanzadas más que a partir de situaciones proporcionadas en base a las estrategias de enseñanza (self 2002).

El modelo *Affordance* indica las posibilidades en determinadas situaciones para el desarrollo de actividades de aprendizaje relevantes para un estudiante cuyo proceso de aprendizaje está en un cierto estado. Es la base para crear espacios de interacción para estudiantes.

#### 2.4.7.3 Ontología de EVAPs

Como ya se ha mencionado, los EVAPs están basado en un enfoque pedagógico constructivista, que se adapta a las características del estudiante, el entorno, y la interacción entre el estudiante y el entorno. Para representar el modelo conceptual de estos sistemas, y basándose en los modelos previos, los autores desarrollaron la siguiente ontología (Figura 2.53) utilizando como método de modelado, Tropos (Bresciani et al. (2004)). De este modo, se puede modelar el mundo (agentes/actores -estudiantes, instructores, etc., sus obligaciones y capacidades, intenciones de los agentes, comunicaciones y diálogos entre agentes, procesos de aprendizaje y sus relaciones, etc.). Los estereotipos de Tropos se muestran en la figura 2.52.

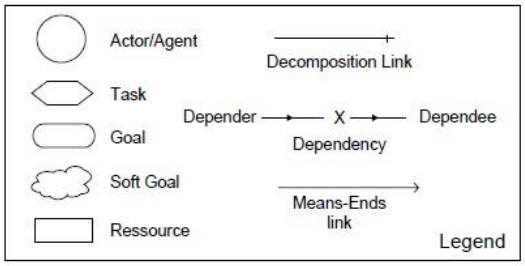


**Figura 2.52:** Estereotipos de *Tropos*. Extraído de (Dongming Xu and Wang (2005))

Usando el método Tropos, cada categoría de conocimiento en un EVAP se trata como una clase (con sus instancias). Todas las clases de información pueden organizarse dentro de una jerarquía. De este modo, la jerarquía de conocimiento se puede dividir en nivel de conocimiento de dominio y nivel de meta-nivel, tal y como se vio anteriormente.

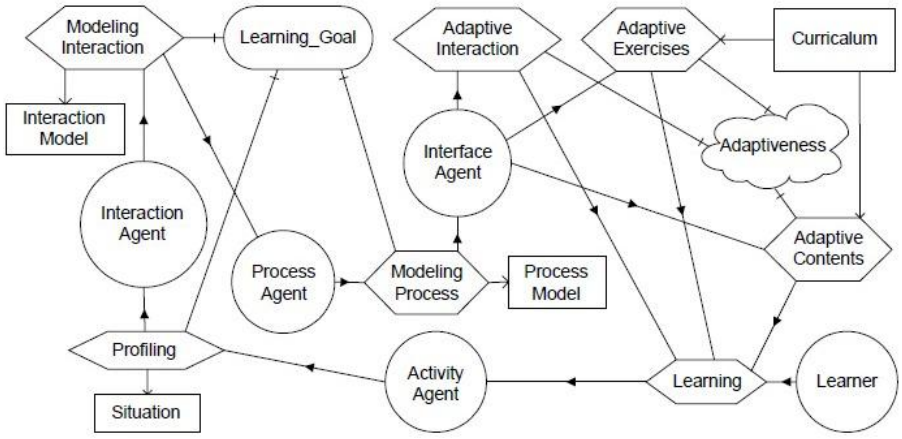


**Figura 2.53:** Ontología de EVAPs. Extraído de (Dongming Xu and Wang (2005))

En la ontología propuesta para EVAPs (Figura 2.53), el *Curriculum* almacena la información del curriculum (contenidos, ejercicios, ejemplos, y las relaciones entre ellos); el *Learner_Profile* almacena la historia del aprendizaje de cada estudiante. Cuando el alumno está estudiando, sus actividades de aprendizaje (camino de aprendizaje, tiempo de aprendizaje, ritmo de aprendizaje, y logros de aprendizaje) son monitorizadas y analizadas por el *Activity Agent* y los datos del perfil son almacenados en el *Learner_Profile*.

Basado en la situación y en el *Learner_Profile*, el *Interaction agent* modela las interacciones y construye el modelo *Interaction*. El *Process agent* evaluará entonces este modelo de interacción y construirá un nuevo modelo de proceso. La aparición de situaciones de aprendizaje, el modelo de interacción y el modelo de proceso, provoca que el *Interface Agent* proporcione interacción, contenidos y ejercicios personalizados al estudiante.

Basado en los modelos conceptuales anteriores, los autores han implementado un prototipo de EVAPs basado en multi-agentes cuya arquitectura se muestra en la Figura 2.54.

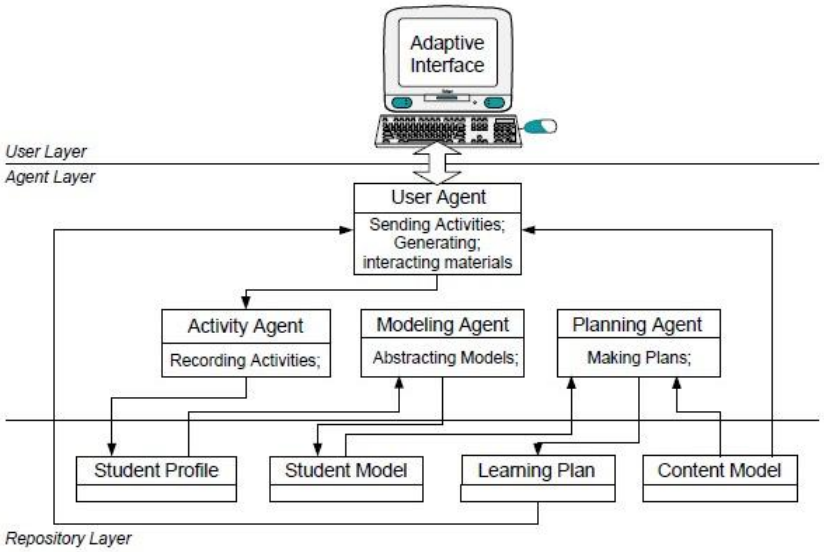


**Figura 2.54:** Arquitectura del prototipo de EVAP de Xu et al. Extraído de (Dongming Xu and Wang (2005))

Como se puede observar, hay tres niveles en la arquitectura. El nivel *Repository*, que contiene recursos tales como el Curriculum (Modelo de Contenido), el perfil del estudiante, el modelo del estudiante y el plan de aprendizaje así como las relaciones entre estas entidades. Basado en el modelo conceptual previo, e implementados los agentes correspondientes (ver Figura 2.54), el proceso de aprendizaje general es el siguiente: el agente *Activity* registra las actividades de interacción del estudiante en el proceso de aprendizaje y genera el *Student Profile* del estudiante. Basado en este perfil del estudiante, el agente *Modeling* revisa el modelo del estudiante en una cierta secuencia de tiempo dentro del modelo de proceso. La modificación en el *Student Model* provoca la invocación al agente *Planning* para revisar el plan de aprendizaje. Este plan indica las posibilidades de situaciones para el desarrollo de actividad de aprendizaje relevante. La personalización del aprendizaje del EVAP se realiza en tres niveles: *aprendizaje* (se proporciona al estudiante diferentes tipos de materiales de aprendizaje en función del modelo del estudiante, el plan de aprendizaje, y el modelo de instrucción), *auto-evaluación* (después del aprendizaje de una sección, se aconseja al estudiante hacer ejercicios. Se generan las preguntas dinámicamente basándose en el modelo de contenido, modelo del estudiante actual y el modelo de instrucción), y *ajuste del sistema* (basado en el análisis anterior, el EVAP realizará la modificación del perfil del estudiante y el plan de aprendizaje actual basándose en el modelo de instrucción. Comienza entonces de nuevo, teniendo en cuenta las modificaciones previas).

### 2.4.8 MAEVIF

El modelado del estudiante que se propone en la presente tesis, ha sido implementado e integrado en una plataforma software para el desarrollo de EVIEs llamada MAEVIF. Esta plataforma se desarrolla por el grupo de investigación de Laboratorio Decoroso Crespo de la Universidad Politécnica de Madrid (de Antonio et al. (2005), Imbert et al. (2007)) desde el 2001. En el año 2006 el proyecto MAEVIF se integra en un proyecto en el que participan más universidades

denominado ENVIRA. Actualmente, se están realizando muchas variaciones tanto en la arquitectura del sistema como en muchos de sus componentes para conseguir un producto de mejor calidad.

MAEVIF fue diseñada para ser fácilmente configurable en diferentes aplicaciones de aprendizaje y su arquitectura está basada en agentes.

#### 2.4.8.1 Arquitectura de MAEVIF

MAEVIF se compone de de dos partes: el Entorno Virtual y el Sistema Inteligente de Tutoría.

El Entorno Virtual (EV) está formado por el conjunto de subsistemas gráficos cuya misión es permitir a los estudiantes interactuar con el escenario a través de dispositivos (ratón, teclado, cascos, guantes, etc.). Habrá tantos subsistemas gráficos en ejecución como alumnos conectados al sistema.

El SIT observa las actividades y movimientos de los estudiantes y los orienta a la consecución de un objetivo mediante mensajes que indican al estudiante si va bien o mal, sugiriendo soluciones o contestando a alguna pregunta del estudiante. Sólo tiene una instancia en MAEVIF.

Para su desarrollo, se identificaron uno o mas agentes para cada módulo de la arquitectura clásica de SITs (sección 1.1.1). Adicionalmente, se requirieron otros agentes para satisfacer las siguientes necesidades (Méndez and de Antonio (2005)):

- El EVIE puede usarse para entrenamiento de equipos, y ello exige que se adapte a las necesidades colectivas, no sólo a las individuales. También debe tener en cuenta tanto el conocimiento colectivo como el individual.
- En el EVIE los estudiantes están inmersos, teniendo cada uno de ellos su propia visión del entorno dependiendo de su posición y orientación.
- En un SIT el módulo de comunicación es realizado por medio de GUI o interfaces de lenguaje natural (siendo éste un medio para interactuar con el sistema), mientras que en un EVIF es un modelo gráfico tridimensional (cuyo conocimiento es parte también del entrenamiento). Por tanto hay que añadir al sistema conocimiento explícito del entorno tridimensional, su estado y las posibilidades de interacción con él.

Los agentes de la arquitectura de MAEVIF pueden verse en la figura 2.55.

- Agente de Comunicación Global (*Global Communication Agent* en la figura 2.55). Su misión sigue siendo la comunicación del sistema de agentes con el entorno virtual. Trata los mensajes de conexiones y desconexiones de los estudiantes del sistema.
- Agente de Comunicación (*Student Communication Agent* en la figura 2.55). Es la interfaz comunicativa entre los estudiantes y el sistema de tutoría. Cada estudiante tiene un Agente de Comunicación asociado.
- Agente Estudiante (*Student Modeling Agent* en la figura 2.55). Se encarga de guardar todas las acciones realizadas por el estudiante en el sistema, sus evaluaciones, y otros tipos de conocimientos sobre el estudiante, en términos de una ontología denominada *Ontología de Modelado del Estudiante*. Dicha información es utilizada por el Agente de Tutoría para evaluar, guiar, felicitar al estudiante, etc. Hay tantos agentes estudiantes como estudiantes haya conectados al sistema.

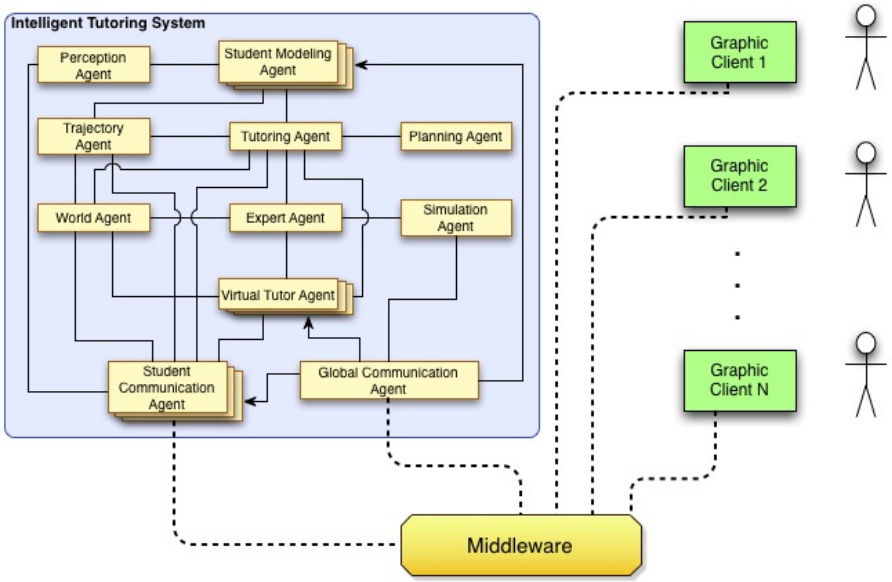


**Figura 2.55:** Arquitectura basada en agentes de MAEVIF

Puesto que el modelado del estudiante integrado en MAEVIF ha sido desarrollado en el ámbito de esta tesis, sus características y sus dos elementos esenciales, modelo del estudiante y mecanismo de diagnóstico cognitivo, son descritos en detalle más adelante (capítulo 5).

- Agente de Tutoría (*Tutoring Agent* en la figura 2.55). Su objetivo es proponer actividades a los estudiantes según el conocimiento que posean de la materia. Se encarga también de la formación de equipos para las actividades colectivas, y además asigna a cada estudiante un rol a desempeñar en cada actividad. Otra misión del agente de tutoría es el seguimiento del estudiante, comparando sus acciones con la solución obtenida por el agente de planificación. El agente de tutoría también puede responder a preguntas, relacionadas con la actividad, planteadas por el estudiante.

- Agente de Planificación (*Planning Agent* en la figura 2.55). Proporciona al Agente de Tutoría la solución a los ejercicios (actividades) que debe realizar el estudiante. Puesto que en este tipo de entornos frecuentemente se lleva a cabo el entrenamiento en procedimientos/tareas, la solución al ejercicio (actividad) consistirá en el plan o secuencia de acciones que el estudiante debe realizar en el EV para concluir la actividad o tarea satisfactoriamente. Una acción se define como la aplicación de un operador, y un operador se define con un conjunto de precondiciones y consecuencias.

  El Agente de Planificación permite la construcción dinámica de planes solución a partir del estado actual del entorno de aprendizaje y las acciones posibles del estudiante. Además, si en algún momento el estudiante se aparta del plan obtenido para el ejercicio inicialmente, el Agente de Planificación puede proporcionar la re-planificación de un plan solución.

- Agente de Trayectoria (*Trajectory Agent* en la figura 2.55). Calcula las trayectorias óptimas

y realiza el seguimiento del estudiante. El agente de tutoría revisa el conjunto de acciones posibles (generado por el agente de planificación), y si detecta que hay una acción de movimiento o transporte, llama al agente de trayectoria con la lista de destinos que puede alcanzar el estudiante. A continuación, el agente de trayectoria calcula la trayectoria óptima que debería seguir el estudiante, y usa dicha trayectoria para compararla con el itinerario real seguido por el estudiante.

- Agente Experto (*Expert Agent* en la figura 2.55). Recoge la información relacionada con las acciones que los estudiantes pueden realizar en el EV. En concreto, las precondiciones y consecuencias de las acciones. Esta información se usa para la validación y ejecución de las acciones del estudiante.

- Agente Mundo (*World Agent* en la figura 2.55). Mantiene y controla la información geométrica y semántica acerca de los objetos y habitantes del EV (estudiantes, personajes auxiliares y tutores virtuales). Esta información es representada en una ontología. Además aporta información al Agente Experto, para que éste pueda realizar las validaciones de acciones. Por último, también se encarga de gestionar la información sobre las posibles interacciones en el entorno virtual y sus consecuencias.

- Agente Percepción (*Perception Agent* en la figura 2.55). Proporciona los personajes virtuales (tutor virtual y otros personajes virtuales autónomos auxiliares) con capacidades perceptuales. Este agente monitoriza el campo visual para cada personaje, proporcionándole información sobre los lugares, objetos y personajes que ellos están observando. Esta información que servirá al agente estudiante y al agente de tutoría para saber si el estudiante está despistado o no sabe como continuar con la tarea que está realizando.

- El Agente de Simulación (*Simulation Agent* en la figura 2.55). Simula las consecuencias de cada acción del estudiante. También, este agente proporciona al estudiante información apropiada en respuesta a preguntas como "Qué sucedería si ...?". Otra tarea de este agente es proporcionar soporte al Agente Experto en el proceso de validación de las acciones.

- Agente Tutor Virtual (*Virtual Tutor Agent* en la figura 2.55). Controla la representación gráfica del tutor del estudiante en el entorno virtual. Se decidió que sería mejor tener un tutor virtual personalizado para cada estudiante para conseguir una forma de enseñanza más individualizada; por lo tanto, hay tantos agentes como estudiantes conectados al sistema.

#### 2.4.8.2 Funcionamiento básico de MAEVIF en una sesión de aprendizaje

Una sesión de aprendizaje se asume que comienza cuando un estudiante se conecta al EVIE a través de su interfaz gráfica 3D (ver Figura 2.55). El *Agente de Tutoría* recibe la petición de los estudiantes para iniciar la sesión de aprendizaje y debe decidir qué ejercicios o actividades están disponibles para este estudiante en base a, esencialmente, el perfil del estudiante, su comportamiento pasado en el curso, y el diseño educativo del curso. La información de un estudiante manejada por el EVIE a lo largo de diferentes sesiones de aprendizaje es almacenada y gestionada por otro agente denominado *Agente de Modelado del Estudiante*. Por lo tanto, cada vez que el *Agente de Tutoría* necesita información acerca de un estudiante, la solicitará al *Agente de Modelado del Estudiante*. Por ejemplo, al comienzo de la sesión de aprendizaje, el *Agente de Tutoría* necesita saber qué objetivos de aprendizaje han sido alcanzados o no por el estudiante, y esta información debe ser proporcionada por el *Agente de Modelado del Estudiante*.

Una vez que el estudiante ha elegido uno de los ejercicios (actividades) disponibles, el *Agente de Tutoría* necesita saber la forma correcta de resolverlo. En MAEVIF, esta información es facilitada al *Agente de Tutoría* por el *Agente de Planificación*. Dado que en este tipo de entornos frecuentemente se dirige el entrenamiento a procedimientos/tareas, la solución al ejercicio consistirá en el plan o secuencia de acciones que el estudiante debe realizar en el EV para finalizar la tarea satisfactoriamente. Una acción se define como la aplicación de un operador, y un operador se define como un conjunto de precondiciones y consecuencias. El *Agente de Planificación* permite permite la construcción dinámica de los planes solución teniendo en cuenta el estado actual del entorno de aprendizaje y las posibles acciones del estudiante. Sin embargo, si en algún momento el estudiante se separa del plan obtenido para el ejercicio inicialmente, el *Agente de Planificación* puede proporcionar la re-planificación de un plan solución.

Dentro del EV, el estudiante puede navegar de una posición a otra, y realizar acciones como coger objetos o pulsar botones. Todas estas acciones serán enviadas por el *Agente de Comunicación del Estudiante* al *Agente de Tutoría* y registradas por el *Agente de Modelado del Estudiante* en forma de traza de la actividad del estudiante. Además, cada vez que el estudiante ejecuta una acción (o al menos intenta hacerlo) en el EV, el *Agente de Modelado del Estudiante* debe actualizar sus creencias sobre el estado de los conocimientos del estudiante, de tal modo que pueda informar al *Agente de Tutoría* de los objetivos de aprendizaje que él cree que el estudiante ya ha alcanzado o no. Por ejemplo, supóngase que el estudiante tiene que llevar a cabo un experimento en un EV que representa un laboratorio de química, y tiene que coger una probeta de un estante donde están también matraces y vasos de precipitado. Si el estudiante realmente coge la probeta, entonces el *Agente de Modelado del Estudiante* puede concluir que el estudante sabe reconocer una probeta entre otros instrumentos de laboratorio. Similarmente, si el estudiante intenta añadir ácido sulfúrico al agua de un vaso de precipitado y no se ha puesto guantes, entonces el *Agente de Modelado del Estudiante* puede inferir que el estudiante no sabe que es obligatorio ponerse un par de guantes cuando se trabaja con ácido sulfúrico.

Una de las posibles acciones debe ser navegar de una posición a otra en el EV. En este caso, el estudiante debería seguir una trayectoria razonablemente buena. Por lo tanto, la calidad de la trayectorias seguidas por el estudiante destaca como otra importante cuestión que el *Agente de Modelado del Estudiante* debe tener en cuenta durante la sesión de aprendizaje. Una mala trayectoria puede revelar que el estudiante ha construido un mapa cognitivo equivocado de algún área en el EV, que en una situación de emergencia en el mundo real puede ser peligroso. El *Agente de Modelado del Estudiante* está encargado de registrar las trayectorias del estudiante y, como parte del diagnóstico, valorar su calidad teniendo en cuenta las características de la trayectoria óptima calculada por el *Agente de Trayectoria*.

El *Agente de Tutoría* debe decidir cuándo termina la actividad de aprendizaje. En el mejor caso, el estudiante realiza el plan asociado al ejercicio (u otro equivalente) con éxito, o el Agente de Tutoría puede decidir que la sesion debe ser abandonada porque el estudiante ha cometido demasiados errores y ha llegado a un callejón sin salida. A continuación, según la ejecución del estudiante, el *Agente de Tutoría* seleccionará la siguiente actividad de aprendizaje. Por ejemplo, si el estudiante completó el ejercicio sin ningún error, se le puede plantear otro ejercicio más complejo. Por el contrario, si el estudiante demostró algunas lagunas de conocimiento, el *Agente de Tutoría* puede redirigirle a lecciones previas relacionadas con esas lagunas.

### 2.4.9 HERA

HERA (*H*elpful agent for saf*E*ty lea*R*ning in virtu*A*l environment) es un EV para entrenamiento en lugares de alto riesgo propuesto por Amokrane, K. et al. (Amokrane et al. (2008b), Amokrane et al. (2008a)).

En HERA las actividades son representadas basándose en las descripciones de los procedimientos necesitados. Estos procedimientos son proporcionados por los expertos en entrevistas. Sin embargo, a diferencia de otros SITs como STEVE, PEGASE, etc., HERA también incorpora las discrepancias y errores proporcionados a través de entrevistas con los operadores in situ. De este modo, el sistema describe cómo las actividades deberían ser realizadas en situaciones normales y en situaciones con averías. Todo ello se realiza utilizando un lenguaje de descripción de actividades denominado HAWAI-DL (*H*uman *A*ctivity and *W*ork *A*nalysis for s*I*mulation-*D*escription *L*anguage, Amokrane et al. (2008c)) que proporciona una descripción jerárquica de tareas con las características previas. Mediante este lenguaje, HERA sabe acerca de la tarea que el estudiante debe realizar a continuación y todas sus posibles formas de realizarla mientras que otros sistemas, como STEVE, no saben nada acerca del plan que el estudiante debe seguir, a menos que el sistema reciba una pregunta sobre el plan (por ejemplo, *¿Cuál es la siguiente tarea?*).

HERA soporta la simulación y almacenamiento de eventos para reconstruir lo que sucedió en el curso de la acción realizada por el estudiante. Para ello, uno de los componentes de que consta es el *World Model*, que representa los objetos del entorno y sus acciones asociadas mediante una ontología específica integrada en un módulo de razonamiento denominado COLOMBO (Ontological Creation Linked to the Organization of the Objects Modeling). La arquitectura global de HERA se muestra en la Figura 2.56.

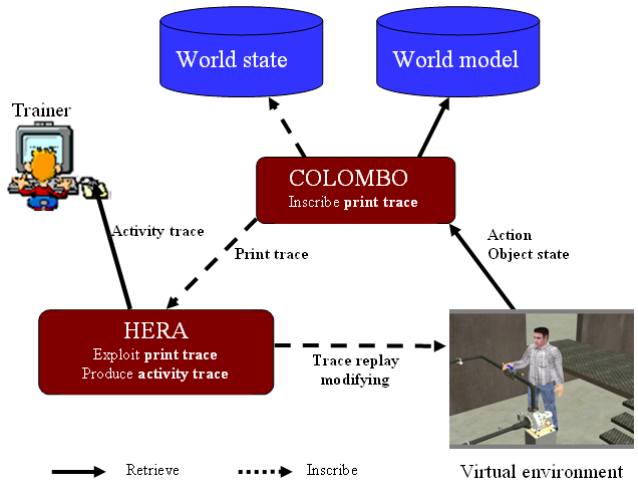


**Figura 2.56:** Arquitectura general de HERA

En HERA se distinguen dos tipos de trazas:

- ***Print trace***. Son los cambios en los estados del entorno y en los estados de sus objetos causados por el estudiante en la realización de las tareas asignadas. Son trazas *explotadas* por HERA.

- ***Activity trace***. Es la secuencia de todas las tareas (complejas y simples) que un estudiante ha llevado cabo durante el entrenamiento (incluye tareas, errores cometidos y riesgos causados por las acciones/decisiones del estudiante). Son trazas *producidas* por HERA.

COLOMBO inicializa y actualiza el estado del mundo (estados instantáneos de todos los objetos en el EV) para cada cambio en el EV. Además, registra y transmite las *print trace*. Como se puede observar en la Figura 2.56, para seguir la pista del estudiante, HERA intercambia mensajes de diferentes tipos con COLOMBO (mensajes de comienzo y finalización de acción, o mensajes de cambio de estado de un objeto del EV).

#### 2.4.9.1 Módulos de HERA

En HERA se distinguen cinco módulos (Figura 2.57):

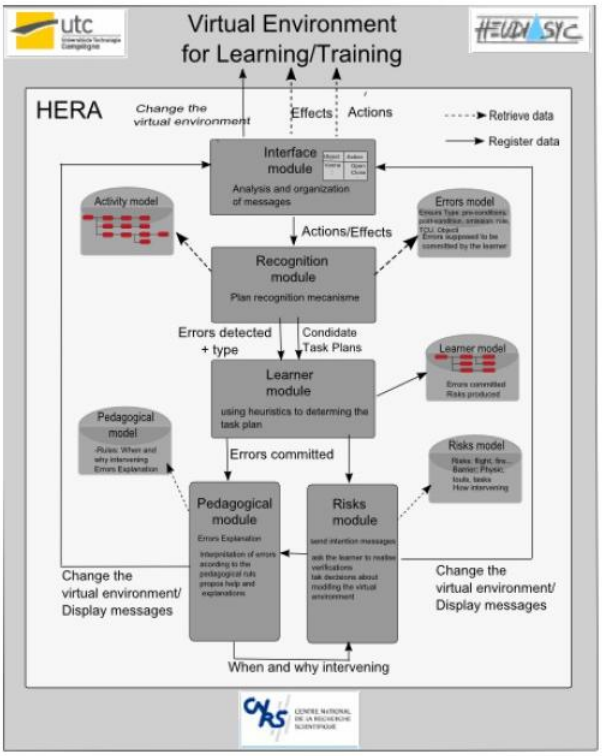


**Figura 2.57:** Módulos de HERA

- *Módulo de interfaz*. Actúa como un intermediario entre el EV y los otros módulos. Transmite mensajes desde otros módulos al EV y viceversa.

- *Módulo de reconocimiento*. Determina lo que se supone que el estudiante está haciendo, de acuerdo a lo observable (acciones y efectos), a la traza *print trace*, y al modelo de actividad. Esta tarea se basa en la técnica de reconocimiento de planes de El-Kechaï (El-Kechaï and Després (2006)). El algoritmo de reconocimiento usado determina la traza de actividad (la actividad actual del estudiante), los errores cometidos y los riesgos producidos. Las líneas generales de este algoritmo se describen en Amokrane et al. (2008a).

El modelo de actividad contiene una descripción detallada de la actividad que el estudiante debe realizar durante el entrenamiento. Para lograr este objetivo, se ha desarrollado, como ya se mencionó previamente, un lenguaje de descripción de actividad llamado HAWAI-DL.

Para gestionar los errores del estudiante, el SIT de HERA incluye un modelo de errores que contiene, según los autores, una clasificación genérica de los tipos de errores que pueden ser cometidos por los estudiantes, pero no de los errores actuales cometidos en una situación dada. Esta clasificación consiste, en primer lugar, en los errores clasificados por Hollnagel (Hollnagel (1993)). A estos errores, los autores han añadido los siguientes tipos de errores:

- *Errores relacionados con la tarea*. En este tipo se distinguen los siguientes subtipos de errores: a) *Errores de pre-condición*. Cuando una tarea llega a estar activa mientras sus precondiciones no se han cumplido aún. En este tipo de errores se distinguen diferentes clases de errores de pre-condición de acuerdo a diversos tipos de pre-condiciones. b) *Errores de post-condición*. Por ejemplo, una tarea considerada acabada cuando sus post-condiciones no se han cumplido aún o son chequeadas antes de la tarea haya acabado. c) *Errores de constructor*. Una tarea compleja detecta un error si su constructor es SEQ (secuencia de sub-tareas) y el orden de ejecución de sus sub-tareas no se respeta.
- Errores relacionados con los objetos meta. Cuando el estudiante manipula un objeto que no es el correcto.
- Errores en el *role*. Dependen del *role* del estudiante al desempeñar una tarea en un entorno colaborativo. Por ejemplo, cuando el estudiante realiza la tarea de otra persona y la tarea no corresponde con el *role* del estudiante, o cuando falla al realizar una tarea relacionada con su propio role.
- *Errores subjetivos*. Cuando una tarea no es realizada a causa de las limitaciones procedentes del punto de vista subjetivo del estudiante. Por ejemplo, cuando se provoca un fuego, y el estudiante no realiza la tarea de çontrolar el fuego"porque el fuego está fuera del campo de visión del estudiante y no es detectado.
- *Errores BTCU* (*B*orderline *T*olerated *C*onditions of *U*se). Estos errores ocurren cuando una tarea relacionada con la seguridad no es llevada a cabo por el estudiante. Por ejemplo, en la actividad de sustituir una tubería, un estudiante puede fallar al realizar la tarea "vaciar la tubería".

El modelo de errores también incluye los errores comunes existentes en el modelo de actividad. Por ejemplo, hay dos formas de abrir una puerta: .ªbrir progresivamenteconsiderada correcta, o .ªbrir repentinamenteerror a menudo cometido por los estudiantes y, por lo tanto, incluido en el módulo de errores.

- *Módulo del estudiante*. Produce la traza de actividad y permite determinar el plan de tareas realizado por el estudiante de entre todos los planes de tareas candidatos proporcionados por el módulo de reconocimiento. Este módulo utiliza heurísticas para determinar el plan de tareas actuales del estudiante si hay conflictos entre los planes de tareas candidatas. Las heurísticas elegidas son adaptaciones de aquéllas definidas por El-Kechaï (El-Kechaï (2007)).

- *Módulo de riesgos.* Determina los riesgos resultantes a partir de los errores del estudiante y sus consecuencias, y calcula la probabilidad de estos riesgos. Si la probabilidad es alta, dispara un riesgo on-line en el EV; en caso contrario, envía un mensaje al módulo pedagógico para tomar una decisión acerca de esta situación. El módulo de riesgos propone barreras (medidas) para eliminarlos, y modifica el EV de acuerdo a un conjunto de reglas que contiene el modelo de riesgos.

- *Módulo pedagógico.* El módulo del estudiante y el módulo de riesgo pueden transmitir errores y mensajes de riesgo respectivamente al módulo pedagógico. Posteriormente, este módulo evalúa si interviene o no de acuerdo al conjunto de reglas del modelo pedagógico. Este modelo contiene todas las reglas de entrenamiento (estrategias) -cuando y porqué interviene. La instrucción puede ser directamente indicar el error y dar la respuesta correcta, o indirectamente proporcionar pistas para conducir al estudiante a la respuesta correcta. En caso de errores con riesgos asociados, el módulo pedagógico coopera con el de riesgo para decidir si interviene para ayudar al estudiante, o para explicarle porqué hay un error (listando las causas del error). Dependiendo de los objetivos pedagógicos y las reglas pedagógicas, este módulo decide si generar o no un incidente. Por ejemplo, si asumimos que la meta pedagógica es enseñar al estudiante cómo manejar situaciones inesperadas, como por ejemplo una fuga, el estudiante no realiza una tarea de control de la fuga, y el tipo de error es una omisión (como ver el fenómeno sin reaccionar), este módulo envía un mensaje al módulo de riesgos para intervenir cambiando el EV (disparar un incidente apropiado).

Es importante señalar que HERA usa también la traza de la actividad de una forma off-line para ayudar al entrenador y al estudiante en la comprensión de la trayectoria de las acciones del estudiante. Asimismo, a diferencia de otros sistemas (STEVE, HAL, etc.), permite al estudiante realizar una tarea sin proponer ayudas o explicaciones on-line. Además, las situaciones de riesgo pueden ser on-line u off-line, de acuerdo a las reglas del módulo pedagógico y la gravedad del riesgo. Al repetir la traza del estudiante, el sistema muestra y explica al estudiante la tarea emprendida, las ocurrencias de las situaciones de error, los riesgos potenciales, y sus causas. Estas decisiones son tomadas por el módulo pedagógico y el módulo de riesgos. Las trazas pueden además mostrarse al estudiante de diferentes formas (como anotaciones, en la pantalla de entrenamiento, etc.). También se puede modificar el propio EV, dependiendo de las órdenes enviadas por el módulo pedagógico o el de riesgos, para permitir al estudiante ver el impacto de sus decisiones (por ejemplo, la aparición de un fuego cuando un estudiante no detiene una fuga provocada por la combustión de una sustancia inflamable en contacto con el aire).

El entorno ha sido desarrollado usando VIRTOOLS. Los estudiantes son representados mediante un avatar en el EV, y realizan las tareas asignadas en colaboración con otros miembros del equipo representados mediante personajes virtuales. El entrenamiento consiste en realizar tareas frente a una pantalla, e interactuar con el entorno usando dispositivos sencillos (ratón, teclado). El escenario asigna a los estudiantes el trabajo de cambiar una tubería y otras tareas dentro del contexto de un lugar de alto riesgo.

Los autores planean expandir las reglas pedagógicas así como el modelo de riesgos con reglas más genéricas, así como usar redes Bayesianas para calcular la probabilidad de un riesgo teniendo en cuenta la existencia y cambios de otros riesgos y errores generados por el estudiante y por otros personajes virtuales.

## 2.5 Análisis crítico de la situación

Las aportaciones en el campo de los SITs y, en concreto, en su modelado del estudiante han sido muy numerosas y en constante evolución. En los apartados anteriores se ha presentado una selección de las propuestas que, a juicio de la autora, han tenido mayor interés y han sido más ampliamente difundidas.

A continuación, se presenta un análisis comparativo del estado del arte del modelado del estudiante a la luz de los tres aspectos que se consideran más relevantes respecto al posterior desarrollo de esta tesis: tipo de modelo del estudiante considerado, taxonomía de conocimientos del estudiante soportada por el modelado del estudiante y el método diagnóstico cognitivo adoptado. Finalmente, se presentan las conclusiones obtenidas tras el análisis de los aspectos citados anteriormente.

### 2.5.1 Análisis crítico de las taxonomías soportadas por los modelos del estudiante

En la Tabla 2.7, se describen por filas los SITs o MEs que se han considerado de mayor interés desde el punto de vista de los tipos de conocimientos y/o taxonomías subyacentes. Por columnas, se han añadido los tipos de conocimientos esenciales que incluyen los MEs y que sirven de soporte para el método de diagnóstico inmerso en el modelado del estudiante. En una celda (i, j) cualquiera, se representa con un '+' que el Modelo o SIT i incluye el tipo de conocimiento j pero no existe o se desconoce una jerarquía de conocimientos asociada a j. Si por el contrario, en el modelo o SIT i se ha considerado el conocimiento j y, además, alguna jerarquía de conocimientos asociada a él, se menciona esta jerarquía en la celda correspondiente. Asimismo, en una celda (i, j) se representa con un '-' que el Modelo o SIT i no considera el tipo de conocimiento j o se desconoce. La columna .ºtros"sirve para representar cualquier otro tipo de conocimiento que no sea perfil del estudiante, objetivo de conocimiento, objeto de conocimiento, estado de conocimiento, y traza de conocimiento y que un SIT puede incluir en su ME.

Tabla 2.7: Análisis de tipos de conocimientos en los MEs

| SIT O MODELO | PERFIL | OBJETIVOS | OBJETOS | ESTADO | TRAZA | OTROS |
|---|---|---|---|---|---|---|
| **Modelo de OLAE** (Martin and Vanlehn (1995)) | - | - | Hecho, Regla del dominio (Física), Acción del estudiante | Estado de objetos de conocimiento | - | - |
| **SHERLOCK II** (Katz et al. (1994)) | Mediante variables de conocimiento: Factores afectivos, Disposición en el comportamiento | Mediante variables de conocimiento: Habilidades | Variables de conocimiento (habilidades, etc.), Acción del alumno, Planes para resolver el problema | Estado de variables de conocimiento, Valoración del estudiante (nivel del alumno) | Secuencia de objetivos alcanzados, Traza de acciones | - |
| **POLA** (Conati and VanLehn (1996)) | - | - | Hecho, Regla, Acción del estudiante, Planes para resolver un problema, Caminos de resolución posibles asociados a los planes | Conocimiento correcto y errores, Camino de resolución del problema más probable | - | - |

Tabla 2.7: Análisis de tipos de conocimientos en los MEs (cont.)

| SIT o MODELO | PERFIL | OBJETIVOS | OBJETOS | ESTADO | TRAZA | OTROS |
|---|---|---|---|---|---|---|
| **Sistema I-Help** (Bull et al. (2001)) | Estilo de aprendizaje (Riding and Cheema (1991)), Interés, Preferencias de interacción (de ayudantes, etc.), Entusiasmo en participar, Tipo de participación en discusiones | - | + | Utilidad del ayudante | - | - |
| **Modelo de Chen y Mizoguchi** (Chen and Mizoguchi (2004)) | Datos personales, Estilo de aprendizaje, Experiencia, Motivación, Medios de interacción preferidos | - | Acción del estudiante, Plan seguido por el estudiante | Estado de conocimiento (correcto y defectuoso), Estado afectivo, Valoración del estudiante (de conocimientos, aprendizaje, y experiencia), Datos acumulados | Traza de la actividad del estudiante, Traza de acción | Objetos de aprendizaje, Información del modelo |
| **Ontología OMNIBUS** (Mizoguchi et al. (2007)) | Datos personales Conocimiento a priori | - | Tipos de acciones | Tipos de estados del estudiante (según diversas teorías) | - | Tipos de eventos educativos, Descripción de estrategias, etc. |

Tabla 2.7: Análisis de tipos de conocimientos en los MEs (cont.)

| SIT o MODELO | PERFIL | OBJETIVOS | OBJETOS | ESTADO | TRAZA | OTROS |
|---|---|---|---|---|---|---|
| **LAHYSTOTRAIN** | Datos personales, Preferencias del estudiante, | - | Acciones del estudiante, Conocimiento (patologías e instrumental), Errores del estudiante | - | - | - |
| **VI-MED** (Mili et al. (2008)) | Datos personales, Número de pacientes asignados en el turno, Mezcla de enfermedades para el turno, Frecuencia de errores en datos del laboratorio, Frecuencia de errores en datos del doctor, Frecuencia de cambios en el estado de los pacientes, | - | - | Métricas de valoración de la ejecución del estudiante | *Log* de la sesión (turno) del estudiante, *Log* por cada evento de un paciente en una sesión | - |
| **LaSiTo** (Noguez and Huesca (2008)) | Formación académica, Interacción previa, Rasgos de personalidad (según Conati and Maclaren (2004)), Categoría del estudiante, etc. | Metas generales: Aprender tópicos del experimento, Realizar el experimento con éxito, Completar el experimento lo antes posible | Tema de conocimiento, Subtema de conocimiento, Elemento (ítem) de conocimiento, Resultado de experimentos, Comportamiento del estudiante | Estado de objetos de conocimiento Situación tutorial: Estado de alcance de metas, Duración de experimentos, etc. Estado afectivo (según Ortony et al. (1988)) | - | - |

Tabla 2.7: Análisis de tipos de conocimientos en los MEs (cont.)

| SIT O MODELO | PERFIL | OBJETIVOS | OBJETOS | ESTADO | TRAZA | OTROS |
|---|---|---|---|---|---|---|
| **Modelo conceptual de Entornos de Aprendizaje Virtual Personalizado de Xu et al.** (Dongming Xu and Wang (2005)) | Datos personales | + | Conocimiento, Plan de aprendizaje | Historia de aprendizaje del estudiante: Actividades, Tiempo de aprendizaje, Ejercicios realizados, etc. | - | Información de *metanivel* (contexto de aprendizaje): Situación, Interacción, Proceso |
| **HERA** | - | - (en el módulo pedagógico, no en el ME) | Clasificación de errores que puede cometer el estudiante (según Hollnagel (1993)) y otros añadidos, Plan de tareas del estudiante | - | Traza de actividad del estudiante (secuencia de tareas complejas y/o simples del estudiante, errores cometidos, riesgos causados por acciones/decisiones del estudiante) | - |

Como ya se vió en el apartado 2.1, han surgido numerosos MEs que representaban sólo el estado del conocimiento del estudiante sobre una materia. A este tipo pertenecen los siguientes modelos:

- MEs que sólo representan conocimiento correcto (*Modelos Superpuestos* o *Modelos Diferenciales*). Su desventaja principal es que asumen que el conocimiento del estudiante es estrictamente un subconjunto del conocimiento de un experto cuando, frecuentemente, no es el caso (un estudiante puede poseer conocimiento, equivocado o no, que no tengan los expertos) y,

- MEs que también representan conocimiento incorrecto, con diferentes aproximaciones para el desarrollo de la biblioteca de errores. Estos últimos tipos de modelos tenían la ventaja de que, al incluir posibles errores en el modelo, ayudan a comprender mejor al estudiante.

En contraposición a estos MEs, aparecieron también muchos otros MEs más completos, que representan el proceso de razonamiento del estudiante. De acuerdo a Clancey (Clancey (1986)), estos modelos pueden dividirse en modelos de simulación del comportamiento, que sólo describen las acciones que el estudiante está realizando (ejemplo destacado es OLAE, mostrado en la tabla 2.7) y los *Modelos de Simulación Funcional*, como por ejemplo SHERLOCK II y POLA, también incluidos en la tabla anterior, y que destacan porque, además, describen las creencias y metas, es decir, lo que el estudiante conoce y lo que está intentando hacer. La ventaja más importante de estos modelos sobre los anteriores es su potencial para generalizar la explicación del razonamiento del estudiante, y para poder modelar conceptos erróneos y poder diferenciar entre errores de hecho, errores en metas, errores en los planes, etc., así como para poder anticipar situaciones. Por lo tanto, los modelos de simulación funcional son los modelos que tienen mayor capacidad para explicar el comportamiento del estudiante y a los que, en nuestra opinión, se debe tender en el desarrollo de SITs.

A pesar de no haber sido incorporadas a la tabla anterior por ser de propósito más general, merecen también ser destacadas por sus aportaciones en este campo las siguientes taxonomías para el modelado del estudiante:

- La taxonomía de De Koning y Bredeweg (de Koning and Bredeweg (1998)), basada en el marco multiestratificado, KADS (Wielinga et al. (1992)). Esta taxonomía distingue como un nivel de conocimiento añadido el *conocimiento estratégico*, permitiendo así representar las metas en la resolución de problemas, el método para alcanzarlas, y el conocimiento requerido para razonar con ellas. En relación a esto, ha sido cada vez más unánime la idea de que en la formación del estudiante se debería considerar su proceso metacognitivo (ejemplo de este tipo de sistemas es TAPS(Derry (1992))).

- La taxonomía de McCalla y Greer (McCalla and Greer (1994)) se sustenta en la idea de razonamiento basado en granularidad (nivel de detalle en la visión de un concepto). Esta característica, incorporada explícitamente en un SIT, puede facilitar el diagnóstico del comportamiento del estudiante a diferentes niveles, y por ello ha sido aplicada también en otros trabajos posteriores, por ejemplo, en el modelo GET-BITs usado en el sistema FLUTE (Jerinic and Devedzic (2000)) o en el ME del laboratorio virtual LaSiTo (Noguez and Huesca (2008)).

Una de las principales conclusiones globales que se pueden extraer del análisis de las aproximaciones mostradas en el campo del modelado del estudiante, es que la mayoría de los trabajos no se sustentan en una taxonomía de conocimientos del estudiante elaborada así como, en general, una falta de representación explícita de la conceptualización en la que se basa cada uno de ellos. La mayoría, tal y como refleja la tabla previa, reconocen objetos de conocimiento, estado de conocimiento del estudiante, objetivos de conocimiento, etc., pero apenas existen modelos que representen expresamente en el modelo, y de forma clara y completa, la jerarquía de subconceptos de los que se componen los tipos de conocimientos. Un ejemplo claro es la representación del estado del conocimiento del estudiante; en gran parte de los modelos (OLAE, POLA, SHERLOCK II, etc.) no hay representación, como tal, del estado del conocimiento del estudiante, sino que es un conocimiento extraído en base a valores de variables obtenidos durante el proceso de diagnóstico asociado durante el modelado. Sin embargo, la ontología OMNIBUS, en contraposición, presenta una jerarquía de posibles tipos de estados del estudiante (tabla 2.7). Una de las consecuencias importantes de esta situación es la dificultad, en gran medida, para reutilizar en ellos partes de conocimiento de otros sistemas. Además, en el desarrollo del modelado del estudiante en un SIT esta cuestión, desde nuestro punto de vista, es clave para lograr un aprendizaje adaptativo y lograr un diagnóstico del estado cognitivo del estudiante más completo. Una excepción a esta crítica es la ontología OMNIBUS, con una extensa base teórica acerca de teorías y estrategias de aprendizaje. No obstante, esta ontología se encuentra en la actualidad, dentro del proceso de aprendizaje, dirigida fundamentalmente a instructores, y a facilitar el diseño formativo, más que a estudiantes como usuarios finales, en los que está enfocado este trabajo. Por ello, como aplicación al modelado del estudiante, la ontología OMNIBUS carece de información sobre bastantes tipos de conocimientos que se consideran un soporte fundamental para el método de diagnóstico asociado al modelado del estudiante pero sí posee otro tipo de conocimientos relacionado con teorías y estrategias de aprendizaje.

Como puede observarse en la tabla 2.7, hay escasos trabajos que modelen y traten a la vez diversas características individuales del estudiante. Hay propuestas, como el modelo de SITs basado en Web de Zhiping et al. (2008), que se centran en varias características del estudiante: psicológicas, estilos de aprendizaje, etc., pero, a juicio de la autora, sin un tratamiento en profundidad de estas características. En esta línea, existen trabajos interesantes pero muy específicos, especializados, por ejemplo, en el modelado de estilos de aprendizaje como el de Carmona et al. (2008) o el modelado del comportamiento afectivo (ABM) para SITs de Hernández et al. (2008).

Merece especial atención la propuesta de Chen y Mizoguchi, donde se define una ontología y un agente para el ME. Sin embargo, una vez analizada en profundidad, adolece de ciertas carencias importantes:

- Ausencia de información relacionada con los objetivos de aprendizaje del estudiante.
- Escasa categorización sobre la mayoría de los tipos de conocimientos considerados en la taxonomía (por ejemplo, sobre conocimiento defectuoso, o sobre la valoración del estudiante).
- En general, gran parte de la jerarquía de conocimientos adolece de claridad tanto en la descripción de los conceptos como en su estructuración de la ontología. Como ejemplo, la distinción entre información estática y dinámica o la propia categorización dentro de información dinámica.

Otro aspecto a destacar en este análisis es la omisión en la mayoría de las taxonomías indicadas de los objetivos de aprendizaje considerado en este trabajo esencial para que el tutor

pueda llevar a cabo una tutoría más personalizada y poder realizar no sólo un diagnóstico de grano más fino (diagnóstico cognitivo y a un determinado nivel de granularidad) sino también un diagnóstico de grano más grueso (diagnóstico pedagógico, a nivel de objetivos alcanzados o no por el estudiante). Desde el prisma del modelado del estudiante en los EVIEs, a las carencias anteriores, se añade, como se puede observar en la tabla 2.7, la ausencia de ciertos tipos de conocimientos específicos de estos entornos, y que consideramos claves para permitir un diagnóstico cognitivo completo y una tutoría del estudiante más invidualizada. Entre estos elementos de información, caben destacar los siguientes:

- Conocimiento relacionado con el dominio espacial (escenarios, subescenarios, y conexiones entre ellos, objetos específicos de diferente naturaleza con los que el estudiante puede interactuar y sus posiciones, etc.).
- Nuevos tipos de acciones que puede realizar un estudiante en este tipo de entornos de simulación (mirar, hablar, interactuar de diferentes formas con los objetos del escenario geométrico, moverse de una posición inicial a otra posición de destino, etc.).
- Conocimiento de las trayectorias que puede seguir el estudiante durante su aprendizaje.

### 2.5.2 Análisis crítico de los métodos de diagnóstico cognitivo soportados por los modelos del estudiante

En la siguiente tabla se describen, por filas los métodos de diagnóstico que se han considerado de mayor interés en el campo del diagnóstico cognitivo. Es importante señalar que, hemos añadido sólo métodos que han incluido en su proceso de diagnóstico la gestión de conocimiento incompleto dado que, como ya se indicó en x, la propia naturaleza del razonamiento humano (del razonamiento del estudiante que se está modelando) es no monótona. Por columnas se han añadido como características analizadas en los métodos, por orden, las siguientes: en qué tipo de métodos de diagnóstico se enmarca, si en el tratamiento de la no monotonía el método es capaz de detectar y resolver diferentes tipos de conflictos (por ejemplo, contradicciones por descuidos, contradicciones por olvidos, contradicciones por el propio conocimiento inconsistente que el estudiante pueda poseer en un instante de su aprendizaje, etc.), si es un método de diagnóstico que implica de alguna forma explícitamente al usuario en el diagnóstico y finalmente, si el método posee capacidades de meta-diagnosis. Del mismo modo que en el análisis previo, en una celda (i, j) cualquiera, se representa con un '+' que el método i posee la característica j. Por el contrario, si en una celda (i, j) aparece un '-' representa que el método no incorpora la característica j. En otro caso, el contenido asociado a la celda representa lo mismo que '+' pero complementado con el significado de su información.

Tabla 2.8: Análisis de los métodos de Diagnóstico Cognitivo

| MÉTODO | Tipo de método | Tipos de contradicciones | Implicación del estudiante | Meta-diagnóstico |
|---|---|---|---|---|
| **Motor de Diagnóstico General (MDG)** (de Kleer and Williams (1987)) | Basado en modelos de dispositivo | - | - | - |
| **Extensión del MDG** (de Koning and Bredeweg (1998)) | Basado en modelo de dispositivo | - | - | + |
| **HSMIS** (Ikeda et al. (1993)) | Basado en inferencia inductiva | Contradicciones por descuidos<br>Contradicciones por cambios en la mente<br>Contradicciones por conocimiento inconsistente<br>Contradicciones por suposiciones en el modelado | - | - |
| **Método de Descomposición-P** (Tsybenko (1995)) | Generación de modelos de dispositivos asociados | - | - | - |
| **Diagnóstico en TAGUS** (Paiva and Self (1995)) | Basado en el cambio externo y dinámico del ME | - | + | + |

Tabla 2.8: Análisis de los métodos de Diagnóstico Cognitivo (cont.)

| MÉTODO | **Tipo de método** | **Tipos de contradicciones** | **Implicación del estudiante** | **Meta-diagnóstico** |
|---|---|---|---|---|
| **Diagnóstico en UMT** (Brajnik and Tasso (1994)) | Basado en el cambio externo y dinámico del ME | - | - | - |
| **Diagnóstico en Mr. Collins** (Bull and Smith (1997)) | Diagnóstico colaborativo Basado en revisión y negociación con el estudiante | - | + | - |
| **Diagnóstico en STYLE_OLM** (Dimitrova et al. (1999)) | Diagnóstico interactivo | - | + | - |

El análisis del diagnóstico cognitivo en el modelado del estudiantes en SITs y EVIEs refleja que la evolución en los métodos y técnicas para modelado del estudiante han traído consigo el desarrollo de nuevas soluciones para diagnosticar el estado del estudiante a partir de su comportamiento. Los primeros avances surgieron de los métodos generales de diagnóstico en IA. Ejemplos destacados fueron el trabajo de Self, J. (ver sección 2.3.3) y el trabajo de de Koning, K. et al. (ver sección 2.3.4). Ambos aplican el paradigma del Motor de Diagnóstico General (de Kleer and Williams (1987)) para definir la naturaleza del problema del diagnóstico cognitivo como un caso particular de diagnóstico de dispositivos en IA pero con diferencias. Los problemas añadidos en el diagnóstico cognitivo surgen por su naturaleza interactiva distinta al diagnóstico de dispositivos (el estudiante puede participar en el proceso de diagnóstico cognitivo), al propósito del diagnóstico cognitivo (ayudar al tutor en el proceso de aprendizaje del estudiante) completamente diferente al propósito del diagnóstico de dispositivos (reparar o sustituir componentes defectuosos) y a la imposibilidad de disponer de un circuito inicial que describa el conocimiento del estudiante a diferencia del diagnóstico de dispositivos (Self (1993)). Más tarde, de Koning, K., et al. (de Koning et al. (1995)) adaptaron la versión del Motor de Diagnóstico General para resolver uno de los problemas planteados previamente por Self: definir un nivel de *meta-diagnóstico*. Sin embargo, la limitación fundamental de las aproximaciones mencionadas previamente es que intentan aplicar técnicas basadas en modelos (tabla 2.8). Sin embargo, frecuentemente, el estudiante no posee un único método para resolver un problema. Así pues, no hay un modelo de dispositivo concreto a priori que pueda ser manejado por el diagnóstico cognitivo. Para resolver este problema, surgió otra propuesta (Tsybenko (1995)) que permite generar los modelos asociados al estudiantes para ser usados por el diagnóstico cognitivo durante la resolución del problema.

Como se observa en la tabla 2.8, es también importante subrayar que hay pocos métodos de diagnóstico que incluyan en su formulación la naturaleza no monótona del razonamiento acerca del estudiante y que, a la vez, detecten y resuelvan diferentes tipos de contradicciones que pueden surgir en este proceso. Una excepción es el sistema de diagnóstico SMDS de Ikeda et al. soportado por un ATMS (*A*ssumption-based *T*ruth *M*aintenance *S*ystem).

Otras líneas de investigación, no detalladas en el estado del arte previo, pero sí incluidas en la tabla precedente por su relación con el diagnóstico cognitivo, están relacionadas con diferentes sistemas educativos inteligentes que involucran al estudiante en el método de diagnóstico para así mejorar la acomodación del sistema. Entre estos sistemas, caben destacar los siguientes: el modelado del estudiante colaborativo en Mr Collins (Bull and Smith (1997)), modelos del estudiante abiertos como el workbench TAGUS (Paiva and Self (1995)), o el diagnóstico interactivo de la herramienta STyLE-OLM (Dimitrova et al. (1999)).

Como conclusión final del análisis del estado del arte de los métodos de diagnóstico propuestos para el ME, se puede afirmar que no hay, en general, métodos que sean capaces de llevar a cabo un diagnóstico cognitivo flexible. La característica de flexibilidad viene determinada, entre otros, por los siguientes factores: a) soporte de una adecuada y extensa taxonomía de conocimientos acerca del estudiante, b) capacidad de razonamiento no monótono, c) capacidad de distinguir diferentes tipos de contradicciones que puedan surgir en el proceso de diagnóstico y, d) resolución de las contradicciones detectadas en el diagnóstico acorde a su distinta naturaleza.

CAPÍTULO 3

# FUNDAMENTOS TEÓRICOS Y TECNOLÓGICOS

En este apartado se realiza una breve descripción de las bases teóricas y tecnológicas que soportan el modelado del estudiante propuesto. El objetivo es introducir y facilitar la comprensión de la solución expuesta en el capítulo 5 de este trabajo.

En primer lugar, se describe la revisión de creencias y el razonamiento no monótono, ambos considerados pilares del mecanismo de diagnóstico inherente al ME. En segundo lugar, se introduce el diseño instruccional, puesto que el modelado del estudiante presentado en la tesis requiere que los SITs a los que se aplique sigan una aproximación de diseño instruccional. Finalmente, se esbozan aspectos de la ingeniería ontológica, que nos ha proporcionado las ventajas de las ontologías como formalismo de representación de los conocimientos del estudiante en el modelado, así como las metodologías, lenguajes y herramientas necesarias para su desarrollo.

## 3.1 Revisión de creencias y razonamiento no monótono

Cuando resolvemos un determinado problema en la vida real es frecuente no disponer de conocimiento completo acerca del dominio del problema y, como consecuencia, no somos capaces de decidir con toda seguridad qué acción realizar a continuación o qué conocimiento manejado es cierto o falso. Esto es lo que sucede con el modelado del proceso de razonamiento, ya que requiere manejar normalmente conocimiento incompleto. Puede inclusive que, aunque este conocimiento sea completo, el modelado del conocimiento que poseemos del dominio sea imposible o no resulte práctico.

Una alternativa para solucionar el problema del razonamiento apoyado en conocimiento incompleto es adoptar hipótesis o suposiciones. De este modo, si durante el proceso de razonamiento, la adquisición de nuevo conocimiento invalida alguna de las hipótesis usadas (es decir, da lugar a que alguna de las hipótesis supuestas ciertas -o falsa- resulte estar equivocada), las inferencias realizadas a partir de esas hipótesis también dejarán de tener base. A este tipo de razonamiento se le denomina *razonamiento no monótono* y el mecanismo requerido para llevarlo a cabo *revisión de creencias*. Este mecanismo permitirá:

- Detectar las equivocaciones que se cometan.

- Deshacer las hipótesis equivocadas y
- Rechazar todas las conclusiones que se hayan inferido partiendo de las hipótesis equivocadas.

La revisión de creencias es un proceso realizado por los seres humanos de forma natural, pero es difícil, aunque necesario, trasladarlo a un sistema software. Fundamentalmente, es útil la revisión de creencias en sistemas software con alguna de las siguientes características:

- *Sistemas que presentan variación de datos con el tiempo.* Por ejemplo, un sistema de diagnóstico de un dispositivo físico en el que el estado de los componentes varía con el tiempo.
- *Sistemas que realizan razonamiento no monótono y suponen creencias por omisión.* Por ejemplo, un sistema de planificación de tareas en el que un robot debe realizar una acción concreta, como levantar un objeto. El sistema puede asumir por omisión que el estado del robot es correcto, que el objeto no es demasiado pesado para el robot y otras suposiciones. Si el sistema se da cuenta de que hay alguna circunstancia que afecta a las consideraciones asumidas inicialmente, todas las acciones o conclusiones derivadas de estas hipótesis deben revisarse.
- *Sistemas en los que aparecen contradicciones.* Un sistema inteligente puede generar contradicciones porque sus premisas (información suministrada al sistema) son inconsistentes, porque las suposiciones adoptadas por omisión contradicen las observaciones de la realidad, o porque algunas de las suposiciones son excluyentes entre sí.

### 3.1.1 Razonamiento no monótono y diagnóstico cognitivo

El Diagnóstico Cognitivo, como el diagnóstico de dispositivos, maneja un conocimiento del dominio que es incompleto, el estado cognitivo del estudiante. La forma de adquirir nuevo conocimiento en este dominio es a través de inferencias sobre el conocimiento que se tiene del alumno y mediante observaciones de su comportamiento frente a diferentes problemas que se le plantean.

Un SIT no puede detener el proceso de enseñanza aunque el conocimiento sobre el alumno sea incompleto. Para resolver este problema, el sistema debe realizar un conjunto de hipótesis que le permita continuar con sus funciones. No obstante, a medida que avanza el proceso de razonamiento, es posible que se descubra que algunas de las hipótesis supuestas anteriormente resulten equivocadas y, en consecuencia, también lo que se ha inferido a partir de ellas.

Otro aspecto importante reside en la misma naturaleza del sujeto que interactúa con el SIT. A medida que transcurre el tiempo, el estudiante aprende nuevos conceptos y puede también olvidar otros previamente adquiridos por él.

Dadas, pues, estas características propias de la naturaleza del diagnóstico cognitivo, este mecanismo necesita un procedimiento de razonamiento capaz de enfrentarse a:

- Conocimiento incompleto.
- Necesidad de formulación de hipótesis y toma de decisiones.
- Eventual invalidación de esas hipótesis y, por lo tanto, de las inferencias realizadas a partir de ellas.

Este tipo de razonamiento es, como se ha visto anteriormente, el razonamiento no monótono, y un mecanismo capaz de detectar contradicciones entre las hipótesis que se manejan y deshacer las inferencias que resulten equivocas es la revisión de creencias.

### 3.1.2 Aproximaciones a la revisión de creencias

En el campo de la Inteligencia Artificial, han surgido distintas aproximaciones para resolver el problema de la revisión de creencias, entre las que destacan las siguientes [DOMÍ93?]: aproximación basada en STRIPS, aproximación basada en demonios disparados por cambios, y aproximación basada en dependencia de datos. Las dos primeras aproximaciones anteriores no disponen explícitamente de las interdependencias entre las distintas creencias (hipótesis realizadas y hechos existentes en el problema). Esta característica permitiría que, conociendo las distintas premisas de las que depende una creencia, si hay que retractarse de una premisa dada, no será necesario revisar el conjunto de todas las creencias, sino sólo aquellas que se vean afectadas y, de la misma manera, si somos capaces de identificar la hipótesis que se encuentra detras de una contradicción, no es necesario revisar el conjunto de todas las suposiciones, sino sólo aquéllas que constituyen la base de la contradicción detectada.

La idea previa es el fundamento de los sistemas de mantenimiento de la verdad (TMS - *T*ruth *M*aintenance *S*ystem), cuya misión es realizar o disponer de un registro explícito de las dependencias mutuas entre las creencias que se manejan en un sistema inteligente. Esta información se podrá usar para retirar de forma selectiva todas las conclusiones alcanzadas a partir de una premisa de la que nos hemos retractado y hacer una traza de una contradicción cuando surja, llegando a sus premisas o hipótesis primeras para retractarnos o estudiar una de ellas de manera selectiva.

Otras ventajas fundamentales de las dependencias explícitas son:

- Facilidades de explicación: siguiendo la traza inversa de una creencia derivada.
- Efecto *cache*: algunas creencias cambian muchas veces de estado (cierto/falso y viceversa). En un sistema de producción, por ejemplo, se traduce esto en una serie de reglas que se disparan en un momento dado, siempre las mismas. Mediante la representación de las dependencias de datos, este cambio de estado se realiza a través de un proceso mucho menos costoso que la selección y disparo de reglas, con la consiguiente mejora en el rendimiento del sistema que lo use.

### 3.1.3 Sistemas de Mantenimiento de Verdad

Un TMS es un sistema que maneja las relaciones explícitas entre las distintas creencias de la base de conocimientos. Su objetivo es garantizar la coherencia del conjunto de creencias, evitando las contradicciones, e impidiendo la presencia en la BC de ninguna creencia que no tenga una razón para ello.

El TMS puede verse como una memoria dinámica asociada a un motor de inferencia (MI) [MART06]. La idea básica es que, cada vez que un MI llega a una conclusión, se almacena en dicha memoria la justificación de dicha conclusión en forma de implicación. Por ejemplo, si se ha deducido C a partir de A y B, se almacena A, B ->C. Si después se deduce E a partir de C y D, se almacena C, D ->E. Si se deduce Q de M y N, se almacena M, N ->Q. Supóngase que, en ese momento, el MI desea retraer el hecho A, es decir, reconsiderar el hecho A asumiendo que ahora no es verdadero. En este caso, apoyándose en las justificaciones, puede concluirse

que también deben retraerse los hechos C y E, pero se mantienen otros, como el hecho Q. Este proceso de verificación sobre qué hechos se mantienen y qué hechos dejan de deducirse cuando se reconsideran supuestos es la tarea principal que realiza un TMS, apoyándose en la memoria de justificaciones suministrada por el MI.

Dependiendo de la forma en la que se identifican las interdependencias entre creencias, se pueden clasificar los TMS en los siguientes tipos:

- Orientados a justificaciones:
  - Basados en justificaciones: JTMS.
  - Basados en hipótesis: ATMS.
- Basados en lógica:
  - Lógica clásica: LTMS.
  - Lógica de relevancia (*relevance logic*), como el *Multiple Belief Reasoner* (MBR) [MART83].

#### 3.1.3.1 Elementos de un TMS

Las entidades básicas manejadas por un TMS son nodos y justificaciones.

**Nodo** representa una creencia individual.

**Justificación** es la estructura que representa la dependencia entre distintas creencias, es decir, entre distintos nodos.

Una justificación da base para la creencia en un nodo en función de la creencia o no creencia en los nodos con los que está relacionado. En el caso general, las justificaciones se expresan así:

CREER en (p1, p2, p3, ..., pm) y
NO CREER en (q1, q2, ..., qn) (3.1)
es una razón válida para CREER en r

**Premisa** nodo que se asume como cierto sin necesidad de estar basado en una justificación; se toma como cierto sin más consideraciones.

Es frecuente adoptar la representación gráfica usada por Michael Reinfrank en su trabajo *"Fundamentals and logical foundations of truth maintenance"* para las justificaciones y que se muestra a continuación con un ejemplo:

En la figura, cada uno de los tres nodos tiene asociada una fórmula, de tal forma que esta asociación es única, es decir, nunca habrá dos fórmulas distintas que estén asociadas al mismo nodo del TMS ni al revés. Por ejemplo, el nodo de la esquina superior izquierda tiene asociado la fórmula *ROBOT A DISPONIBLE*. Llamaremos a este nodo N1, y así con el resto de los nodos. Por lo tanto, según 3.3, podemos escribir la justificación de la siguiente forma:

CREER en N1 y
NO CREER en N2 (3.2)
es una razón válida para CREER en N3

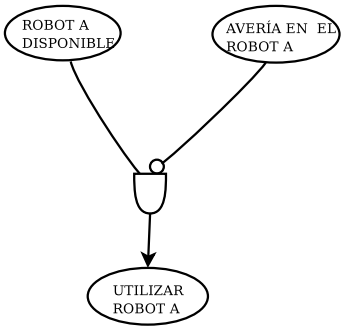


**Figura 3.1:** Representación gráfica de una justificación

La fórmula contenida en un nodo, por ejemplo la fórmula *ROBOT A DISPONIBLE* asociada al nodo N1, tiene pleno significado para el MI que trabaja con el TMS. Sin embargo, el TMS maneja la justificación adecuada con los nodos, en este ejemplo, N1, N2 y N3, sin importarle las fórmulas asociadas a cada uno de ellos.

#### 3.1.3.2 Estructura y ciclo de funcionamiento de un sistema basado en un TMS

El TMS funciona como un sistema de gestión de base de datos que se encarga de mantener las creencias relativas al problema sin preocuparse del carácter cierto o falso de las creencias desde el punto de vista de la lógica, sino que garantiza que todas las creencias manejadas por el sistema cumplen un criterio de coherencia (evitar mantener creencias contradictorias) o racionalidad (mantener la presencia en la BC de una creencia sólo mientras se tenga una razón para ello).

La estructura básica de un sistema basado en un TMS se representa en la siguiente figura:

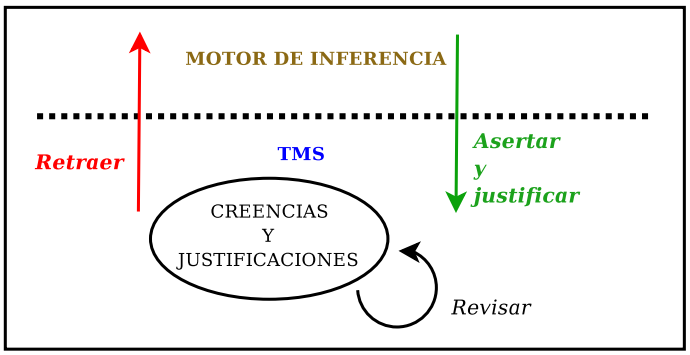


**Figura 3.2:** Estructura básica de un sistema basado en un TMS

### 3.1.4 Red de dependencias

Una red de dependencias es un modelo de BC que dispone de información explícita sobre las dependencias entre los elementos de la base (creencias).

Se define la red de dependencias como una terna D=(N, P, J) donde:

N: es un conjunto finito de nodos tomados del universo de trabajo U (que puede ser infinito).

P ⊆ N (puede ser vacío): es un conjunto de premisas (nodos asumidos como ciertos sin que deban estar apoyados por una justificación).
J: es un conjunto de justificaciones de la forma < A | B → c >, donde A y B son conjuntos de nodos, que pueden ser vacíos, y c es un solo nodo.

Sea j = < A | B → c > una justificación, decimos que:
A = cm(j) = {p1, p2, . . . , pm}, es el conjunto de componentes monótonas de la justificación:

CREER en (p1, p2, ..., pm) y
NO CREER en (q1, q2, ..., qn) (3.3)
es una razón válida para CREER en c.

B = cnm(j) = {p1, p2, . . . , pm}, es el conjunto de componentes no monótonas de la justificación.
c = con(j), es la conclusión de la justificación.

Los conjuntos A y B pueden expresarse en forma de enumeración entre llaves, y en caso de estar vacía una de las componentes, se puede omitir:

< a, b → c >es igual a < {a, b} | $\phi$ → c >

Sea F el conjunto de todas las fórmulas bien construidas del lenguaje formal que utiliza el SBC para representar las creencias, hay una función:

form: N → F que a cada nodo *n* le hace corresponder su fórmula asociada.

### 3.1.5 ATMS

ATMS (de Kleer (1986)) adopta la misma estructura de sistema de razonamiento que el JTMS convencional, con un MI y el TMS. Las inferencias realizadas por el MI le son comunicadas al TMS, que determina las creencias que se sostienen y las que no. Esta tarea se lleva a cabo mediante inferencia proposicional, en la que las justificaciones hacen el papel de axiomas proposicionales.

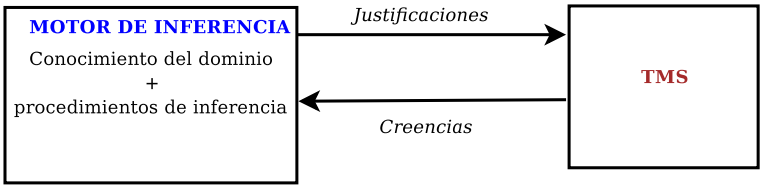


**Figura 3.3:** Estructura del ATMS

#### 3.1.5.1 Definiciones básicas

A continuación, se definen los conceptos básicos necesarios de los sistemas ATMS:

- *Hipótesis:* son las razones últimas que dan fundamentación a la creencia en cualquiera de los datos que maneja el MI.
- *Nodo ATMS:* representa un dato que maneja el MI. Una hipótesis es un nodo especial.
- *Justificación ATMS:* describe cómo se deriva un nodo a partir de otros. Consta de las siguientes partes:

  *consecuente*: nodo derivado.

*antecedentes*: lista de nodos.

*informador*: descripción de la justificación del MI.

Las justificaciones se escriben de la forma siguiente:

$x_1, x_2, \ldots \Rightarrow n$

donde n es el consecuente y $x_1, x_2, \ldots$ los antecedentes. Una justificación representa una cláusula de Horn:

$x_1 \wedge x_2 \wedge \ldots \rightarrow n$

- *Entorno ATMS*: es un conjunto de hipótesis. Desde el punto de vista lógico es una conjunción de hipótesis.

  Un nodo *n* se sostiene en un entorno E, si *n* puede derivarse de *E* y del conjunto de justificaciones *J* actual. En forma de cálculo proposicional:

  $E, J \vdash n$

  *E* es una conjunción de átomos y *J* un conjunto de implicaciones.

  Un entorno es inconsistente si FALSO ($\perp$) es derivable proposicionalmente:

  $E, J \vdash \perp$

- *Contexto ATMS*: es un conjunto formado por las hipótesis de un entorno consistente y todos los nodos derivables a partir de esas hipótesis.

- *Entorno característico para un contexto*: es un conjunto de hipótesis del que se pueden derivar todos los nodos del contexto. Todo contexto tiene al menos un entorno característico. Normalmente (cuando las hipótesis no tienen justificaciones) cada contexto tiene exactamente un entorno característico.

Dadas las definiciones anteriores, el objetivo del ATMS es determinar los contextos de forma eficiente alimentado por un conjunto de hipótesis y justificaciones que le envía el MI.

#### 3.1.5.2 Etiquetas

Una etiqueta ATMS es un conjunto de entornos asociados a cada nodo. Todo entorno *E* de la etiqueta de *n* es consistente y tiene la propiedad:
$E, J \vdash n$ o $J \vdash E \rightarrow n$
La etiqueta describe las hipótesis de las que depende el dato en última instancia. Las construye el ATMS, al contrario que las justificaciones.
La misión del ATMS es garantizar que la etiqueta de cada nodo es:

- *Consistente*: todos sus entornos son consistentes.

- *Bien fundamentada*. Si *n* es derivable de cada entorno *E* de la etiqueta, se dice que la etiqueta está bien fundamentada. $J \vdash E \rightarrow n$

- *Completa*. La etiqueta de un nodo *n* es completa si todo entorno *E* consistente para el cual se cumple $J \vdash E \rightarrow n$ es un superconjunto de algún entorno *E'* de la etiqueta de n, es decir, verifica que $E' \subset E$.

- *Mínima*. La etiqueta es mínima si ningún entorno de la misma es superconjunto de cualquier otro entorno. Para cualquier entorno E de una etiqueta, no debe existir otro entorno E' de una etiqueta que cumpla $E' \subset E$.

Como consecuencia de estas definiciones:

- Un nodo *n* es derivable de un entorno *E*, si *E* es un superconjunto de algún entorno de su etiqueta. De esta forma, el MI puede saber directamente si un nodo se mantiene en un entorno determinado.
- Un nodo tiene una etiqueta vacía sii no es derivable de un conjunto consistente de hipótesis.
- Un nodo es miembro de un contexto si puede ser derivado de las hipótesis de un entorno característico del contexto. Un nodo está en un contexto si tiene al menos un entorno que es subconjunto de un entorno característico del contexto.

Así, un nodo especifica implícitamente los contextos a los que puede pertenecer. Aquellos contextos que tengan un entorno característico que sea superconjunto de algún entorno de su etiqueta.

Hay cuatro conjuntos de nodos, sin intersección, independientemente del contexto:

- Conjunto TRUE: nodos que se sostienen en el entorno vacío. Se sostienen en cualquier contexto consistente presente o futuro. Este conjunto crece monótonamente.
- Conjunto IN: nodos con etiquetas en las que hay al menos un entorno no vacío. Se sostienen en al menos un contexto consistente no universal. Este conjunto crece no monótonamente.
- Conjunto OUT: nodos con etiqueta vacía. No se sostienen en ningún contexto consistente conocido. En el proceso de inferencia, alguno de estos nodos puede pasar al conjunto IN.
- Conjunto FALSE: nodos que no se sostienen en ningún contexto. Este conjunto crece monótonamente.

#### 3.1.5.3 Estructuras de datos básicas

La estructura básica es el nodo ATMS. Según la nomenglatura usada por (de Kleer (1986)), se representará por $\gamma_{dato}$ el nodo que maneja el ATMS y que representa el dato que manipula el MI.

$\gamma_{dato} < dato, etiqueta, justificaciones >$

donde:

*dato* es la información comprensible para el MI.

*etiqueta* es creada por el ATMS y nunca modificada por el MI y

*justificaciones* son provistas al ATMS por el MI. Consultadas por el ATMS pero nunca modificadas por él.

Tipos de nodos:

- *Premisa*: tiene una justificación sin antecedentes. Se sostiene universalmente.

  < p, {{}} , {{}} >

- *Hipótesis*: nodo con una etiqueta de un solo entorno que hace mención a sí mismo.

  $< A, \{\{A\}\}, \{(A)\} >$

  Las hipótesis pueden estar justificadas:

  $< A, \{\{A\}, \{B, C\}\}, \{(A), (d)\} >$

- *Nodo asumido*: no es una premisa ni una hipótesis y tiene una justificación que hace mención a una hipótesis.

  $< A, \{\{A\}\}, \{(A)\} >$. Este es el nodo asumido A que se apoya en la hipótesis A.

- Nodo derivado: el resto de nodos que presentan etiqueta y justificaciones con diferentes formas.

  $< w = 1, \{\{A, B\}, \{C\}, \{E\}\}, \{(b), (c, d)\} >$

Sea el nodo:
$< n, \{\{A_1, A_2, \ldots\}, \{B_1, B_2, \ldots\}, \ldots\}, \{(z_1, z_2, \ldots), (y_1, y_2, \ldots), \ldots\} >$
desde el punto de vista de la lógica, la etiqueta representa la implicación siguiente:
$(A_1 \wedge A_2 \wedge \ldots) \vee (B_1 \wedge B_2 \wedge \ldots) \vee \ldots \rightarrow n$
y las justificaciones la implicación siguiente:
$(z_1 \wedge z_2 \wedge \ldots) \vee (y_1 \wedge y_2 \wedge \ldots) \vee \ldots \rightarrow n$
A los entornos inconsistentes se les llama *nogoods* y representan conjunciones inconsistentes de hipótesis.

#### 3.1.5.4 Retículo del entorno

Todo entorno consistente caracteriza un contexto. Si hay n hipótesis, entonces hay $2^n$ contextos potenciales. Los entornos forman un retículo respecto a la relación "contenido en" (subconjuntos y superconjuntos).

Cualquier entorno que permita la derivación de $\perp$ es un *nogood* y se eliminará del retículo, así como todos sus superconjuntos.

A continuación (Figura 3.4), se muestra un ejemplo de retículo para las hipótesis A, B, C, D y E.

#### 3.1.5.5 Operaciones básicas

El ATMS dispone de tres operaciones básicas:

- Crear un nodo ATMS para cualquier dato que maneja el MI.
- Crear una hipótesis.
- Añadir una justificación a un nodo.

El ATMS es incremental. Los algoritmos aseguran que, después de cada operación primitiva, todas las etiquetas de los nodos son consistentes, bien fundamentadas, completas y minimales. Todas las operaciones ATMS se pueden construir a partir de las tres operaciones básicas.

Antes de cualquier acción ATMS, hay que asociar los datos que manipula el MI con nodos ATMS.

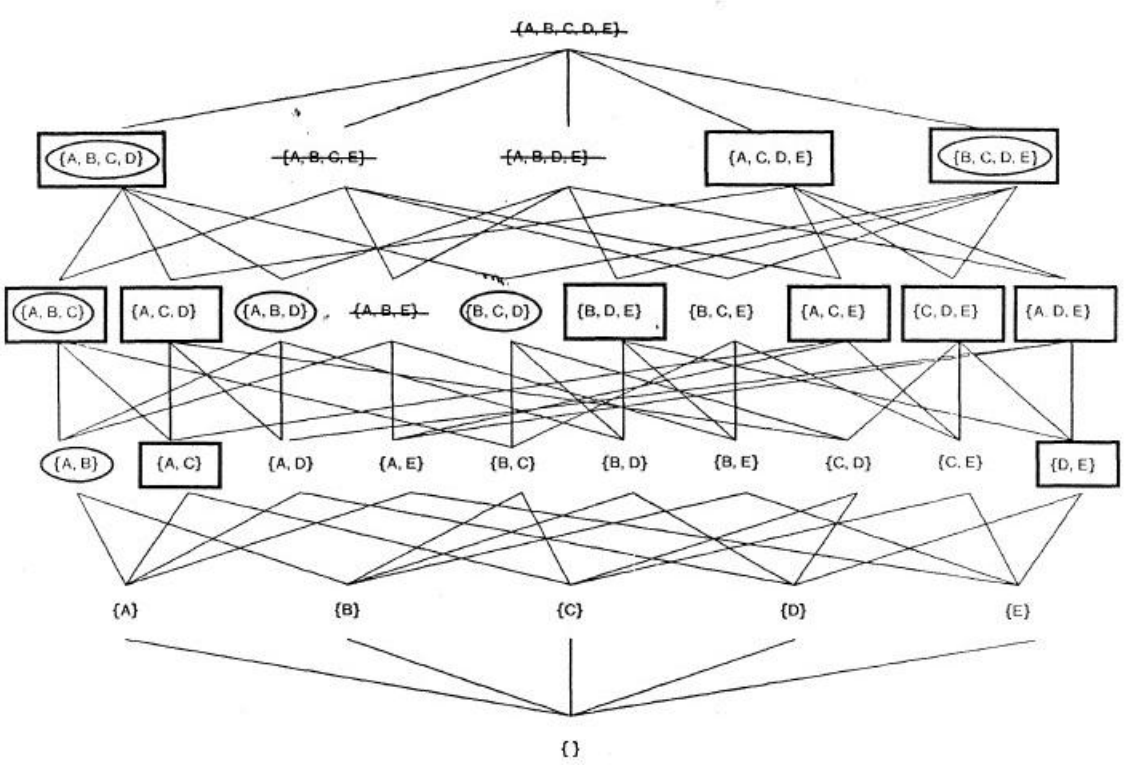


**Figura 3.4:** Ejemplo de retículo de ATMS

#### 3.1.5.6 Algoritmo de actualización de etiquetas en un ATMS

El ATMS debe asegurar que, para cada justificación, la intersección de contextos de los antecedentes es igual a los contextos del consecuente.

El algoritmo de actualización de etiquetas que satisface los objetivos de ATMS es el siguiente:

El proceso comienza en el momento en que el MI suministra una nueva justificación J de entrada al ATMS de la forma:

$P_1, \ldots, P_n \Rightarrow Q$ (en el antecedente, algunos elementos son supuestos).

1. Obtener el conjunto E (etiqueta) de Q de la siguiente forma:

   1.1 Se incluyen en E todos los conjuntos que son unión de un entorno de la etiqueta del nodo $P_1$, un entorno de la etiqueta del nodo $P_2$, ..., y un entorno de la etiqueta del nodo $P_n$, haciendo todas las combinaciones posibles.

   1.2 Se eliminan de E los superconjuntos de algún conjunto de E, y los superconjuntos de algún entorno de la etiqueta de un *nogood* (entornos inconsistentes).

   1.3 Se actualiza la etiqueta. Para ello, si A es la etiqueta anterior de Q:

      1) Se eliminan de E los entornos repetidos en A o superconjuntos de entornos de A.
      2) Se eliminan de A los entornos superconjuntos de E.
      3) La nueva etiqueta es la unión de E y de A.

2. Si E es igual que la etiqueta anterior (A), finaliza el tratamiento de ese nodo.

3. Si Q es una contradicción (nodo FALSO: $\perp$), marcar todos los entornos de E como *nogood* y recorrer todos los nodos para eliminar de sus respectivas etiquetas los entornos que incluyan algún entorno de E (que contengan al *nogood* o sean superconjuntos de él) y finalizar el tratamiento de ese nodo.

4. Si Q no es una contradicción (nodo FALSO), se actualizan recursivamente todas las etiquetas de los nodos afectados por el nuevo nodo modificado, es decir, que son consecuencia de Q, de acuerdo con las justificaciones.

Por ejemplo, consideremos el siguiente conjunto de nodos:
*Nodo a*: < a, {{A,B}, {B, C, D}}, {...} >
*Nodo b*: < b, {{A,C}, {D, E}}, {...} >
*Nodo c*: < c, {}, {} >
*Nogood*: {A, B, E}
Además, supongamos que el MI realiza una deducción y añade la siguiente justificación:
a, b $\Rightarrow$ c
Al añadir esta justificación, el ATMS debe actualizar la etiqueta del nodo *c* para que resulte consistente, completa, bien fundamentada y mínima. Para calcular la etiqueta del consecuente:

- Se hace la unión de todas las posibles combinaciones de tomar un entorno de cada etiqueta de los nodos antecedentes:

  {{A,B,C}, {A, B, C, D}, {A, B, D, E}, {B, C, D, E}}

- Se eliminan superconjuntos:

  {{A,B,C}, {A, B, D, E}, {B, C, D, E}}

- Se eliminan entornos que contengan *nogoods* (en el ejemplo, {A, B, E} es un *nogood*):

  {{A,B,C}, {B, C, D, E}}

Para que la programación de este algoritmo sea más eficiente, Forbus y de Kleer [FORB93] proponen una versión incremental del algoritmo.

### 3.1.6 Limitaciones de los JTMS versus ATMS

Johan de Kleer (de Kleer (1986)) destaca un conjunto de problemas que presentan los JTMS y que se resuelven de alguna forma con el modelo del ATMS. Estos problemas se describen brevemente a continuación:

- *Problema del estado único*. Dado un conjunto de hipótesis que admite múltiples soluciones, los algoritmos JTMS sólo permiten considerar una solución en cada momento. Esto hace difícil comparar dos soluciones igualmente posibles, algo que interesa mucho en aplicaciones de diagnóstico. Sin embargo, en un ATMS pueden coexistir diversas soluciones (incluso contradictorias) y poder así compararlas.

- *Cómo evitar contradicciones*. Si A y B son contradictorios, sólo A o B (una de las dos creencias) seguirán adelante en el proceso de razonamiento en un JTMS. Esta no es la mejor táctica en cuanto a resolución de problemas, ya que si A y B son contradictorios, se deberían evitar las inferencias sobre A y B a la vez y no sobre A o B por separado.

- *Dificultad para cambiar de estado*. El MI no puede cambiar temporalmente una hipótesis. De hecho, no puede cambiar de ninguna forma una hipótesis si ésta no se encuentra afectada por una contradicción. Pero si se introduce una contradicción, el estado de conocimiento del problema queda alterado de forma irrevocable, ya que la contradicción no se puede eliminar de la base de conocimientos (se salva la situación mediante justificaciones para que la base de conocimientos siga consistente). JTMS no tiene ninguna forma de volver a un estado anterior, lo único que puede hacer es garantizar que la base de conocimientos se encuentra libre de contradicciones. Sin embargo, en un ATMS, el cambio de estado dentro del espacio de estados del problema es muy eficiente, ya que el ATMS estudia todas las alternativas al mismo tiempo.

- *Concepto de hipótesis*. Según Doyle, una hipótesis es cualquier nodo cuya justificación soporte depende de otros nodos que están OUT. Esta consideración es dependiente del contexto. A medida que avanza el proceso de razonamiento, las justificaciones soporte y, por tanto, los nodos considerados hipótesis cambian. Esto es problemático para los motores de inferencia que consultan las hipótesis y las justificaciones en busca de datos. Sin embargo, en un ATMS las hipótesis son fácilmente identificables como, por ejemplo, las hipótesis asociadas a una contradicción.

- *Mecanismo pesado*. El algoritmo JTMS puede gastar una cantidad enorme de recursos para encontrar una solución que satisfaga todas las justificaciones. Para determinar el soporte bien fundamentado de un nodo, se requiere un proceso de satisfacción de restricciones. Los procesos de detección de ciclos impares de justificaciones no monótonas o el retroceso dirigido son procesos que duran mucho, y en el transcurso de ellos algún nodo puede cambiar de IN a OUT y al revés varias veces.

- *Paso de nodos de OUT a IN* (*unonting*). A medida que avanza el proceso de razonamiento, se puede derivar un dato, a continuación retraerlo por producirse una contradicción y, finalmente, puede volver a estado IN si cambia el estado por una segunda contradicción. En estos casos, hay que tener mucho cuidado en la interfaz del MI con el TMS porque se podrían inferir datos que ya habían sido derivados.

Muchos de los problemas anteriores de un JTMS aparecen por la imposibilidad de referirse a los contextos en los que un nodo es creíble. Por ello, la base de conocimientos debe mantenerse libre de contradicciones, debe describir un único estado en cada momento y no puede ser etiquetada para ser reexaminada después. La razón fundamental de esto es que, cada dato se encuentra marcado con un conjunto de justificaciones que especifican cómo se deriva de otros nodos, pero que sólo describen el contexto en el que se sostiene dicho nodo de forma implícita.

### 3.1.7 Tipo de Sistema de Mantenimiento de la Verdad aplicado al diagnóstico cognitivo

Como ya se ha descrito, y dadas las características del DC, es imprescindible la aplicación de un mecanismo de revisión de creencias en este tipo de diagnóstico.

Dentro de los TMS, se han destacado los dos tipos más usados: basados en justificaciones y basados en hipótesis. Para poder decir cuál de ellos se adapta mejor al problema del DC, es necesario analizar algunas características distintivas de este tipo de diagnóstico [DOMI93]:

- El DC debe considerarse como un proceso interactivo que evoluciona a medida que avanza el aprendizaje del estudiante. El procedimiento de mantenimiento de la verdad (revisión de contradicciones) de los JTMS podría hacer inviable la idea de sistema interactivo en el momento en el que crece el número de justificaciones que se manejan, o cuando la reparación de una contradicción provoca otras llamadas al procedimiento de mantenimiento de la verdad por la necesidad de deshacer ciertas inferencias apoyadas en contradicciones (retroceso dirigido por las dependencias).

  Sin embargo, el ATMS no presenta ninguna restricción en cuanto al componente que realiza las inferencias, aportando una gran flexibilidad a la realización del sistema total (TMS y MI acoplado).

- En el razonamiento que, a diario, requerimos para resolver problemas de sentido común, utilizamos con toda naturalidad la idea del diagnóstico al buscar diferencias entre el modelo de un proceso u objeto que tenemos en la mente y los resultados que experimentamos en la realidad. Generalmente, comprobamos que en la mayoría de los casos son varias las causas que provocan los fallos, no una sola. Es necesario por lo tanto un mecanismo capaz de manejar múltiples elementos como causantes de los fallos. Por ello, los TMS basados en justificaciones (JTMS) quedan descartados. La idea básica de estos sistemas, según ya se ha visto, es la del estado admisible único; la red de dependencias del sistema debe presentar la combinación adecuada de nodos IN y OUT para que resulte un único estado admisible, es decir, que no presente contradicciones.

  En cambio, los TMS basados en hipótesis ó ATMS, poseen la capacidad de presentar diferentes estados admisibles en función de las distintas hipótesis que se tomen como contexto. De esta forma, se puede disponer de un mecanismos de revisión de creencias capaz de presentar múltiples fallos como posibles causantes de las anomalías observadas.

## 3.2 Diseño instruccional

El diseño instruccional describe el método que permite a los estudiantes alcanzar los objetivos de aprendizaje después de llevar a cabo un conjunto de actividades utilizando los recursos de un entorno (Amorim et al. (2006)).

En los últimos años, el auge de Internet ha impulsado la aparición de nuevas formas de aprendizaje, herramientas educativas y aplicaciones. En este ámbito, la necesidad de gestionar recursos reutilizables ha dirigido el desarrollo de diversas especificaciones de metadatos para representar el contenido del aprendizaje, recursos educativos y metodologías de diseño instruccional de tal modo que, los materiales y los diseños educativos realizados en una plataforma (hardware y software que soportan el entorno de aprendizaje, donde se llevarán a cabo las actividades formativas) puedan ser intercambiados con otras plataformas. En este aspecto, han jugado un papel fundamental los estándares, ya que se han convertido en la puerta que permite compartir tanto los contenidos como los diseños para el aprendizaje entre distintas plataformas compatibles con los estándares.

Las especificaciones para el diseño instruccional, conocidas como Lenguajes de Modelado Education (Educational Modelling Languages, EML), son modelos de información y agregación semántica que describen, desde un punto de vista pedagógico, el contenido así como las actividades educativas. Estos elementos están organizados en unidades de estudio con el objetivo de permitir su reutilización e interoperabilidad (posibilidad de intercambiar la información entre

sistemas con diferentes funcionalidades y objetivos). Además, los EMLs facilitan la descripción de aspectos pedagógicos que están relacionados con los Objetos de Aprendizaje (*Learning Objects* -LOs) en procesos educativos[1].

Algunas de las principales especificaciones EML son las siguientes:

- *CDF*[2]. Usa el Formato de Descripción de Cursos ARIADNE (A-CDF, Verbert and Duval (2004)). Un curso en A-CDF consisten en documentos XML junto con un generador de cursos LMS. Se centra fundamentalmente en el contenido y su agrupación, pero es suficientemente expresivo para describir el proceso de aprendizaje de acuerdo con un modelo pedagógico. El material didáctico que puede ser gestionado a través de CDF debe tener formato texto y usa un conjunto de herramientas (editores de curriculum, LMS -Sistemas de Gestión de Aprendizaje, en inglés *Learning Management System*, usados para la gestión y distribución de cursos a través de Internet, etc.), desarrolladas por el consorcio ARIADNE.

- *LMML*[3]. *Learning Material Mark-up Language* esta basado en un meta-modelo que puede ser usado en diferentes dominios de aplicación. Esta especificación utiliza XML para describir material de e-learning[4]. Comprende diversos módulos de material de aprendizaje que contienen, a su vez, otros sub-módulos. Está centrado en una estructura conceptual, modular y jerárquica de contenido e-learning. Además, LMML puede adaptarse a diferentes situaciones de aprendizaje y a diferentes estudiantes. Usa el concepto de *Curso* como unidad de estudio.

- *PALO*[5]. Es un lenguaje de modelado (Rodríguez-Artacho et al. (1999)) que describe cursos organizados en módulos que contienen actividades de aprendizaje, contenido, y un plan de enseñanza asociado. El lenguaje permite definir tipos de escenarios de aprendizaje con sus propiedades pedagógicas asociadas y se pueden establecer también la secuencia de módulos y tareas de aprendizaje, dependencias entre ellos, plazos, etc., según las restricciones del curso. Usa el concepto de *Módulo* como unidad de estudio.

- *Targeteam*[6]. Targeted Reuse and GEneration of TEAching Materials es un lenguaje que soporta la producción y gestión (uso y reutilización) de material de aprendizaje (Koch (2002)), que incluye notas y contenidos como explicaciones, motivación, y ejemplos, todos ellos interrelacionados. Usa un lenguaje basado en XML (TeachML), y utiliza el concepto de Tema como unidad de estudio. Además, permite el uso de material en diferente situaciones de aprendizaje y dominos pedagógicos (enseñanza primaria, enseñanza superior, etc.).

[1] Un objeto de aprendizaje se puede definir como cualquier recurso digital reutilizable que está encapsulado junto con otros objetos de aprendizaje en una lección. La lección puede estar compuesta de unidades, módulos, cursos o incluso programas, dependiendo de los objetos de aprendizaje que se pretenden abarcar (McGreal, 2004)

[2] http://www.ariadne-eu.org

[3] http://www.lmml.de

[4] El término e-learning (Carrasco (2000)) hace referencia, por una parte, al uso de tecnologías de Internet (-e), y por otra, a una metodología de transmisión de conocimientos y desarrollo de habilidades centrada en el sujeto que aprende y no tanto en el profesor que enseña. Según Holmes and Gardner (2006), la acepción más común de e-learning es *la enseñanza a través de Internet.*

[5] http://sensei.lsi.uned.es

[6] http://www.targeteam.net

- *IMS Learning Design* (IMS LD)[7]. Esta especificación es una integración del EML desarrollado por el OUNL (Open University of Netherlands), con otras especificaciones IMS existentes para el intercambio e interoperabilidad de material e-learning. El EML IMS describe la estructura y procesos educativos basado en un meta-modelo pedagógico, usando unidades de aprendizaje llamadas *Learning Design* (IMS (2003a)). IMS LD describe un método que está formado por un número de actividades de aprendizaje realizadas para alcanzar algunos objetivos educativos. Permite la combinación de diversas técnicas (tradicional, colaborativa, etc.), y facilita la descripción de otras. El IMS LD ha surgido como el estándar *de facto* para la representación de cualquier diseño instruccional que pueda estar basado en un amplio rango de técnicas pedagógicas.

A partir de las especificaciones propuestas, y dada la relevancia actualmente del IMS LD, a continuación se presenta, un poco más en detalle, esta especificación de diseño instruccional.

### 3.2.1 Especificación de Diseño Instruccional

La especificación de Diseño Instruccional (LD -del inglés *Learning Design*) de IMS es una especificación centrada en el proceso de aprendizaje, y no tanto en los contenidos finales (IMS (2003b)). Mediante IMS LD, los diseñadores instruccionales pueden representar un escenario de aprendizaje sustentado en cualquier teoría pedagógica, ya que es completamente neutra en este aspecto.

IMS LD se encuentra definida en cinco documentos: el modelo conceptual, el modelo de información, los esquemas XSD, la guía de implementación y los ejemplos.

La especificación puede asemejarse a un escenario de actuación (Koper and Olivier (2004)) donde:

- Las personas actúan con diferentes *roles* (profesor, tutor, alumno, supervisor, etc.).

- Los roles trabajan dirigidos por objetivos específicos mediante la ejecución de *actividades* de aprendizaje y/o soporte. Las actividades se pueden ensamblar en estructuras de actividades. Una estructura de actividades es la agregación de un conjunto de actividades relacionadas en una estructura individual que se puede asociar con un rol.

- Las actividades se realizan dentro de un *entorno*. Este entorno contiene los recursos y referencias a los recursos que se necesitan para llevar a cabo una actividad o un conjunto de actividades. Un entorno consta de objetos de aprendizaje, servicios complementarios, relación entre roles, agrupación de usuarios, etc.

- El rol que se adquiere en una actividad en un momento dado es determinado por un *método* o por una notificación. El método se diseña para proporcionar la coordinación de los roles, actividades y entornos asociados que permiten a los aprendices alcanzar los *objetivos* de aprendizaje. Un método puede contener condiciones (por ejemplo, reglas del tipo Ïf-Then-Else") que determinan la asignación de actividades y entornos a las personas y roles. Las condiciones se pueden usar para personalizar el diseño instruccional de los usuarios. Un ejemplo de esta personalización podría ser: *Si la persona tiene conocimiento previo del tema X, Entonces la actividad Y puede ser omitida en el proceso de aprendizaje.*

[7] http://www.imsglobal.org/learningdesign

Existen tres niveles de implementación que definen la estructura conceptual de la especificación IMS LD (Figura 3.5):

- *Nivel A.* Constituye el núcleo y comprende la definición de usuarios, actividades de aprendizaje, actividades de soporte, entornos, recursos, métodos, ejecuciones o plays, actos, roles y la coordinación entre todos ellos.
- *Nivel B.* Añade al nivel A la utilización de servicios de monitorización y elementos globales para la gestión de la especificación desde archivos externos a la misma. Constituye el nivel que aporta más flexibilidad en la representación didáctica porque permite ocultar y mostrar elementos, condicionar el flujo de aprendizaje, almacenar datos del usuario y la instancia (a nivel local o global).
- *Nivel C.* Añade al nivel B la utilización de notificaciones, es decir, mecanismos de lanzamiento y ejecución automática de procesos según el cumplimiento de ciertas condiciones o la ejecución de ciertas acciones de alguno de los usuarios implicados.

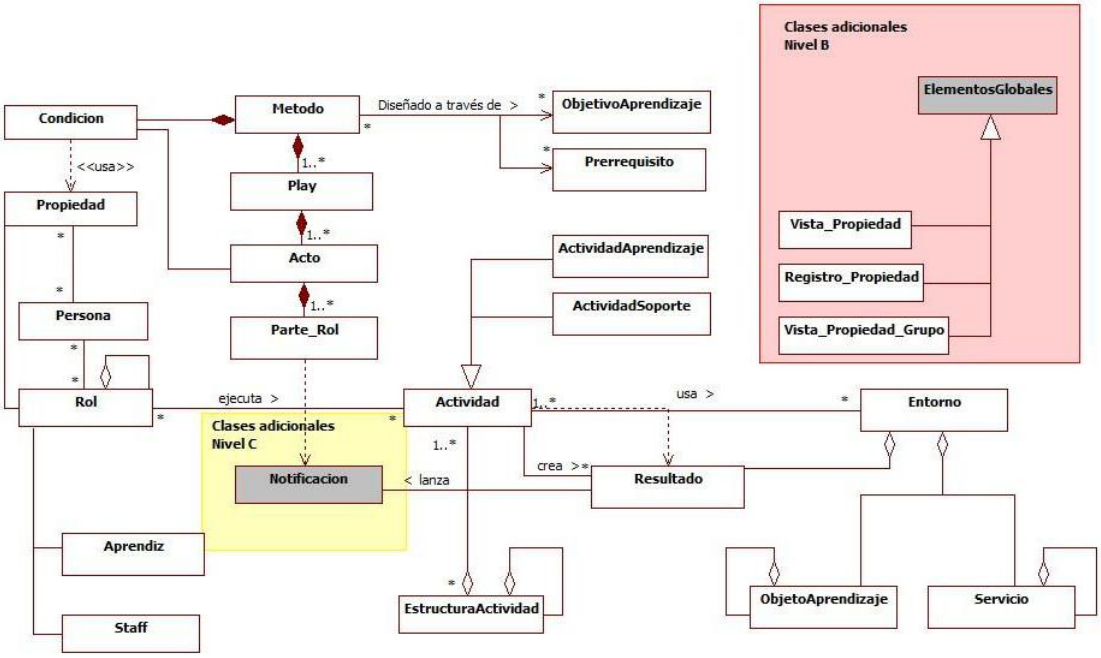


**Figura 3.5:** Modelo Conceptual de los niveles de implementación LD

### 3.2.2 Objetivos educativos

La forma en que se expresa qué logros educativos concretos se desean alcanzar en los alumnos (aprendices en general) es a través de los objetivos educativos, y deben expresarse en términos de conducta final del alumno (no del profesor).

Los objetivos educativos deberían constituir pues la base de programación de cualquier materia. Un programa, para ser consecuente, debe estar basado en los objetivos educativos que el

tutor pretende alcanzar de tal modo que, a partir de su formulación, se organicen el resto de factores a considerar en un determinado proceso educativo (actividades, medios, métodos de enseñanza/aprendizaje, medios y ayudas, evaluación, etc.).

Los fines (a dónde se pretende llegar) marcan las directrices generales que orientan la actividad educativa. Sin embargo, los fines se caracterizan por su amplitud y falta de concreción, por lo que deben especificarse a través de objetivos más concretos e inmediatos que deriven de esos fines delimitándolos.

Las principales cualidades que una buena formulación de objetivos educativos debe poseer son las siguientes:

- Ayuda y orientación al alumno en su aprendizaje.
- Orientación del profesor/tutor en su función educativa.
- Facilitar la elección de las estrategias y medios de enseñanza.
- Facilitar la construcción de pruebas e instrumentos de evaluación y la selección de criterios de calificación.

Por otra parte, cada objetivo educativo debe poseer las siguientes características fundamentales:

- Claro y preciso. Que todo el mundo entienda lo mismo y que transmita fielmente el comportamiento a lograr.
- Dirigido al alumno. Que esté redactado en términos del aprendizaje que el alumno debe lograr.
- Realista y alcanzable. No tiene sentido plantearse un objetivo que no se puede conseguir.
- Por escrito. Para evitar la deformación o distorsión del objetivo.
- Comunicado. Los objetivos deben ser comunicados a los alumnos (a los que aprenden).
- Unificado. Que busque un único comportamiento.

En el siguiente apartado se van a analizar diversas clasificaciones de los objetivos educativos (Sánchez (1995)).

### 3.2.3 Clasificación de los objetivos educativos

Los objetivos educativos se pueden clasificar en función de su amplitud, nivel de exigencia y formulación como se describe a continuación:

- Por su amplitud. Los objetivos educativos se clasifican a grandes rasgos en los siguientes tipos:
  - Objetivos generales. Expresan el comportamiento final que se desea obtener.
  - Objetivos específicos. Expresan los comportamientos intermedios para lograr el comportamiento final de los objetivos generales.

Cuanto más precisos sean los objetivos, el proceso de enseñanza-aprendizaje se podrá estructurar mejor. Sin embargo, la referencia para organizar la enseñanza hay que buscarla en los objetivos generales. Lo que se persigue no son comportamientos específicos sino procesos de aprendizaje que modifiquen las actitudes y habilidades de los individuos que aprenden.

Dentro de los objetivos generales puede haber varios niveles, según la generalidad y ámbito de su formulación. Por ejemplo, en el ámbito de la Universidad pueden definirse objetivos educativos en varios niveles: los objetivos de la Universidad española están definidos en la Ley de Reforma Universitaria pero, de ellos derivan los objetivos generales de los Centros, de éstos surgen los objetivos generales de los departamentos de los que derivan los objetivos generales de las asignaturas y así hasta llegar a los objetivos generales de las unidades didácticas a partir de las que se formulan los objetivos específicos. En cualquiera de estos niveles se podrían definir objetivos con un nivel de generalidad específico para cada uno de ellos.

- Por su nivel de exigencia, los objetivos se pueden clasificar en los siguientes tipos:

  - Mínimos u obligatorios, optativos. Son los objetivos que definen los conocimientos mínimos necesarios para pasar al nivel inmediato superior. Son objetivos imprescindibles y deben ser alcanzados por la totalidad de los alumnos que aprenden una determinada materia.
  - Optativos. Son los objetivos cuya finalidad es proporcionar al alumno la posibilidad de que elija libremente, de entre los objetivos optativos que se le ofrecen, aquéllos que se comprometerá a superar.
  - Libres. Son los objetivos elegidos libremente por que alumno, una vez superados los anteriores.

- Por su formulación, los objetivos se pueden clasificar en los siguientes tipos:

  - Genéricos. Son los objetivos que se formulan de manera amplia, con verbos que describen procesos, habilidades o actitudes internas del individuo, como comprender, entender, reconocer, valorar, apreciar, estimar, etc. Se corresponden con los objetivos generales.
  - Operativos. Son aquéllos que están expresados en términos de conductas observables y medibles. Para su formulación se utilizan verbos de acción como señalar, distinguir, aplicar, ordenar, clasificar, calcular, etc. Se corresponde con los objetivos específicos.

  La formulación operativa de objetivos tiene la ventaja de que proporciona un mayor rigor para el análisis de las actividades de enseñanza ya que facilita la evaluación de sus resultados. Sin embargo, hay determinadas materias en las que resulta difícil -y a veces no deseable-, llegar a definiciones operativos de los objetivos a lograr, como en las materias de áreas artísticas.

Seguidamente, se analiza la utilidad de las taxonomías en la formulación de objetivos educativos y se profundizará brevemente en las principales taxonomías propuestas en este campo haciendo especial hincapié en las usadas como uno de los soportes de este trabajo.

### 3.2.4 Taxonomías de objetivos educativos

El objetivo fundamental de las taxonomías (*conjunto de principios de clasificación o estructura*) de los objetivos educativos es la racionalización y sistematización del proceso educativo posibilitando así la estructuración coherente de los distintos niveles de aprendizaje. Como consecuencia, las taxonomías educativas son un soporte de ayuda eficaz para el tutor o profesor en la sistematización del proceso de formulación de los objetivos de aprendizaje a establecer en una determinada materia a enseñar.

En las taxonomías existentes se distinguen tres campos o dominios de aprendizaje interdependientes entre sí:

- Dominio cognitivo. Este dominio incluye los aprendizajes relacionados con la razón, conocimientos y saberes. Se corresponde con el recuerdo o reconocimiento de conocimiento y el desarrollo de habilidades y capacidades intelectuales por parte del alumno.
- Dominio afectivo. Es el dominio que incluye los aprendizajes referentes a sentimientos, emociones, valores, actitudes y voluntades.
- Dominio psicomotor. Está constituido por los aprendizajes relativos a los movimientos voluntarios del hombre, destrezas y habilidades.

Es imposible encontrar una taxonomía perfecta, que aúne todas las condiciones necesarias para la formulación y clasificación de objetivos en un amplio abanico de tipos de entornos de aprendizaje. Sin embargo, parece haber un consenso en la mayoría de la literatura al respecto que las más destacables son las taxonomías de objetivos educativos de Bloom et al. (1956), la de Gagne y Briggs [GAGN77] y la de Merril [MERR83]. Entre estas taxonomías también parece bastante unánime la idea de que la taxonomía de Bloom es considerada como el modelo más influyente al respecto en el mundo académico, entre educadores, profesores y entrenadores y por muchos es considerado casi un estándar.

A continuación, detallamos los dos dominios, cognitivo y afectivo, estudiados más en profundidad por Bloom y sus colaboradores.

### 3.2.5 Taxonomía de objetivos educativos de Bloom en el dominio cognitivo

Bloom y colaboradores enfocaron su estudio inicialmente en el dominio cognitivo (Bloom et al. (1956)) y fueron los precursores del estudio de las taxonomías educativas sirviendo, como ya hemos mencionado anteriormente, de base para la mayoría de los posteriores estudios en este área. A continuación, trataron en detalle el dominio afectivo (Krathwohl et al. (1973)) y, finalmente, diversos autores estudiaron el tercer dominio, el dominio psicomotor, que completase la citada taxonomía. Las tres versiones más referenciadas de objetivos en el dominio psicomotor parecen ser las taxonomías de Dave (Dave (1970)), de Simpson (Simpson (1972)) y Harrow (Harrow (1972)) que se detallan más adelante.

La taxonomía de Bloom no se fundamenta en una teoría determinada de la inteligencia ni de la personalidad, sino que es más bien un estudio básicamente empírico. Cada uno de los tres dominios en la taxonomía de Bloom, se basa en la premisa de que las categorías o niveles en todos ellos están ordenadas en grado de dificultad, es decir, dentro de cada uno de los tres dominios hay niveles en el desarrollo del aprendizaje y estos niveles aumentan en dificultad y deben ser dominados para pasar al siguiente.


**Figura 3.6:** Niveles de la taxonomía de objetivos educativos de Bloom

Según Bloom, el dominio cognitivo está formado por los seis niveles que aparecen en la figura 3.6:

Estos niveles no son mutuamente excluyentes. Los tres primeros niveles (conocimiento, comprensión y aplicación) se jerarquizan bien pero en los tres niveles restantes a veces resulta difícil clasificar un objetivo en un determinado nivel dada la complejidad de las tareas necesarias para conseguirlo. Un modelo adaptado fue presentado por dos de sus colaboradores, Anderson y Krathwohl (Anderson et al. (2001)) en el que, concretamente, se han invertido los niveles de síntesis y evaluación. A pesar de ello, el debate sobre el orden de los niveles continúa ya que depende del entorno concreto de aprendizaje y el criterio exacto establecido en ambos niveles para él (por ejemplo, en el desarrollo y entrenamiento industrial, el nivel de evaluación podría representar valoración estratégica y creación en la decisión así que, en este caso, los niveles superiores podrían invertirse).

- **Nivel 1: Conocimiento.** Incluye los objetivos relacionados con la mera adquisición de conocimientos. En este nivel, el alumno debe ser capaz de recordar conocimientos específicos y universales, clasificaciones, criterios, métodos, procesos, patrones, situaciones, teorías, estructuras, lugares, etc. En este caso, recordar es similar a tener en la mente el material apropiado. No se requiere en este nivel, generalmente, que el alumno varíe el material que se le ha ofrecido, sólo que lo registre, memorice y evoque la información recibida. A su vez, este nivel puede dividirse en distintos subniveles dependiendo de la naturaleza del conocimiento enseñado:

  - *Conocimiento de lo específico*: subnivel de aprendizaje en el que se encuentra el conocimiento de terminología y hechos. El material en este caso está en un nivel ínfimo de abstracción.
  - *Conocimiento de modos y medios para el tratamiento de lo específico*. Subnivel en el que se encuentra el conocimiento de la existencia de convenciones, tendencias, secuencias, clasificaciones, categorías, criterio y metodologías. Es un nivel medio de abstracción entre el conocimiento de lo específico y el conocimiento de lo universal. Se trata de conocer las técnicas y metodologías pero en absoluto de su aplicación. Se sigue tratando de una actitud pasiva por parte del alumno.
  - *Conocimiento de lo universal y lo abstracto*. Para alcanzar este tipo de objetivos, el alumno debe conocer los esquemas y patrones bajo los cuales están organizados las ideas y fenómenos en un campo, es decir, las estructuras, teoría y generalizaciones

que dominan un área y que son utilizadas para estudiar fenómenos o resolver problemas. Es el nivel más alto de abstracción y complejidad dentro del simple conocer.

Ejemplo de verbos: definir, nombrar, listar, escribir, expresar, etc.

Ejemplo de comportamiento: .El estudiante define los 6 niveles de la taxonomía de Bloom en el dominio cognitivo".

- **Nivel 2: Comprensión.** Este nivel integra los objetivos que desarrollan habilidades de asimilación de la información recibida. El alumno sabe lo que le está siendo comunicado y puede hacer uso de ello, pero no se requiere que lo relacione con otra información ni que vea todas sus implicaciones. Es importante destacar el proceso mental de organizar y reorganizar por parte del alumno el material aprendido, no basta simplemente con recordarlo. Se le pedirá que sea capaz de traducir la información recibida a su propio lenguaje, interpretarla o bien extrapolar más allá de los datos disponibles pero siempre facilitando al alumno toda la información necesaria para el logro del objetivo.

  Ejemplo de verbos: explicar, resumir, describir, ilustrar, interpretar, mostrar, etc.

  Ejemplo de comportamiento: .El estudiante explica el propósito de la taxonomía de Bloom en el dominio cognitivo".

- **Nivel 3: Aplicación.** Incluye los objetivos que persiguen la capacidad del sujeto para usar las abstracciones aprendidas en situaciones concretas y particulares. Las abstracciones pueden ser ideas generales, reglas de procedimiento o métodos generales, principios técnicos, así como ideas y teorías que deben ser recordados y aplicados por el alumno.

  Mientras en el nivel de Comprensión, al alumno se le facilitan todas las informaciones para la solución, en este nivel, él mismo debe aportar información complementaria.

  Ejemplo de verbos: usar, resolver, demostrar, aplicar, organizar, construir, etc.

  Ejemplo de comportamiento: .El estudiante construye un objetivo instruccional por cada nivel de la taxonomía de Bloom".

- **Nivel 4: Análisis.** La persona debe ser capaz de dividir la información recibida en sus elementos constitutivos o partes de forma que, la jerarquía de ideas quede clara y/o la relación entre las ideas expresadas se haga explícita.

  Ejemplo de verbos: diferenciar, discriminar, analizar, distinguir, identificar, relacionar, seleccionar, separar, etc.

  Ejemplo de comportamiento: .El estudiante compara y contrasta los dominios cognitivo y afectivo".

- **Nivel 5: Síntesis.** En este nivel, el alumno es capaz de juntar elementos y partes de modo que formen un todo que es nuevo para él. Implica el trabajo con partes, elementos, etc., así como la combinación de los mismos para constituir un patrón o estructura que no existía previamente.

  Existen varios niveles de composición de un todo: producción de una única comunicación, producción de un plan o conjunto de operaciones y, finalmente, derivación de un conjunto de relaciones abstractas que clasifiquen o expliquen ciertos datos o fenómenos. El proceso de síntesis implica el desarrollo de la capacidad creado del alumno, por lo que debe ser especialmente potenciado.

Ejemplo de verbos: categorizar, combinar, componer, diseñar, modificar, reordenar, reconstruir, reorganizar, revisar, crear, etc.

Ejemplo de comportamiento: .El estudiante diseña un esquema de clasificación para formular objetivos educativos que combinen los tres dominios: cognitivo, afectivo y psicomotor".

- **Nivel 6: Evaluación.** El objetivo educativo más alto que se puede alcanzar es la evaluación por parte del alumno del valor de materiales y métodos para un cierto propósito y la emisión de juicios cualitativos y cuantitativos de hasta qué punto los métodos satisfacen ciertos criterios. La evaluación puede realizarse en términos de evidencias internas o de criterios externos.

  Ejemplo de verbos: concluir, criticar, evaluar, justificar, juzgar, defender, etc.

  Ejemplo de comportamiento: .El estudiante juzga la eficiencia de formular objetivos educativos usando la taxonomía de Bloom".

### 3.2.6 Taxonomía de objetivos educativos de Krathwohl en el dominio afectivo

La taxonomía en el segundo dominio, el dominio afectivo, fue detallado por Krathwohl, Bloom y Masia (Krathwohl et al. (1973)). Este dominio incluye objetivos que describen cambios en las emociones y sentimientos, motivaciones, actitudes, interés, valores y desarrollo de apreciaciones por parte del alumno. Por lo tanto, el detalle del dominio afectivo, como los otros dominios, proporciona un marco para la enseñanza, entrenamiento, valoración y evaluación de la eficiencia en el entrenamiento y en el diseño de lecciones o actividades, etc.

Clasificar los objetivos en este área es extremadamente difícil ya que los objetivos no están planteados con precisión. Incluso, no existe unanimidad en cuanto a las experiencias educativas más apropiadas para estos objetivos. Además, se suma la dificultad de describir el comportamiento del alumno en estos casos.

A pesar de todo, es necesario considerar este dominio de objetivos educativos por su estrecha influencia con el dominio cognitivo así como por su especial interés en muchos entornos de aprendizaje (por ejemplo, entrenamiento de personas para ciertas profesiones de riesgo en las que, uno de los objetivos sea controlar el miedo o el estrés, o en preparación de personas en las que se desea despertar cierta actitud futura a través del aprendizaje, por ejemplo, a alumnos de una determinada asignatura).

Para Bloom y sus colaboradores, el dominio afectivo está compuesto de los siguientes cinco niveles o categorías:

- **Nivel 1: Recepción de fenómenos.** En este nivel se pretende sensibilizar al alumno en la existencia de ciertos fenómenos y estímulos. Es decir, se busca que el alumno quiera recibir fenómenos y participar en ellos (lecciones, charlas, etc.). Es, evidentemente, el primer paso crucial si se espera que el alumno aprenda lo que el profesor desea enseñar.

  Este nivel se ha dividido en tres categorías para indicar los tres grandes grados de interés que existen cuando el alumno asiste a un fenómeno. Tales subcategorías son continuas, sin un punto claro de división entre ellas.

  El grado de interés varía entre una posición extremadamente pasiva por parte del alumno, en la que el profesor tiene la responsabilidad de capturar la atención del alumno, hasta un punto en el cual el estudiante dirige su atención hacia estímulos elegidos previamente, a un nivel, como mínimo, semiconsciente:

- *Percatarse*: Se trata prácticamente de un comportamiento cognitivo, pero al contrario que el conocimiento (nivel inferior en la taxonomía cognitiva) no se relaciona con la memoria sino con tener en cuenta una situación, un fenómeno, un estado de cosas, etc. No implica atención, se trata simplemente de darse cuenta de algo, sin discriminación o reconocimiento.
- *Deseo de recepción*. Todavía este escalón pertenece a lo cognitivo. En este nivel se representa la permisividad de tolerar los estímulos, es decir, no evitarlos. Sigue tratándose de neutralidad ante ellos. En el mejor de los casos, el alumno desea tener noticias de ciertos fenómenos y prestarles su atención.
- *Atención controlada o selectiva*. En este momento aparece un nuevo fenómeno en el alumno, la diferenciación de ciertos estímulos a un nivel consciente (o semiconsciente). Existe un control por parte del alumno de la atención, de forma que los estímulos favorecidos son seleccionados frente a estímulos que puedan distraer o competir por su atención (denominado *inhibición*).

Ejemplo de verbos: seguir, mantener, localizar, apuntar a, seleccionar, preguntar, etc.

Ejemplo de comportamiento: “El estudiante escucha y recuerda el nombre de ciertas personas”.

- **Nivel 2: Respuesta a fenómenos.** En este nivel se trata con respuestas del alumno que van más allá de la mera atención a un fenómeno. El estudiante está lo suficientemente motivado, de modo que no sólo desea atender sino que, se puede decir, atiende activamente. El alumno se compromete por sí mismo en pequeña medida con el fenómeno en el que está envuelto. Se trata de un nivel muy bajo de compromiso, no se trata de una cierta actitud por su parte ni tampoco de una valoración. Se trata de que el alumno está haciendo algo más que meramente percibir el fenómeno. Esta categoría se puede identificar con lo que, comúnmente, se denomina por los profesores "mostrar interés"por parte del alumno.

  Ejemplo de verbos: responder, asistir, ayudar, discutir, presentar, recitar, informar, decir, obedecer, etc.

  Ejemplo de comportamiento: “El estudiante participa en discusiones en una clase”.

- **Nivel 3: Valoración.** Es la única categoría donde se usa un término ampliamente usado en las expresiones de objetivos. Se trata del significado común: una cosa, fenómeno o comportamiento vale la pena. Este concepto abstracto de valor puede ser el resultado de la valoración llevada a cabo por el individuo, pero también puede ser un producto social que ha sido lentamente interiorizado o aceptado, llegando a ser usado por el propio estudiante como su criterio de valoración. El comportamiento del alumno en este punto es lo suficientemente consistente como para considerar que posee tal valor.

  La consciencia del individuo se ha desarrollado hasta el punto de tomar control activo de su comportamiento. Además, tal actitud no ha sido motivada por obediencia, sino por el compromiso del alumno al valor subyacente que guía su comportamiento. Tal actitud puede tener tres niveles:

  - Aceptación de un valor. En este nivel, el alumno considera que un cierto fenómeno, comportamiento, objeto, etc., vale la pena. No obstante, tal merecimiento se da con un grado mínimo de convicción, es decir, existe la posibilidad de que el estudiante reconsidere su posición.

- Preferencia por un valor. Los comportamientos en este nivel implican no sólo la aceptación de un valor hasta el punto de desear ser identificado con él, sino que el individuo está lo suficientemente comprometido con tal valor como para que lo busque y lo quiera.
- Compromiso. “Creer” a este nivel conlleva un alto grado de convicción. La lealtad a una posición, un grupo o una causa estarían incluidas también en este punto de la clasificación. La persona con comportamientos en este nivel es percibida claramente por los demás como poseedora del valor en cuestión. Además, dependiendo de su personalidad, actuará de modo que aquello en lo que cree se desarrolle, crezca y se extienda; trata de convencer a otros y convertirlos a su causa.

Ejemplo de verbos: explicar, seguir, formar, invitar, justificar, demostrar, compartir, proponer, trabajar, estudiar, etc.

Ejemplo de comportamiento: “El estudiante propone un plan de mejora social y perseguirlo con compromiso”.

- **Nivel 4: Organización.** A medida que el estudiante interioriza los valores, encuentra situaciones en las que puede ser relevante más de un valor. Por tanto, aparece la necesidad de organizar sus valores en un sistema, determinar las interrelaciones entre ellos y establecer los valores permanentes. Tal sistema se construye de modo gradual, estando sujeto a cambios a medida que son incorporados nuevos valores.

  Esta categoría está subdividida en dos niveles, dado que un prerrequisito para interrelacionar los valores es la conceptualización del valor en una forma que permita su organización. La conceptualización es, por tanto, la primera tarea en el proceso de organizar. La segunda tarea es la organización por prioridades de un sistema de valores.

  Ejemplo de verbos: alterar, ordenar, combinar, comparar, completar, organizar, preparar, adherirse a, sintetizar, identificar (valores).

  Ejemplo de comportamiento: “El estudiante reconoce la necesidad de equilibrio entre comportamiento libre y responsable”.

- **Nivel 5: Caracterización.** En este nivel de interiorización, los valores tienen ya un lugar en la jerarquía de valores del individuo, están organizados de algún modo en un sistema consistente y han controlado el comportamiento del individuo durante un tiempo, de modo que éste ha aceptado el comportarse de ese modo. Se llega a este nivel cuando el individuo tiene un sistema de valores que ha marcado su conducta durante un tiempo suficientemente largo como para haber creado un determinado estilo de vida. Raramente, los objetivos educativos alcanzan los niveles 4 y 5 de la clasificación afectiva. La madurez e integración personal requeridas para estos niveles las alcanza generalmente el individuo en etapas avanzadas de su vida. El tiempo y la experiencia deben interactuar con el aprendizaje cognitivo y afectivo antes de que el individuo pueda contestar a las preguntas cruciales de “¿Quién soy yo?”, “¿Para qué estoy aquí?”.

  Ejemplo de verbos: revisar, proponer, modificar, actuar, discriminar, practicar, etc.

  Ejemplo de comportamiento: .El estudiante revisa los juicios y cambia de comportamiento a la luz de nuevas evidencias".

A continuación, vamos a detallar brevemente algunas de las taxonomías de objetivos educativos en el dominio psicomotor más conocidas y cuál de ellas se ha considerado de mayor interés en el desarrollo de este trabajo dadas sus características.

### 3.2.7 Taxonomía de objetivos educativos en el dominio psicomotor

El dominio psicomotor incluye movimiento físico, coordinación y el uso en areas relacionadas con habilidades motoras. El desarrollo de estas habilidades requieren práctica y se miden en términos de velocidad, precisión, distancia, procedimientos, o técnicas en ejecución. Actualmente, las habilidades "motoras""se extienden más allá de las destrezas manuales o físicas que tradicionalmente se han considerado, cubriendo y estando relacionadas también, por ejemplo, con técnicas o destrezas en el mundo de las comunicaciones, hablar en público, etc. Si se desea incluir una taxonomía de objetivos educativos lo más amplia posible, es necesario por lo tanto incluir el dominio psicomotor, sobre todo, si se desea una taxonomía de objetivos que abarque aprendizajes en entornos de entrenamiento o formación.

Hay tres taxonomías de objetivos educativos en el dominio psicomotor más destacadas: la taxonomía de Dave, la taxonomía de Harrow y la taxonomía de Simpson que se describen a continuación:

#### 3.2.7.1 Taxonomía de objetivos educativos de Dave

Dave, estudiante de Bloom, en 1967, presentó un modelo de formulación de objetivos educativos caracterizado por los cinco niveles siguientes(Dave (1970)):

- **Nivel 1: Imitación.** Observar una habilidad e intentar repetirla, o ver un producto acabado e intentar replicarlo de tal modo que la ejecución puede ser de baja calidad.

  Ejemplo de verbos: duplicar, copiar, imitar, hacer mímica, repetir, etc.

  Ejemplo de comportamiento: "Observar a un profesor o entrenador y repite la acción, proceso o actividad".

- **Nivel 2: Manipulación.** Reproducir la habilidad o un determinado producto de forma reconocible a partir instrucciones o la memoria más que a través de la observación.

  Ejemplo de verbos: producir, completar, ejecutar, implementar, etc.

  Ejemplo de comportamiento: "Crear un trabajo propio tras recibir lecciones o leyendo sobre ello".

- **Nivel 3: Precision.** Realizar una cierta habilidad o producir un determinado producto independientemente, con fiabilidad, proporción, y exactitud de tal modo que pocos errores sean aparentes.

  Ejemplo de verbos: alcanzar (automáticamente), dominar, refinar, perfeccionar, etc.

  Ejemplo de comportamiento: "Realizar cualquier tarea o actividad con experiencia y alta calidad sin asistencia o instrucción siendo capaz de demostrar la actividad a otros estudiantes".

- **Nivel 4: Articulación.** Coordinar o combinar un conjunto de habilidades o acciones con armonía y consistencia interna para satisfacer un objetivo no estándar.

Ejemplo de verbos: construir, combinar, coordinar, integrar, formular, desarrollar, modificar, adaptar, etc.

Ejemplo de comportamiento: “Producir un video que integre música, drama, color, sonido, etc.”.

- **Nivel 5: Naturalización.** Dominar de forma automatizada e inconsciente una cierta actividad y habilidades o destrezas relacionadas con el nivel estratégico.

  Ejemplo de verbos: diseñar, especificar, dirigir, etc.

  Ejemplo de comportamiento: “Un jugador profesional juega un partido de baloncesto”

#### 3.2.7.2 Taxonomía de objetivos educativos de Simpson

La taxonomía de objetivos educativos en el dominio psicomotor presentada por Elizabeth Simpson (Simpson (1972)) se diferencia de la taxonomía de Dave principalmente en que incluye dos categorías adicionales previas a la imitación inicial o etapa de copia. A continuación se describen brevemente los siete niveles de que consta:

- **Nivel 1: Percepción.** Nivel que implica la capacidad de usar señales sensoriales para guiar la actividad motora. Abarca desde estimulación sensorial, pasando por selección de señales, hasta traducción.

  Ejemplo de verbos: elegir, describir, detectar, diferenciar, distinguir, identificar, aislar, relacionar -señales sensoriales, escuchar, sentir, tocar, etc.

  Ejemplos de comportamiento: “Ajustar el calor de la cocina a una temperatura correcta usando el olfato y probando la comida”,“Detectar señales de comunicación no verbal”.

- **Nivel 2: Preparar.** Este nivel consiste en la disposición para actuar. Incluye el prepararse mental, física y emocionalmente. Estas tres son disposiciones que predeterminan la respuesta de una persona a diferentes situaciones (algunas veces llamadas preparaciones mentales). Está estrechamente relacionado con la subdivisión en el dominio Afectivo “Respuesta a fenómenos”.

  Ejemplo de verbos: comenzar, reaccionar, ofrecerse voluntario, mostrar, conocer, etc.

  Ejemplos de comportamiento: “Conocer y actuar sobre una secuencia de pasos en un proceso de fabricación”. “Reconocer las capacidades y limitaciones de uno”. “Mostrar deseos de aprender un nuevo proceso”.

- **Nivel 3: Respuesta guiada.** Es la etapa inicial en el aprendizaje de una habilidad compleja. Incluye imitación, prueba y error. La adecuación para la ejecución se alcanza mediante la práctica.

  Ejemplo de verbos: copiar, reproducir, seguir, etc.

  Ejemplos de comportamiento: “Seguir las instrucciones para construir un modelo”. “Realizar una ecuación matemática como se demostró”.

- **Nivel 4: Mecanismo.** Es la fase intermedia en el aprendizaje de una habilidad compleja. Las respuestas aprendidas llegan a ser habituales y los movimientos pueden ejecutarse con cierta confianza y pericia.

  Ejemplo de verbos: construir, manipular, medir, organizar, fijar, calibrar, juntar, etc.

Ejemplos de comportamiento: “Usar un ordenador personal”. “Conducir un coche”

- **Nivel 5: Respuesta compleja.** La ejecución hábil de actos motores que implican modelos de movimiento complejos. La habilidad viene indicada por una ejecución rápida, precisa, y altamente coordinada y que requiere un mínimo de energía. Esta categoría incluye la ejecución sin dudas, y automáticamente.

  Ejemplo de verbos: construir, comer, manipular, medir, organizar, arreglar, mostrar, etc.

  Ejemplos de comportamiento: “Maniobrar con el coche en una plaza de aparcamiento muy reducida”. “Mostrar competencia tocando el piano”.

- **Nivel 6: Adaptación.** Las habilidades son desarrolladas bien y el individuo puede modificar patrones de movimiento para adecuarlos a requerimientos especiales.

  Ejemplo de verbos: adaptar, alterar, cambiar, reordenar, reorganizar, revisar, variar, etc.

  Ejemplos de comportamiento: “Responder a experiencias inesperadas”. “Modificar instrucciones para satisfacer las necesidades de los estudiantes”.

- **Nivel 7: Creación.** Crear nuevos patrones de movimiento para adaptarse a una situación particular o a un problema específico. Los resultados del aprendizaje enfatizan la creatividad basada en habilidades muy bien desarrolladas.

  Ejemplo de verbos: componer, crear, construir, componer, diseñar, etc.

  Ejemplos de comportamiento: “Construir una nueva teoría”. “Crear una nueva rutina de gimnasia”.

#### 3.2.7.3 Taxonomía de objetivos educativos de Harrow

La interpretación de Harrow del dominio psicomotor (Harrow (1972)) tiende fuertemente hacia el desarrollo de una buena forma física, destreza, agilidad, control del cuerpo, para lograr un considerable nivel de pericia (*expertise*) y es la única de las tres taxonomías descritas a nivel psicomotor que implica especialmente influencia emocional sobre otros en el nivel más experto de control corporal.

Los niveles que presenta esta taxonomía son los siguientes:

- **Nivel 1: Reflejo.** Este nivel incluye reacciones que no son aprendidas, reacciones involuntarias.

  Ejemplo de verbos: reaccionar, responder, etc.

  Ejemplo de comportamiento: “Responder físicamente de modo instinto”.

- **Nivel 2: Movimientos básicos fundamentales.** Este nivel incluye movimientos individuales básicos o simples, locomotores, no locomotores o manipulativos.

  Ejemplo de verbos: saltar, andar, masticar, sujetar, tocar, etc.

  Ejemplo de comportamiento: “Agarrarse a un objeto”

- **Nivel 3: Capacidades perceptuales.** Nivel que implica la capacidad de reconocer, recibir y clasificar entradas sensoriales; discernir que algo está sucediendo y determinar qué es. En definitiva se trata de usar más de una capacidad en respuesta a diferentes percepciones sensoriales.

Ejemplo de verbos: explorar, escribir, atrapar, etc.

Ejemplo de comportamiento: "Ver a un bailarín y reconocer que se está moviendo sincronizado con una cierta música que está siendo oída".

- **Nivel 4: Habilidades físicas.** Adquirir la dureza, resistencia, agilidad, control o fuerza física para completar una acción para completar un cierto comportamiento.

  Ejemplo de verbos: resistir, mantenerse, mejorar, moverse rápidamente, soportar, etc.

  Ejemplo de comportamiento: "Levantar un cierto peso".

- **Nivel 5: Movimientos especializados.** Este nivel incluye operaciones complejas. Un conjunto de movimientos planificados, iniciados y completados por una persona, que demuestran un nivel avanzado de habilidad dentro de una actividad psicomotriz.

  Ejemplo de verbos: conducir, construir, tocar un instrumento musical, nadar, hacer juegos malabares, etc.

  Ejemplo de comportamiento: "Actuar improvisando".

- **Nivel 6: Comunicación no verbal.** Este nivel incluye la comunicación a través de movimientos sin usar palabras o lenguajes formales incluyendo lenguajes no orales gesticulados. Se trata pues de expresar sentimientos y significado a través de movimientos y acciones (gestos, expresiones faciales, etc.).

  Ejemplo de verbos: componer, crear, interpretar, etc.

  Ejemplo de comportamiento: "Expresar algo con mímica".

Cada uno de estos tres modelos presentados a nivel psicomotor pueden ser más adecuados para ser aplicados en entornos de aprendizaje específicos. El modelo de Dave es el más sencillo desde el punto de vista de la autora de este trabajo. Sin embargo, dado que nuestro objetivo es aplicar la taxonomía de conocimientos del estudiante, en general, a diversos entornos de aprendizaje, no a un cierto tipo de ellos, se ha encontrado más útil los dos últimos modelos, el de Simpson y el de Harrow. Ambos están más enfocados al aprendizaje de personas que desean desarrollar alguna habilidad física que requiera, por ejemplo, especial atención en la percepción sensorial o su propia preparación mental, emocional, y física e incluso cuyo fin principal sea desarrollar habilidades que impliquen expresión, sentimiento y emoción. Así, la taxonomía de Simpson podría ser particularmente útil, por ejemplo, en entornos de formación o entrenamiento de personas en situaciones de conflicto, peligrosidad, estrés, etc., tales como entrenamiento de bomberos, paracaidistas, personal de una central nuclear, o para entrenamiento en situaciones que requieran tareas de alta resistencia física, por ejemplo, atletas. En cuanto a la taxonomía de Harrow, hay que destacar que es la única de las tres taxonomías citadas que implica, además, una influencia emocional sobre otros en el nivel más experto de control corporal (nivel 6). Por este motivo, esta taxonomía es especialmente útil si el objetivo del aprendizaje es desarrollar habilidades para expresar, comunicar y/o influir en los sentimientos, emociones de los demás a través del movimiento, lenguaje corporal, etc. Por ejemplo, aprendizaje de bailarines, aprendizaje para hablar en público, etc. Por estas razones, la taxonomía de Harrow ha sido elegida en esta trabajo como el modelo de objetivos a nivel psicomotor más extenso y es el que, como se describirá más adelante, será soporte de parte de la taxonomía de conocimientos del estudiante reflejada en la ontología del ME propuesta.

Cabe destacar, no obstante, que las tres taxonomías seleccionadas en los tres dominios -cognitivo, afectivo y psicomotor-, han sido utilizadas para definir los objetivos en la ontología de Modelado del Estudiante propuesta, aun cuando no se ha profundizado en ello por quedar fuera del ámbito estricto de esta tesis; aun con todo ello, y considerando cómo se ha diseñado dicha ontología de Modelado del Estudiante (descrita en la sección 5.2), resultará sencillo añadir nuevas taxonomías -incluso definir una jerarquía de ellas- que brinden, al diseñador de un curso, un mayor abanico de posibilidades a la hora de seleccionar, de entre todas ellas, las que considere más adecuadas, valorando, al tiempo, sus posibles interdependencias.

## 3.3 Ontologías

La palabra ontología tiene su origen en la Filosofía, en la que significa una explicación sistemática de la existencia. En la literatura se han propuesto muchas definiciones de ontología en sentido filosófico y, desde los 90's, con especial relevancia en la comunidad de Ingeniería del Conocimiento. A continuación, se analizan algunas de estas definiciones, haciendo especial hincapié en las que se consideran más claras y más exactas.

Algunos autores como Neches et al. (Neches et al. (1991)), , Gruber (1993), Borst (1997), etc.) han definido el concepto de ontología haciendo especial referencia a qué define una ontología y su objetivo. De este grupo de definiciones, se ha seleccionado la de Studer, et al. (Studer et al. (1998)):

*Una Ontología es una especificación formal y explícita de una conceptualización compartida. Conceptualización se refiere a un modelo abstracto de algún fenómeno en el mundo en el que se han identificado los conceptos relevantes. Explícita significa que el tipo de conceptos usados y las restricciones en su uso están explícitamente definidos. Formal se refiere al hecho de que una ontología debería ser procesable por una máquina. Compartida refleja la noción de que una ontología captura conocimiento consensuado, esto significa que no es privado de un individuo sino aceptado por un grupo.*

Hay otro grupo de definiciones que destacan por la interrelación que se establece entre ontologías y bases de conocimiento. Por ejemplo, la definición proporcionada por Bernaras y colegas (Bernaras et al. (1996), Swartout y colegas (Swartout et al. (1997)), etc.). En este grupo, destacamos la siguiente definición aportada por Gruber (Gruber (2008)):

*En el contexto de la ciencia de la computación y de la información, una ontología define un conjunto de primitivas de modelado con las cuales se puede modelar un dominio de conocimiento o de discurso. Las primitivas de modelado son típicamente clases (o conjuntos), atributos(o propiedades), y relaciones (o relaciones entre miembros de clases). Las definiciones de las primitivas de modelado incluyen información acerca de su significado y restricciones en su aplicación consistente y lógica. En el contexto de los sistemas de bases de datos, una ontología puede ser vista como un nivel de abstracción de los modelos de datos, análoga a los modelos jerárquicos y relacionales, pero con el objetivo de modelar conocimiento acerca de individuos, sus atributos y sus relaciones con otros individuos. Las ontologías son especificadas típicamente en lenguajes que permiten abstracción de la estructura de los datos y las estrategias de implementación; en la práctica, los lenguajes de ontologías son más cercanos en poder de expresividad a la lógica de primer orden que los lenguajes usados para modelar bases de datos. Por esta razón, las ontologías se dicen que están en nivel semántico, mientras que los esquemas de bases datos son modelos de datos a nivel lógico o físico. Debido a su independencia de los niveles más bajos de los modelos de datos, las ontologías son usadas para integrar bases de datos heterogéneas, habilitando la interoperativi-*

*dad entre sistemas diferentes y para especificar interfaces entre servicios independientes basados en conocimiento.*

Las ontologías son ampliamente usadas para diferentes propósitos (procesamiento de lenguaje natural, gestión del conocimiento, Web Semántica, comercio electrónico, etc.) y en diversas comunidades como ingeniería del conocimiento, ingeniería del software, bases de datos, etc., Uschold y Jasper (Uschold and Jasper (1999)) proporcionaron una nueva definición de la palabra ontología:

*Una ontología puede tomar una variedad de formas, pero incluirá necesariamente un vocabulario de términos y alguna especificación de su significado. Esto incluye definiciones y una indicación de cómo los conceptos están inter-relacionados que colectivamente impone una estructura sobre el dominio y restringe las posibles interpretaciones de los términos.*

Sin embargo, a nuestro juicio, la definición anterior adolece de falta de claridad en la medida en que no acota de manera explícita la forma que puede adoptar una ontología.

La amplia difusión de las ontologías en diferentes comunidades ha dado lugar a que, a veces, la noción de ontología se identifique con la de taxonomía y se considere a éstas últimas ontologías, en el más amplio sentido. Por ejemplo, una taxonomía para búsqueda en la Web es considerada una ontología. En este aspecto, la comunidad ontológica distingue dos tipos de ontologías. Las ontologías *lighweight*, que son esencialmente taxonomías de las ontologías y las ontologías *heavyweight*, que modelan más ampliamente el dominio, proporcionando más restricciones sobre las semánticas de dominio. Las ontologías *lighweight* incluyen conceptos, taxonomías de conceptos, relaciones entre conceptos, y propiedades que describen conceptos mientras que las ontologías *heavyweight* añaden axiomas y restricciones a las anteriores.

Las ontologías, en general, ayudan a capturar conocimiento *consensuado* de una forma genérica, pueden ser reutilizadas y compartidas por diferentes aplicaciones software y por diferentes grupos de personas, normalmente localizados en diferentes lugares. Por ejemplo permiten establecer marcos de trabajo uniformes dentro de organizaciones para reducir la confusión conceptual y terminológica. También sirven para lograr la interoperatividad entre paradigmas, lenguajes y herramientas de programación y métodos de modelado. En este último contexto, sirve como un lenguaje intermedio desde y hacia donde se pueden realizar mapeos de vocabularios propios y compartidos.

### 3.3.1 Componentes

Una ontología consta de: un conjunto no vacío de conceptos identificados como entidades relevantes en el dominio a modelar; un conjunto de relaciones; un conjunto de atributos que describen los conceptos y que pueden ser propios o heredados en una especialización; y un conjunto de axiomas que vinculan elementos de la ontología con condiciones que siempre deben ser satisfechas.

- **Conceptos** Un concepto puede ser cualquier cosa acerca de la cual se pueda aseverar algo. Por lo tanto, puede ser un objeto físico, la descripción de una tarea, función, acción, estrategia, etc. Cada concepto tiene asociado un término como nombre y un conjunto de atributos que lo identifican.

- **Relaciones** Se establecen para representar el tipo de interacción entre los conceptos de una determinada parte del mundo real. Formalmente, se definen a partir de un conjunto finito de dominios DOM = {D1, D2, ..., Dn}, siendo cada dominio un conjunto finito o

infinito de símbolos. La Relación se define formalmente como un subconjunto del producto cartesiano de los dominios: DOM : R ⊆ D1xD2x...xDn. Ejemplos: relaciones binarias como *subclase-de* (IS-A) o *es-parte-de* (PART-OF), relaciones ontológicas como las relaciones temporales, que implican precedencia en el tiempo, o las topológicas, que implican conexión espacial entre conceptos.

- **Funciones** Son un caso especial de relaciones donde el enésimo elemento de la relación es único para los n-1 anteriores. Formalmente, las funciones se definen como:

  F : C1xC2x...xCn – 1fiCn.

  Ejemplos de funciones son las relaciones siguientes: *Madre-de* y *Precio-de-un-auto-usado* (calcula el precio de un auto usado en función de elementos como modelo, fecha de fabricación y cantidad de kilómetros recorridos).

- **Axiomas** Un axioma define restricciones adicionales que complementan las definiciones de los componentes de la ontología, o limitan el uso que se puede hacer de ellos (por ejemplo: *Si A y B son de la clase C, entonces A no es subclase de B* o la fórmula: F = m ∗ a, que debe cumplirse siempre entre los atributos *F* -fuerza-; *m* -masa-; y *a* -aceleración- de un determinado concepto).

- **Instancias** Representan elementos del dominio de la ontología.

### 3.3.2 Lenguajes ontológicos

Existen diferentes lenguajes de ontologías que han sido creados durante las últimas décadas y otros lenguajes y sistemas de representación de los conocimientos (RCs) se han usado para implementar ontologías pero que no han sido creados específicamente para este propósito. En primer lugar, se describen los lenguajes ontológicos más representativos y posteriormente, se describe una aproximación informal usada para comparar sus representación y/o capacidades de razonamiento. Con esta aproximación, finalmente se muestra una comparativa de los principales lenguajes ontológicos.

#### 3.3.2.1 Lenguajes tradicionales

A comienzo de los 90's, se crearon un conjunto de lenguajes de ontologías basados en IA. Normalmente, los paradigmas de Representación del Conocimiento (RC) subyacentes a estos lenguajes estaban basados en lógica de primer orden (por ejemplo, KIF), en marcos combinados con lógica de primer orden (como CycL, Ontolingua, OCML y FLogic) o en lógica de descripción (por ejemplo, LOOM). También se creó OKBC, un protocolo de acceso a ontologías implementado en diferentes lenguajes con un paradigma de RCs basado en marcos.

De los lenguajes anteriores, el primero en crearse fue CycL (Lenat and Guha (1989)) y posteriormente, en 1992, KIF (Geneserth and Fikes (1992)), con extensiones para tratar con conjuntos, números, etc. Sin embargo, las ontologías eran difíciles de crear directamente en KIF por lo que se creó por encima de él Ontolingua (Farquhar et al. (1997)) con una sintaxis similar a Lisp. Este último lenguaje fue considerado un estándar por la comunidad ontológica en los años 90.

LOOM (MacGregor (1991)) fue también construido en este periodo. Fue creado para implementar BCs en general, no específicamente para ontologías. Este lenguaje está basado en lógica descriptiva (LD) y reglas de producción, con características de clasificación automática de conceptos. OCML se desarrolló más tarde (Motta (1999)), en 1993, como una especie de .ºntolingua

operacional"(muchas de las definiciones de OCML son similares a las correspondientes definiciones en Ontolingua). OCML se construyó para desarrollar ontologías ejecutables y modelos en métodos de resolución de problemas (MRPs). Finalmente, apareció FLogic (Kifer et al. (1995)) dotado de un motor de inferencias que permite comprobar el cumplimiento de las restricciones y deducir conocimiento.

En 1997, comenzó el programa *H*igh *P*erformance *K*nowledge *B*ase (HPKB), promovido por DARPA. El objetivo de este programa era resolver los problemas fundamentales que surgían cuando se trataban grandes BCs. Uno de sus resultados fue el desarrollo del protocolo OKBC (*O*pen *K*nowledge *B*ase *C*onnectivity, Chaudhri et al. (1998)). Se trata de un protocolo para intercambio de conocimiento, que permite acceder a BCs almacenadas en diferentes Sistemas de RC basados, a su vez, en diferentes paradigmas de RC. De los lenguajes mencionados, CycL, LOOM y Ontolingua son compatibles con OKBC.

XML (E*x*tensible *M*arkup *L*anguage) es la base de todos los lenguajes de este ámbito. Este lenguaje proporciona la sintaxis para construir documentos estructurados, pero no impone ninguna restricción semántica. Por este motivo, no se puede considerar un lenguaje de ontologías.

Mediante los espacios de nombres XML (*XML namespaces*), es posible identificar unívocamente los elementos de los documentos basados en él. Se basa en la asignación de un espacio de nombres único al documento en cuestión (una *URI*), así como en la asignación de una referencia relativa a cada elemento única en el contexto de ese documento. Esto permite reutilizar los elementos de los documentos a lo largo de toda la Web y permite la creación de documentos únicos (incluyendo las ontologías definidas por el resto de lenguajes) con ubicación distribuida.

Los primeros lenguajes de ontologías en este entorno aparecieron con el objetivo de proporcionar metainformación para los distintos elementos de información de los documentos web (información referente al idioma empleado, autor, documentos relacionados, etc.). Entre ellos, podemos nombrar a SHOE (*S*imple *H*TML *O*ntology *E*xtensions), que apareció en 1996 (Luke and Heflin (2000)). Este lenguaje define extensiones para incluir anotaciones en HTML, XOL (*X*ML-base *O*ntology exchange *L*anguage), que trata de especificar un subconjunto de las primitivas de OKBC y OML (*O*ntology *M*arkup *L*anguage), basado en lógica descriptiva de primer orden y grafos conceptuales.

#### 3.3.2.2 Lenguajes de la Web Semántica

En el entorno de la Web Semántica (Berners-Lee et al.) existen numerosos grupos de investigación estudiando las ventajas de trabajar con lenguajes de ontologías para explotar las características de la Web y se han desarrollado numerosas herramientas para edición de ontologías, validación (comprobación de integridad sintáctica y semántica), inferencia de conocimiento, realización de consultas, etc. Estos avances en materia de ontologías en la Web Semántica han llevado a la aparición de diferentes propuestas (*member submissions*) y recomendaciones por parte de la W3C (the *W*orld *W*ide *W*eb *C*onsortium).

Los lenguajes de ontologías son normalmente llamados lenguajes de ontologías basados en web o lenguajes de marcado de ontologías. Su sintaxis se basa en lenguajes de marcado existentes tales como HTML (Raggett et al. (1999)) y XML (Bray et al. (2004)), cuyo propósito no es el desarrollo de ontologías sino la presentación e intercambio de datos. La interrelación entre estos lenguajes se muestra en la Figura 3.7.

El primer lenguaje de marcado de ontologías es SHOE. Es un lenguaje que combina marcos y reglas y es una extensión de HTML, que permite la inserción de ontologías en documentos

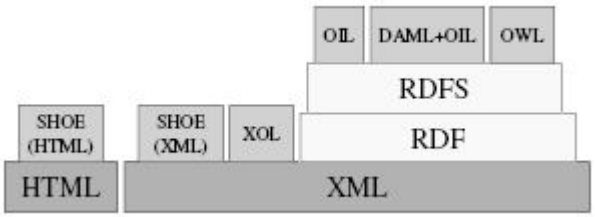


**Figura 3.7:** Lenguajes de marcado de ontologías

HTML. Posteriormente, su sintaxis se adaptó a XML. El resto de lenguajes de marcado de ontologías están basados en XML.

A continuación, se describen brevemente algunos de los lenguajes de la Web Semántica, así como su interrelación.

#### 3.3.2.3 RDF

RDF (*R*esource *D*escription *F*ramework) fue desarrollado por la W3C y define un lenguaje de especificación de ontologías ligero. Es el lenguaje de ontologías sobre el que se ha basado gran parte del cuerpo de conocimiento de la Web Semántica. Su uso se describe en Manola and Miller (2004), mientras que en Klyne and Carroll (2004) se detalla su sintaxis abstracta, en Hayes (2004) su semántica y en McBride (2004), su sintaxis de intercambio de información más usada, basada en XML. Es el primer lenguaje de la Web Semántica que incluye una semántica formal (los siguientes también la incorporan), es decir, su semántica puede ser comprendida por las máquinas. Esta aportación es fundamental porque permitirá razonar con los lenguajes para inferir información. Sin embargo, el razonamiento tiene poca aplicabilidad en RDF dado que su semántica es muy sencilla y se basa en la *tripleta*, que simplemente especifica una relación entre un sujeto y un objeto a través de una propiedad. La conjunción de *tripletas* referidas a un conjunto concreto de recursos forma un grafo RDF, que se considera una ontología RDF. Sujetos, objetos y propiedades se identifican mediante una URI (*U*niform *R*esource *I*dentifier). Los objetos, además de elementos identificados por URIs, pueden ser literales, es decir, valores con su tipo de dato asociado. Un mismo recurso puede usarse en unas *tripletas* como sujeto, en otras como objeto, y en otras incluso como propiedad. La especificación de literales puede hacer uso de los tipos definidos en XML Schema, y apenas se establecen restricciones adicionales.

El RDF Schema (RDFS) (Brickley and Guha (2002)), extiende la sintaxis y la semántica de RDF creando un nuevo lenguaje o metamodelo a partir de éste. RDFS puede considerarse como un modelo RDF más, al estar basado completamente en este modelo pero, dada la semántica definida para este modelo RDF, se convierte en un nuevo lenguaje para especificar otros dominios de conocimiento. Lo peculiar de este nuevo lenguaje es que los modelos de información basados en él también pueden hacer uso de RDF. Es decir, ambos lenguajes pueden convivir al mismo nivel puesto que la semántica de RDFS extiende la de RDF. RDFS permite definir restricciones sobre los recursos y propiedades de RDF. Por ejemplo, se definen el recurso *class* y la propiedad *type*, que permiten definir ejemplares y las clases a las que pertenece. También se crean las propiedades *subclassOf*, para especificar herencia de clases, *range* y *domain* para especificar clases como rango y dominio de una propiedad, *subpropertyOf* para definir especialización de propiedades, etc. Utilizando estos nuevos elementos del metamodelo RDFS, se construyen los vocabularios RDF que sí podemos considerar como ontologías ligeras.

#### 3.3.2.4 OWL

OWL (Smith et al.), ha sustituido como recomendación del W3C en materia de especificación de ontologías a su predecesor DAML+OIL. Este último surgió como resultado de los trabajos de OIL, desarrollado en el contexto de un proyecto europeo (*European IST project On-To-Knowledge*) y DAML (*D*ARPA *A*gent *M*arkup *L*anguage), fruto de un proyecto estadounidense. Su sintaxis y semántica son muy parecidos a las de OWL pero este último es más completo. OWL es un verdadero lenguaje de especificación de ontologías, entre ligeras y pesadas (permite definir ciertos axiomas pero la expresividad está muy limitada). Se ha concebido como un modelo independiente, que puede interpretarse por sí solo, pero también puede ser compatible con RDF y RDFS, tanto en sintaxis como en semántica. De hecho, su sintaxis para intercambio de información más usada se basa en la sintaxis XML de RDF y RDFS (definida como RDF/XML, aunque también existe una sintaxis basada directamente en XML). Además, OWL utiliza los tipos de datos definidos en XML Schema.

Se han desarrollado tres versiones de OWL con diferentes propósitos:

**OWL Lite** es la versión más reducida, no compatible con todos los documentos RDF/RDFS. Se define como un subconjunto de las construcciones totales existentes para OWL y establece restricciones en su uso. Está pensado para principiantes o aquellos que buscan sobre todo la sencillez. Formalmente, su semántica puede considerarse como una extensión de un subconjunto de RDFS.

**OWL DL** (*OWL D*escription *L*ogic). Está destinado a aquellos usuarios que quieren máxima expresividad pero garantizando completitud computacional (posibilidad de llegar a conclusiones basadas en la información existente) e inferencia en tiempo finito. Incluye todas las construcciones definidas para OWL, pero se deben utilizar con ciertas restricciones para alcanzar las propiedades mencionadas. Esto hace que no sea compatible con documentos que usan la máxima expresividad de RDF/RDFS, ya que su semántica es también una extensión de un subconjunto de la de éste.

**OWL Full** Es la versión más amplia. Está destinada a aquellos usuarios que quieren máxima expresividad y la libertad sintáctica de RDF. Es compatible con RDF/RDFS y su semántica puede concebirse como una extensión de la de éste, es decir, sigue el mismo procedimiento de construcción sobre RDF y RDFS que RDFS en relación a RDF. Por lo tanto, los modelos basados en OWL Full pueden utilizar libremente construcciones propias de RDF y RDFS. A cambio de ello, no existen garantías sobre la completitud y el tiempo finito en el razonamiento. Ello se debe a que las propias construcciones que definen el metamodelo pueden utilizarse en los modelos. Esto significa mucha flexibilidad pero también demasiada libertad. Por ejemplo, se puede extender el elemento *class* de RDFS para definir un subtipo de clase y, además, los distintos elementos de información pueden pertenecer a la vez a diferentes tipos (por ejemplo, clases y propiedades que son a la vez ejemplares).

En la figura 3.8[8], se han representado de forma simbólica los ámbitos de las semánticas de estos lenguajes. Cada elipse representa la semántica de cada lenguaje y sus solapamientos implican compatibilidad desde un punto de vista semántico. Por ejemplo, el solapamiento entre las elipses de OWL DL y RDFS quiere decir que OWL DL, desde el punto de vista semántico, es compatible con un subconjunto de construcciones RDFS y viceversa.

[8] Proyecto REWERSE: *Reasoning on The Web with Rules and Semantics*, 6th Framework Programme, Information Society Technologies. http://rewerse.net/

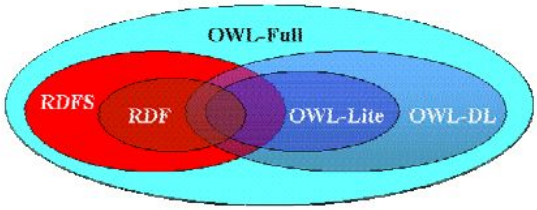


**Figura 3.8:** Ámbitos de las semánticas de los lenguajes de ontologías de la Web Semántica

#### 3.3.2.5 WSML: Lenguaje para el modelado de servicios Web

WSML (Web Service Modeling Language) (De Bruijn et al. (2005b)) proporciona sintaxis y semántica formal para la ontología de modelado de servicios web WSMO (Web Service Modelling Ontology) (De Bruijn et al. (2005a)). WSML se basa en diferentes formalismos lógicos, como la lógica descriptiva, la lógica de primer orden y la programación lógica. Estos formalismos son útiles para el modelado de los servicios web semánticos.

WSML está formado por variantes basadas en los diferentes formalismos lógicos, WSML-Core, WSML-DL, WSML-Flight, WSML-Rule y WSML-Full. La especificación de este lenguaje se encuentra publicada en (WSML, 2005).

En la Figura 3.9 se muestran las diferentes variantes de WSML y la relación entre ellas. Las flechas indican extensión. El lenguaje básico WSML-Core se extiende en las direcciones de la lógica descriptiva (WSML-DL) y de la programación lógica (WSML-Flight, WSML Rule). WSML-DL y WSML-Rule se extienden a una lógica de primer orden que unifica ambos paradigmas (WSML Full).

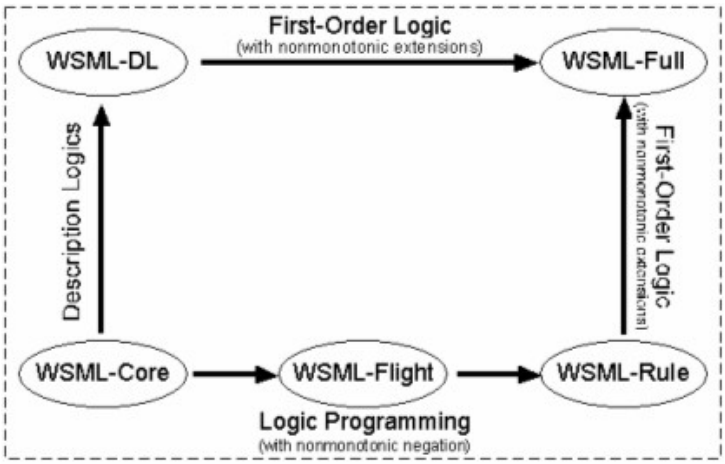


**Figura 3.9:** Espacio de WSML (Especificación WSML)

WSML cuenta con las siguientes variantes, según el formalismo lógico utilizado:

- WSML-CORE se corresponde con la intersección de lógica descriptiva y lógica de Horn, extendida con soporte a tipos de datos para que sea útil en las aplicaciones prácticas.
- WSML-DL extiende a WSML-CORE en la dirección de la lógica descriptiva, de tal forma que cubre la parte de OWL que puede ser implementada eficientemente.

- WSML-Flight extiende a WSML-CORE en la dirección de la programación lógica. Tiene un rico conjunto de primitivas de modelado para modelar diferentes aspectos de los atributos, tales como restricciones en los valores y restricciones de integridad.
- WSML-Rule extiende WSML-Flight a un lenguaje de programación lógica como tal, permitiendo el uso de funciones. La única diferencia con WSML-Flight se presenta en la sintaxis de las expresiones lógicas.
- WSML-Full unifica todas las variantes de WSML (lógica descriptiva y el paradigma de programación lógico).

Todas las variantes de WSML se describen en términos de una sintaxis normativa que puede ser leída y entendida por personas, además de esta sintaxis se provee una sintaxis XML y RDF para intercambio entre máquinas.

En la especificación de WSMO (De Bruijn et al. (2005a)) se definen también los componentes esenciales que definen una ontología y que son soportados por WSML. Las ontologías según esta especificación, definen una terminología común consensuada por medio de conceptos y relaciones entre los conceptos. Además, para capturar las propiedades semánticas de las relaciones y conceptos, una ontología generalmente también contiene un conjunto de axiomas expresados en algún lenguaje lógico.

#### 3.3.2.6 Otros lenguajes

Lenguajes de ontologías como RDF Schema y OWL ofrecen unas capacidades limitadas para razonamiento lógico. Por este motivo, se han desarrollado varios trabajos en los que se proponen lenguajes de reglas, que permiten definir axiomas como parte de una base de conocimiento, para la Web Semántica. Uno de los primeros trabajos fue TRIPLE (Sintek and Decker (2002)), lenguaje de reglas basado en cláusulas de Horn que se define sobre RDF junto con un motor de inferencias capaz de razonar sobre modelos definidos en TRIPLE. Otra iniciativa que merece ser destacada es RuleML, que es un lenguaje altamente expresivo, de etiquetas, para publicar y compartir descripciones de reglas en la Web. El entorno de RuleML incluye una jerarquía de sublenguajes de reglas sobre XML, RDF, XSLT y OWL. Este lenguaje contiene aspectos de programación lógica, programación funcional y orientación a objetos. Los impulsores de RuleML han enviado una propuesta de lenguaje llamado SWRL (*S*emantic *W*eb *R*ule *L*anguage[9]) para su consideración como estándar del W3C, combinando OWL-DL y un subconjunto de RuleML.

### 3.3.3 Herramientas de ontologías

Para facilitar la tarea de creación de ontologías, a mitad de los años 90's se crearon los primeros entornos de construcción de ontologías. Estos entornos, proporcionaban interfaces que ayudaban a realizar alguna de las principales tareas en el desarrollo de ontologías como conceptualización, implementación, documentación, chequeo de la consistencia, etc. En los últimos años, el número y tipo de herramientas de ontología ha crecido enormemente. Gomez-Perez et al. (2002) distingue los siguientes grupos:

- ***Herramientas de desarrollo de ontologías***. Son herramientas y paquetes integrados que pueden usarse para construir una nueva ontología desde cero. Además de funciones de

[9] http://www.w3.org/Submission/2004/SUBM-SWRL-20040521/

edición y búsqueda, son herramientas que proporcionan normalmente soporte para documentación de ontologías, para exportar e importar ontología a/desde diferentes formatos y lenguajes de ontologías, edición gráfica de ontologías, gestión de bibliotecas de ontologías, etc.

- ***Herramientas de evaluación de ontologías***. Se usan para evaluar el contenido de las ontologías y sus tecnologías relacionadas. Estas herramientas intentan prevenir problemas cuando se necesita integrar y usar ontologías y tecnologías basadas en ontologías en otros sistemas de información.

- ***Herramientas de fusión e integración de ontologías***. Se usan para resolver el problema de la combinación y la integración de diversas ontologías del mismo dominio. Esto ocurre, por ejemplo, cuando se unen dos organizaciones diferenciadas, o cuando se desea obtener una ontología de calidad a partir de las ya existentes.

- ***Herramientas de anotación basadas en ontologías***. Con ellas se pueden insertar instancias de conceptos y relaciones en ontologías así como mantener (semi)automáticamente, en páginas Web, marcados basados en ontologías. Muchas de estas herramientas han sido creadas recientemente, en el contexto de la Web Semántica.

- ***Herramientas de consulta de ontologías y motores de inferencia***. Permiten realizar fácilmente consultas a las ontologías e inferencias con ellas. En general, son herramientas que están estrechamente relacionadas con el lenguaje usado para implementar ontologías. En la siguiente sección se hace hincapié en este tipo de herramientas ya que se ha utilizado una de ellas en el desarrollo del método de diagnóstico propuesto.

- ***Herramientas de aprendizaje de ontologías***. Pueden derivar ontologías (semi)automáticamente a partir de textos en lenguaje natural, fuentes semiestructurados y bases de datos usando técnicas de análisis de lenguaje natural y aprendizaje automático.

Existen herramientas que están constituidas por paquetes integrados de herramientas de algunos de los tipos anteriores. Sin embargo, por su aplicación en la tesis, sólo nos vamos a centrar en describir con más detalle dos de los tipos previos de herramientas de ontologías: herramientas de desarrollo de ontologías y motores de inferencia.

### 3.3.4 Herramientas de desarrollo de ontologías

De acuerdo a su evolución, desde su aparición a mitad de los años 90's, se pueden distinguir dos tipos de herramientas de desarrollo de ontologías:

- Herramientas cuyo modelo de conocimiento mapea directamente un lenguaje de ontología. Son herramientas que se desarrollaron como editores de ontologías para un lenguaje específico. Entre estas se encuentran:

  **Servidor de Ontolingua** (Farquhar et al. (1997)), que soporta construcción de ontologías con Ontolingua y KIF. Fue la primera herramienta de ontologías, creada a mitad de los años 90's. Se construyó para facilitar el desarrollo de ontologías Ontolingua con una interfaz Web basadas en formularios. Inicialmente, la principal aplicación dentro de este Servidor fue un editor de ontologías. Más tarde, se añadieron otros sistemas en el entorno, tales

como un Webster, un servidor OKBC, la herramienta Chimaera para fusión de ontologías, etc.

**Ontosaurus** (Swartout et al. (1997)). En paralelo al desarrollo del Servidor Ontolingua, esta herramienta fue implementada como un editor Web y buscador de ontologías LOOM. Ontosaurus consiste en dos módulos principales: un servidor de ontologías, que usa el sistema de representación del conocimiento asociado al lenguaje LOOM, y un buscador Web que permite editar y buscar ontologías LOOM con formularios HTML.

**WebOnto** (Domingue (1998)). En 1997 fue hecha pública. WebOnto es un editor de ontologías para OCML. El editor de ontologías no estaba basado en formularios HTML, sino en *applets* de Java, Su gran ventaja sobre el resto de herramientas de desarrollo de ontologías es su fuerte soporte para edición de ontologías colaborativas, que permitía entre grupos de usuarios discusiones asíncronas y síncronas sobre las ontologías a construir.

**OilEd** (Bechhofer et al. (2001)) fue desarrollado en el 2001 como un editor para ontologías OIL inicialmente. Más tarde, con la creación de DAML+OIL, esta herramienta fue adaptada para gestionar ontologías DAML+OIL, y después, ha sido adaptada para gestionar OWL. OilEd proporciona funciones de chequeo de consistencia y clasificación automática de conceptos, por medio del motor de inferencia FaCT [Horrocks et al., 1999] aunque otros motores de inferencia DL tales como RACER [Haarslev and Möller, 2001] pueden también usarse.

- En los últimos años, han surgido una nueva generación de entornos en el ámbito de la ingeniería ontológica, que se construyen como paquetes de herramientas integradas. Estos entornos proporcionan soporte tecnológico para una amplia variedad de arquitecturas basadas en componentes, donde nuevos módulos pueden añadirse para proporcionar más funciones para el paquete de herramientas. Por ello, los modelos de conocimiento subyacentes a estos entornos son normalmente lenguajes independientes y proporcionan traducciones desde y a diversos lenguajes y formatos. En su evolución, cabe destacar:

  **Protégé-2000** (Noy et al. (2001)) es *open source* y una aplicación *standalone* con una arquitectura extensible. El núcleo de Protégé-2000 es su editor de ontologías, que puede ser extendido con *plugins* y así, añadir más funciones al entorno, tales como importar y exportar lenguajes de ontologías (FLogic, Jess, OIL, XML, Prolog), acceso OKBC, creación y ejecución de restricciones (PAL), fusión de ontologías (PROMPT), etc. También se ha creado un *plug-in* para edición gráfica de ontologías, para desarrollar ontologías OWL, etc.

  **WebODE** (Corcho et al. (2002)) es también un paquete de ingeniería de ontologías extensible, basado en una aplicación servidora. Su desarrollo comenzó en 1999. El núcleo de WebODE es su servicio de acceso a ontologías que se usa por todos los servicios y aplicaciones añadidas en el servidor. El Editor de Ontologías de WebODE permite editar y buscar ontologías WebODE, está basado en formularios HTML y *applets* Java. El *workbench* integra servicios para importar y exportar lenguajes de ontologías (XML, RDF(S), OIL, DAML+OIL, OWL, CARIN [Levy and Rousset, 1998], FLogic, Jess, Prolog), para edición de axiomas con WAB (*W*ebODE *A*xiom *B*uilder), documentación, evaluación, fusión y un motor de inferencia. WebODE puede también interoperar con Protégé-2000.

  **OntoEdit** (S. et al. (2002)) es un entorno extensible y flexible basado en una arquitectura *plug-in*. Su editor de ontologías es una aplicación *standalone*, que permite editar y buscar ontologías, incluye funciones para construcción colaborativa de ontologías, para inferir, etc. El editor exporta e importa ontologías en diferentes formatos (XML, FLogic, RDF(S) y

DAML+OIL). Hay dos versiones de OntoEdit: OntoEdit Free y OntoEdit Professional, cada una con un conjunto diferentes de funciones.

**KAON** (Maedche and Staab (2003)) es un entorno de ontologías *open source* extensible . El núcleo de este paquete de herramientas es la API de ontología, que define su modelo de conocimiento subyacente basado en una extensión de RDF(S). OI-Modeller es el editor de ontologías del paquete de herramientas y proporciona capacidades para evolución, mapeo y generación de ontologías a partir de base de datos, etc.

**OntoStudio** (Weiten (2009)) es el front-end homólogo a OntoBroker, una rápida máquina de inferencia F-Logic basada en datalog. En consecuencia, un foco del desarrollo de OntoStudio ha sido el soporte de diversas tareas relacionadas con la aplicación de reglas como la creación de reglas mediante un editor de reglas textual y gráfico, la aplicación de reglas para la integración dinámica de fuentes de datos (usando una importación de un esquema de base de datos y una herramienta de mapeo). OntoStudio es escalable y, por lo tanto, apropiado incluso para grandes ontologías. Al estar basado en la plataforma Eclipse, OntoStudio proporciona un marco abierto para los desarrolladores de plug-in. Proporciona ya plug-ins de consultas, visualizador y de informes (soporta la herramientas de informes BIRT). OntoStudio también incluye un editor para la creación y gestión de mapeo de ontologías. OntoStudio está fuertemente unido a F-Logic, la importación/exportación desde/a OWL/RDF está restringida principalmente a conceptos que pueden expresarse en F-Logic.

**NeOn Toolkit** (Hasse et al. (2008)) es parte de la implementación de la arquitectura NeOn. Se trata de un entorno de ingeniería de ontologías multi-plataforma, que ayuda a proporcionar un soporte completo para todas las actividades en el ciclo de vida de la ingeniería de ontologías. NeOn toolkit está basado en la plataforma Eclipse y proporciona un conjunto extenso de plug-ins que cubren todos los aspectos de la ingeniería de ontologías. Este conjunto incluye: integración de bases de datos relacionales, modularización, visualización, alineación y gestión de proyectos.

**SWOOP** (Kalyanpur et al. (2005)) es un editor de ontologías OWL *open source* basado en hypermedia. Los usuarios pueden visualizar y editar múltiples ontologías OWL de una forma "similar a la Web", que incluye la navegación vía hipervínculos y características de anotaciones. SWOOP proporciona así una alternativa a herramientas de ontologías basadas en Web pero ofrece características adicionales tales como un mecanismo plugin. Esta herramienta también soporta chequeo de la consistencia de las ontologías basándose en las capacidades de razonadores adjuntos. Otras características principales de Swoop son la depuración para ontologías OWL, explotando las características de los razonadores OWL (como el razonador DL y *open source* Pellet). Esto último incluye, por ejemplo, la generación automática de explicaciones para un conjunto insatisfacible de axiomas. Sin embargo, Swoop no incluye funcionalidades típicas adicionales como la integración o importación de fuentes externas (no OWL ni RDF).

**TopBraid Composer**[10] es una herramienta de modelado para la creación y mantenimiento de ontologías. Es un editor completo para modelos RDF y OWL construido sobre la plataforma Eclipse. TopBraid Composer soporta chequeo de la consistencia y otras tareas de razonamiento. El sistema tiene el razonador Pellet como motor de inferencia por defecto, aunque pueden asociarse otros clasificadores. Históricamente el desarrollo de TopBraid

[10] http://www.topbraidcomposer.com/

Composer tiene sus raíces en Protégé de tal modo que, algunos de sus conceptos son similares. TopBraid Composer ofrece funcionalidades más allá de la creación y gestión de archivos OWL/RDF(s), como importación de bases de datos, XML-Schemas, UML y hojas de cálculo, así como un soporte básico para reglas, en formato Jena o SWRL. Ambos tipos de reglas se ejecutan con el motor de inferencia interno de Jena. Otras características de TopBraid Composer son la visualización de relaciones entre recursos RDF/OWL(s) de una forma gráfica y el soporte para la edición concurrente de varias ontologías. Además, proporciona una característica de explicación para OWL DL basada en Pellet similar a SWOOP. A diferencia de Protégé, está orientado esencialmente a profesionales, no a una gran comunidad.

### 3.3.5 Motores de inferencia

En el campo de la Web Semática otra de las actividades importantes que se están realizando por diversos grupos de investigación, es el desarrollo de motores de inferencia o herramientas que den soporten a motores de inferencia para algunos de los lenguajes de la Web Semática. Se pueden clasificar en dos grupos, razonadores basados en reglas y razonadores basados en lógica descriptiva. Entre las herramientas más destacadas en ambos grupos, se pueden citar las siguientes:

**Razonadores basados en reglas**:

- **Bossam**[11] Es un motor de reglas basado en RETE con soporte nativo para razonamiento sobre ontologías OWL, SWRL y reglas RuleML. Bossam posee diversas características de expresividad que incluyen: a) referencias URI como símbolos, b) sintaxis en lógica de segundo orden, c) disyunciones en el antecedente y conjunciones en el consecuente y d) soporte para la negación clásica y la negación por ausencia (*negation-as-failure*). Además, Bossam carga, realiza razonamiento y responde a las consultas sobre un conjunto de conocimiento, que puede incluir cualquier combinación de documentos de los tipos siguientes: RDF(s), OWL, Bossam rules, SWRL (+OWL) así como llamar a objetos java desde el antecedente y consecuente de las reglas mediante el método *attachment* java basado en URIs.

- **Jena**[12] es un entorno Java para el desarrollo de aplicaciones de la Web Semática desarrollado en los Laboratorios HP en Bristol [CARROLL 04] en el marco de un proyecto *open source*. Proporciona un entorno de programación para RDF, RDFS, OWL y SPARQL e incluye un motor de inferencia basado en reglas. El entorno Jena incluye una API RDF, lectura y escritura RDF en RDF/XML, N3 y N-triples, una API OWL y un motor de consultas SPARQL.

- **Sweet-Rules**[13] es una toolkit de reglas para el lenguaje RuleML, mencionado en el apartado 3.3.2.6. El motor de SweetRules tambien proporciona traducción entre otros lenguajes de reglas y ontologías (axiomas implícitos), conservando las semánticas correspondientes.

**Razonadores basados en lógica descriptiva**:

[11] http://bossam.wordpress.com
[12] http://jena.sourgeforge.net/
[13] http://sweetrules.projects.semwebcentral.org/

- **Pellet** Sirin et al. (2007) es un razonador OWL DL *open source*, basado en Java. Fue desarrollado en la Universidad de Maryland. Pellet puede usarse en conjunción con Jena y bibliotecas API de OWL. Asimismo, proporciona una interfaz DIG y funcionalidades para ver la validación de (species?), chequeo de la consistencia de ontologías, clasificar taxonomías, chequeo (entailments) y respuesta a un subconjunto de consultas RDQL. Soporta, además, la completa expresividad de OWL DL, incluyendo razonamiento acerca de *nominals* (clases enumeradas).

- **FaCT++** (Horrocks (2003)) desarrollado en la Universidad de Manchester, es el descendiente del sistema de razonamiento FaCT. Proporciona soporte completo para OWL-Lite. Sus futuras versiones ayudarán a proporcionar completo soporte para razonamiento OWL-DL.

- **KAON2** (Motik and Studer (2004)) Es una infraestructura para la gestión de ontologías OWL-DL, SWRL, y F-Logic. KAON2 es el sucesor del proyecto KAON (denominado KAON1) cuya principal diferencia es el lenguaje de ontología soportado. Sin embargo, KAON2 es un sistema completamente nuevo, no compatible con KAON1. Otras características importantes que proporciona este razonador son las siguientes: a) un servidor stand-alone que proporciona acceso a ontologías de una forma distribuida, b) un motor de inferencia para responder a consultas conjuntivas usando sintaxis SPARQL, una interfaz DIG (interfaz estandarizada XML desarrollada por el grupo de implementación DL) para acceder a herramientas como Protégé y c) un módulo para extraer instancias de ontología a partir de bases de datos relacionales. A diferencia de otros razonadores DL (Fact++, RACER, Pellet, etc.), KAON2 implementa el razonamiento mediante nuevos algoritmos (Hustadt et al. (2004)) que permiten aplicar técnicas de bases de datos deductivas que aumenta considerablemente el rendimiento del razonador en la ejecución de los procesos de inferencia.

- **RacerPro**[14] es una razonador OWL basado en lógica descriptiva y un servidor de inferencia para la web semántica [Haar 2001]. Racer soporta inferencia sobre ontologías RDFS/DAML-/OWL a través de reglas especificadas explícitamente por el usuario.

- **TRIPLE**[15] es un motor de inferencia (y un lenguaje) basado en Lógica de Horn y usa muchas de las características de F-Logic. A diferencia de F-Logic, no tiene semánticas establecidas para clases y objetos. Puede ser usado traduciendo Lógicas Descriptivas basadas en OWL en un lenguaje, denominado TRIPLE, manejado por el razonador. Extensiones de Lógicas Descriptivas que no pueden ser manejadas mediante lógica Horn pueden ser soportadas incorporando otros razonadores, tales como FaCT, para crear un sistema de razonamiento híbrido.

En la tabla 3.1 se realiza una comparativa de los razonadores semánticos anteriores. En cada celda, un símbolo '+' en una celda $c_{ij}$ representa que el razonador *j* tiene la característica *i*. Un símbolo '-' representa que el razonador *j* no tiene la característica *i* y, en caso contrario, se detalla en qué medida el razonador *j* cumple la característica *i*.

[14] http://www.sts.tu-harburg.de/r.f.moeller/racer/
[15] http://triple.semanticweb.org/

Tabla 3.1: Comparativa de razonadores semánticos

| CARACTERÍSTICA | BOSSAM | PELLET | KAON2 | RACERPRO | JENA | FACT++ | SWEET-RULES |
|---|---|---|---|---|---|---|---|
| **Vinculación OWL-DL** | *?* | + | + | + | *No completo. Razonador incluido con la distribución estándar* | + | - |
| **Expresividad soportada para razonamiento** | *?* | *SROIQ(D)* | *SHIQ(D)* | *SHIQ(D-)* | *Varía según el razonador (incompleta para lógicas descriptivas no triviales* | *SROIQ(D)* | *?* |
| **Algoritmo de razonamiento** | *Basado en reglas* | *Tableau* | *Resolución + Datalog* | *Tableau* | *Basado en reglas* | *Tableau* | *Basado en reglas* |
| **Chequeo de la consistencia** | *?* | + | *?* | + | *Incompleta para OWL DL* | + | - |
| **Soporte DIG** | - | + | + | + | + | + | - |
| **Soporte de reglas** | + *(SWRL y formato propio)* | + *(SWRL->DL Safe Rules)* | + *(SWRL->DL Safe Rules)* | + *(SWRL -no soportado completamente y formato propio)* | + *(formato propio)* | - | + *(SWRL, RuleML, y Jess)* |
| **Licencia** | *Free/closed-source* | *Free/open-source Non-Free/closed-source* | *Free/closed-source* | *Non-Free/closed-source* | *Free/open-source* | *Free/open-source* | *Free/open-source Non-Free/closed-source* |

#### 3.3.5.1 Almacenamiento de instancias de la ontología

En muchos escenarios prácticos, la mayoría de las herramientas citadas anteriormente no son suficientes para el desarrollo de ciertas aplicaciones en la Web Semántica. En estos escenarios, los usuarios requieren acceder al conocimiento no sólo declarado sino también al conocimiento inferido que puede derivarse mediante la inferencia basada en la ontología. En la Web Semántica, las instancias de las ontologías (por ejemplo, el conocimiento representado mediante tripletas RDF) se almacenan en los llamados *triple stores* o *RDF databases*.

Hay fundamentalmente tres estrategias alternativas para gestionar el conocimiento inferido en el *triple store* influenciadas por la inferencia lógica:

- *Encadenamiento hacia adelante*. Aplica la consecución de las reglas tan pronto como las tripletas RDF se han añadido en el almacén de tripletas. Esta aproximación elimina la inferencia en tiempo de ejecución, enumerando y almacenando todo el conocimiento posible. Por ello, da lugar a respuestas rápidas a consultas, con el coste de carga e incremento del espacio de almacenamiento. Esta aproximación no es tan prometedora ya que: a) no hay garantía de que las tripletas inferidas sean consultadas en el futuro; y b) el espacio de almacenamiento para las tripletas inferidas puede ser prohibitivamente grande e imponer una sobrecarga en el acceso.

- *Encadenamiento hacia atrás*. Aplica la consecución de las reglas cuando el almacén de tripletas sea consultado. Esta aproximación realiza la inferencia en tiempo de ejecución, sin necesidad de almacenar el conocimiento inferido. Por ello, da lugar a un tiempo de carga bajo, con el consiguiente incremento del tiempo de respuesta a una consulta.

- *Inferencia híbrida*. Combina encadenamiento hacia adelante y hacia atrás para evitar las desventajas de ambos ya citadas previamente.

En cuanto a opciones de almacenaje hay dos muy conocidas, denominadas almacenaje en memoria (*in-memory storage*) y almacenamiento persistente (*persistent storage*). Su rendimiento ha sido evaluado utilizando varios criterios. Entre los más populares almacenes de tripletas está Jena, ya mencionada previamente, con ambas opciones de almacenaje, persistente y en memoria (la biblioteca de acceso a datos basada en Java oculta los detalles de almacenaje y, por esta razón, no hay mucha diferencia entre los accesos a los dos tipos de almacenes, persistente o en memoria). Además, Jena permite encadenamiento de los tres tipos mencionados previamente. Otros almacenes de tripletas muy conocidos y con almacenamiento persistente son RSSDB, Kowari, Sesame, 3Store, Instance Store y DLDB (Ding et al. (2005)). Se presta atención a almacenes con esta opción de almacenaje puesto que el conocimiento en la Web Semántica se supone que será de grandes proporciones.

En la tabla 3.2 se muestran las capacidades de los Almacenes RDF persistentes. Las tres clases de estrategias de inferencia se representan con la siguiente notación: 'F', para encadenamiento hacia adelante, 'B', para encadenamiento hacia atrás y 'H', para encadenamiento híbrido. '-' se usa cuando la información correspondiente no está disponible.

### 3.3.6 Metodologías de desarrollo de ontologías

Los términos **metodología**, **método**, **técnica**, **proceso** y **actividad** son usados indiscriminadamente en la literatura. IEEE define metodología, método y técnica de la forma siguiente:

**Tabla 3.2:** Comparativa de las capacidades de Almacenes Persistentes RDF. Extraído de (Ding et al. (2005))

| | Jena | RSSDB | Kowari | Sesame | 3Store | Instance Store | DLDB |
|---|---|---|---|---|---|---|---|
| Query Language | RDQL | RQL | iTQL, RDQL | SeRQL | OKBC, RDQL | Racer, FaCT++ based | conjunctive KIF |
| Level of inference | RDFS, OWL(partial) | RDFS | Explicit Rules | RDFS | RDFS | OWL-Lite, OWL-DL | OWL-DL |
| Strategy | F, B, H | - | F | - | H | - | F |

*Una metodología es una serie completa e integrada de técnicas o métodos para crear una teoría de sistemas general de cómo una clase de trabajo intensivo en ideas debería ser realizado.*

*Un método es un conjunto de procesos o procedimientos ordenados usado en la ingeniería de un producto o en la realización de un servicio.*

*Una técnica es un procedimiento técnico y directivo usado para alcanzar un objetivo dado.*

El término *metodología* no significa lo mismo que *método*. La metodología se refiere al conocimiento sobre los métodos (Hodge (2000)). Establece "qué", "quiénesz çuándoüna actividad debería realizarse. Asimismo, un método no es lo mismo que una técnica. Un método es un procedimiento general, mientras que una técnica es una aplicación específica de un método y la forma en que el método se ejecuta (Greenwood (1973)). Normalmente, hay varias técnicas para aplicar un método dado. Ambos, método y técnica, están estrechamente relacionados ya que se usan para llevar a cabo tareas en los diferentes procesos de que consta una metodología. En este sentido, IEEE define también proceso, actividad y tarea de la siguiente forma:

*Un proceso es una función que debe ser realizada en el ciclo de vida software.*

*Una actividad es un conjunto de trabajo que es realizado, incluyendo la información de entrada y salida que requiere.*

*Una tarea es la asignación de un trabajo bien definido a uno o más miembros de proyecto. Las tareas relacionadas son normalmente agrupadas para formar actividades.*

Las relaciones entre las definiciones previas se resume en la Figura 3.10.

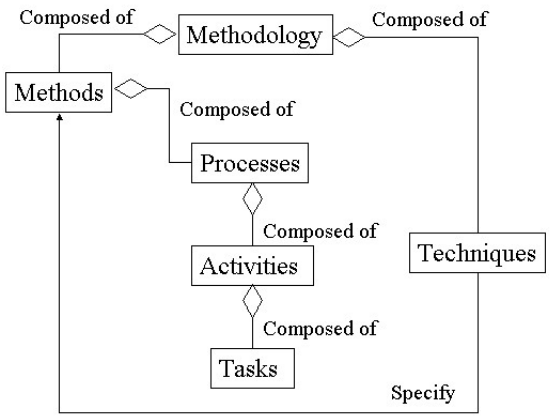


**Figura 3.10:** Representación gráfica de las relaciones terminológicas en metodologías (Gomez-Perez et al. (2003))

Una vez presentadas las definiciones proporcionadas por IEEE para los términos que se van a manejar en esta sección, se presenta a continuación, brevemente y de forma cronológica, las metodologías más importantes que se han propuesto para el desarrollo de las ontologías para, posteriormente, describir algunas de las más destacadas.

#### 3.3.6.1 Estado del arte de metodologías de desarrollo de ontologías

En 1990, se desarrollaron algunos pasos generales sobre el desarrollo de la ontología Cyc. Más tarde, en 1995, a partir de la experiencia en el desarrollo de dos ontologías en el dominio del modelado de la empresa (Enterprise y TOVE -TOronto Virtual Enterprise, respectivamente), se propusieron las primeras guías metodológicas. En 1996, se presentó como parte del proyecto Esprit KACTUS, un método para construir una ontología en el dominio de las redes eléctricas. Simultáneamente, apareció la metodología METHONTOLOGY (Gomez-Perez et al. (2003)). Posteriormente, en el 2001, apareció la metodología On-To-Knowledge (Staab et al. (2001)), dentro del proyecto del mismo nombre. Tras ésta, en 2004, apareció la metodología DILIGENT (Pinto et al. (2004)) como soporte de los expertos de dominio en un entorno distribuido para ingenieros y desarrollo de ontologías.

En el desarrollo de la ontología de este trabajo se busca una guía metodológica adecuada a sus características. Por esta razón, se incluye a continuación una breve descripción de tres de las metodologías consideradas más representativas; METHONTOLOGY, On-To-Knowledge y DILIGENT. Una explicación más detallada de ellas puede encontrarse en Gomez-Perez et al. (2003).

**METHONTOLOGY**

La metodología de construcción de ontologías METHONTOLOGY (Fernández-López et al. (1997), Blázquez et al. (1997)) fue desarrollada por el grupo de Ingeniería Ontológica en la Universidad Politécnica de Madrid.

METHONTOLOGY incluye la identificación del proceso de desarrollo de la ontología (que es el conjunto de actividades que se deben llevar a cabo para construir ontologías), un ciclo de vida basado en el desarrollo de prototipos, y técnicas para llevar a cabo cada actividad en la gestión, cada actividad de desarrollo y cada actividad de soporte.

METHONTOLOGY propone la construcción de un ciclo de vida en el desarrollo de prototipos, permitiendo añadir, cambiar y eliminar términos en cada nuevo prototipo. Para cada prototipo, esta metodología comienza con la actividad de planificación que identifica las tareas a realizar, su disposición y el tiempo y recursos necesarios para su finalización. A continuación, la actividad de especificación de la ontología comienza y, a la vez, varias actividades de gestión (control y garantía de calidad) y de soporte (adquisición de conocimiento, integración, evaluación, documentación y gestión de la configuración) empiezan también. Por lo tanto, todas las actividades previas de gestión y soporte se realizan en paralelo con las actividades de desarrollo (especificación, conceptualización, formalización, implementación y mantenimiento) durante el ciclo de vida completo de la ontología (ver Figura 3.11).

Para la actividad de especificación de la ontología, METHONTOLOGY propone el uso de cuestiones de competencia o representaciones intermedias para describir los requerimientos que la ontología debería cumplir sin especificar una guía detallada de cómo llevar a cabo esta actividad. Para la actividad de adquisición de conocimiento, la metodología propone el uso de técnicas tomadas del campo de la ingeniería de conocimiento (técnicas como diversos tipos de

entrevistas -estructurada, semi-estructurada, etc., cuestionarios, técnicas de modelado como mapas conceptuales, etc.).

Una vez que el primer prototipo ha sido especificado, el modelo conceptual se construye en la actividad de conceptualización para la que METHONTOLOGY proporciona, en este caso, guías detalladas. A continuación, las actividades de formalización e implementación son realizadas. Si se detecta alguna carencia después de estas actividades, se puede regresar a cualquiera de las anteriores para hacer modificaciones o refinamientos. Herramientas como el editor de ontologías WebODE, Protégé, etc. permiten implementar automáticamente el modelo de conceptualización en varios lenguajes de ontologías. Por lo tanto, no es obligatoria en esta metodología la fase de formalización. La Figura 3.11 muestra el ciclo de vida propuesto en METHONTOLOGY descrito anteriormente.

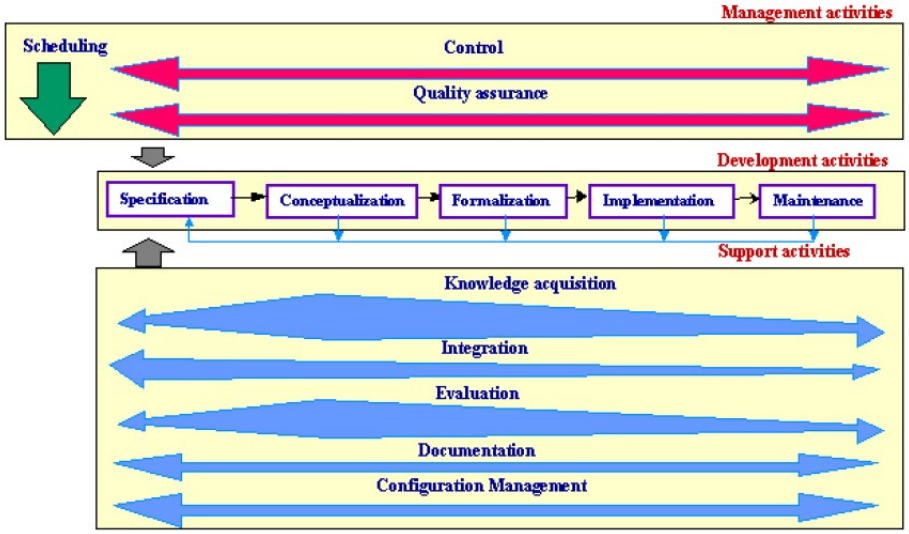


**Figura 3.11:** Ciclo de Vida de Ontologías en METHONTOLOGY (Suárez-Figueroa et al. (2008a))

Como puede observarse en la figura, la adquisición de conocimiento, integración y evaluación son mayores durante la conceptualización de la ontología y decrecen durante la formalización e implementación. Las razones de esto son fundamentalmente:

- La mayor parte del conocimiento es adquirido en el comienzo de la construcción de la ontología.
- La integración de otras ontologías en la que se está construyendo debería realizarse en el nivel de conocimiento y no pos-ponerse hasta la actividad de implementación.
- La conceptualización de la ontología debería evaluarse con precisión para evitar la propagación de errores en pasos futuros del ciclo de vida de la ontología.

METHONTOLOGY considera no sólo intra-dependencias, es decir, relaciones entre las actividades realizadas durante el desarrollo de la ontología sino también inter-dependencias o relaciones entre las actividades realizadas cuando se construyen ontologías diferentes ya que, frecuentemente, antes de integrar una ontología en una nueva, la ontología a ser reutilizada es modificada o fusionada con otras ontologías del mismo dominio. Relacionado con ello, METHONTOLOGY incluye una lista de actividades a realizar durante la reutilización de la ontología

pero sin proporcionar guías detalladas para tales actividades. METHONTOLOGY tampoco considera diferentes niveles de granularidad al reutilizar las ontologías.

### On-To-Knowledge

El objetivo del proyecto On-To-Knowledge (Staab et al. (2001)) fue aplicar ontologías a la información disponible electrónicamente para mejor la calidad de la gestión de conocimiento en organizaciones grandes y distribuidas. Algunos de los participantes en este proyecto fueron: el instituto AIFB de la Universidad de Karlsruhe, la Universidad Vrije de Amsterdam y British Telecom. En este proyecto se desarrolló una metodología y herramientas para acceso inteligente a grandes volúmenes de información semi-estructurada y textual en entornos intranet y extranet.

La metodología describe cómo construir ontologías utilizadas en la gestión de conocimientos. Esta metodología propone construir ontologías teniendo en cuenta cómo van a ser éstas utilizadas en nuevas aplicaciones. Además, On-To-Knowledge propone el aprendizaje de ontologías para reducir los esfuerzos en el desarrollo de la ontología, que incluye la identificación de metas a ser alcanzadas por las herramientas de gestión de conocimiento basándose en un análisis de escenarios de uso. Los procesos propuestos por la metodología se muestran en la Figura 3.12 y se describen a continuación brevemente.

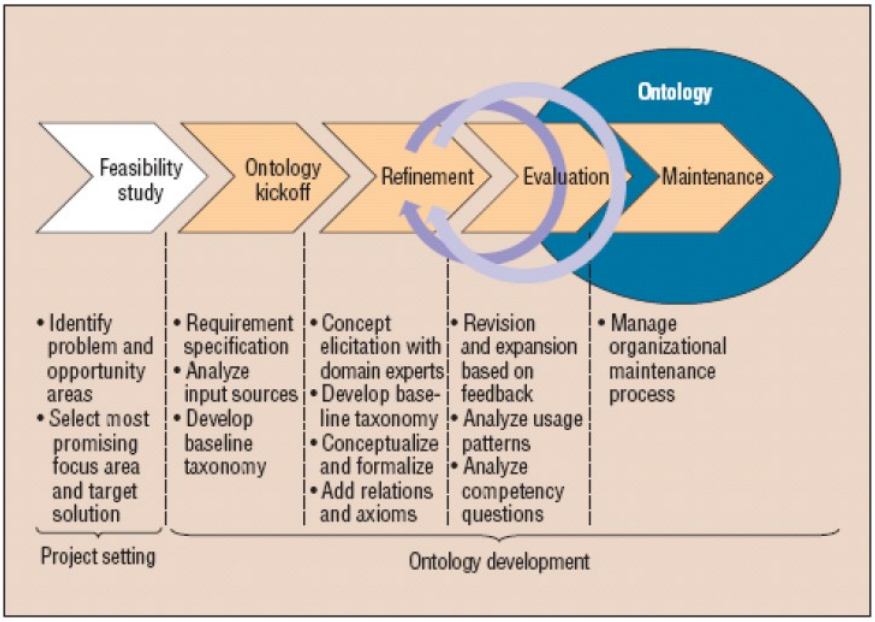


**Figura 3.12:** Ciclo de Vida de las ontologías en On-To-Knowledge (Staab et al. (2001))

- Proceso 1: *Feability study*. On-To-Knowledge adopta la clase de estudio de viabilidad descrita en la metodología CommonKads (Schreiber et al. (1999)). El estudio de viabilidad se aplica en esta metodología a la aplicación completa y, por lo tanto, debería realizarse antes de desarrollar las ontologías sirviendo como base del proceso de inicio siguiente.

- Proceso 2: *Kickoff*. El resultado de este proceso es el documento de especificación de requerimientos de la ontología que describe lo siguiente: el dominio y meta de la ontología; las guías de diseño (por ejemplo, convenciones de nombrado); fuentes de conocimiento

disponibles (libros, revistas, entrevistas, etc.); usuarios potenciales y casos de uso así como aplicaciones soportadas por la ontología. Adicionalmente, otro resultado de este proceso es un borrador semi-formal con la descripción de la ontología.

On-To-Knowledge propone cuestiones de competencias (Grüninger and Fox (1994)) para realizar la actividad de especificación de la ontología sin dar líneas detalladas para esta actividad. Estas preguntas pueden ser útiles para elaborar el documento de especificación de requerimientos. La especificación de requerimientos debe permitir al ingeniero de ontologías decidir sobre la inclusión o exclusión de conceptos en la ontología o sobre su estructura jerárquica.

Aunque la metodología menciona que en este proceso los desarrolladores deberían buscar ontologías potencialmente reutilizables ya desarrolladas, no proporciona guías detalladas para identificar tales ontologías así como tampoco para reutilizarlas. Además, la metodología no aporta explícitamente guías para la reutilización y reingeniería de recursos no ontológicos, ni para reutilizar patrones de diseño ontológicos.

- Proceso 3: *Refinement*. La meta de este proceso es producir una *target ontology* madura y orientada a la aplicación de acuerdo a la especificación dada en el proceso 2. El proceso de refinamiento se divide en dos actividades:

  - Actividad 1: *Knowledge elicitation process with domain experts*. En esta actividad, la ontología *baseline*, obtenida en el proceso 2, se refina mediante la interacción con los expertos del dominio. On-To-Knowledge propone el uso de representaciones intermedias para modelar el conocimiento. Si varios expertos participan en la construcción de la ontología, es necesario alcanzar un acuerdo. Una forma complementaria para enriquecer la ontología es usarla como semilla en un proceso de aprendizaje de ontologías.
  - Actividad 2: *Formalization*. La ontología es implementada usando un lenguaje de ontologías que es seleccionado según los requerimientos específicos de la aplicación prevista.

- Proceso 4. *Evaluation*. Este proceso sirve como prueba de la utilidad de las ontologías desarrolladas y su entorno software asociado. El producto obtenido se llama ontología basada en la aplicación. Durante este proceso se realizan dos actividades:

  - Actividad 1: *Checking the requirements and competency questions*. Los desarrolladores revisan si la ontología satisface los requerimientos y "puede responder.ª las preguntas de competencias.
  - Actividad 2: *Testing the ontology in the target application environment*. El refinamiento adicional de la ontología puede lograrse en esta actividad.

  Se necesitan varios ciclos hasta que la ontología objetivo alcance el nivel previsto.

- Proceso 5: *Maintenance*. La metodología On-To-Knowledge propone realizar el mantenimiento de la ontología como parte del mantenimiento del sistema software. La metodología propone implantar una nueva versión después de probar posibles efectos en la aplicación, pero no proporciona guías sobre cómo gestionar diferentes versiones ni cuándo crear nuevas versiones.

**DILIGENT**

La metodología DILIGENT (Pinto et al. (2004)) está orientada al soporte de expertos de dominio en un entorno distribuido para ingenieros y ontologías de desarrollo. Está enfocada al desarrollo colaborativo de ontologías. No obstante, en la metodología no se considera la noción de contexto.

El modelo de ciclo de vida de las ontologías propuesto en la metodología consta de 5 actividades principales (Engler et al. (2006)) como se puede observar en la Figura 3.13:

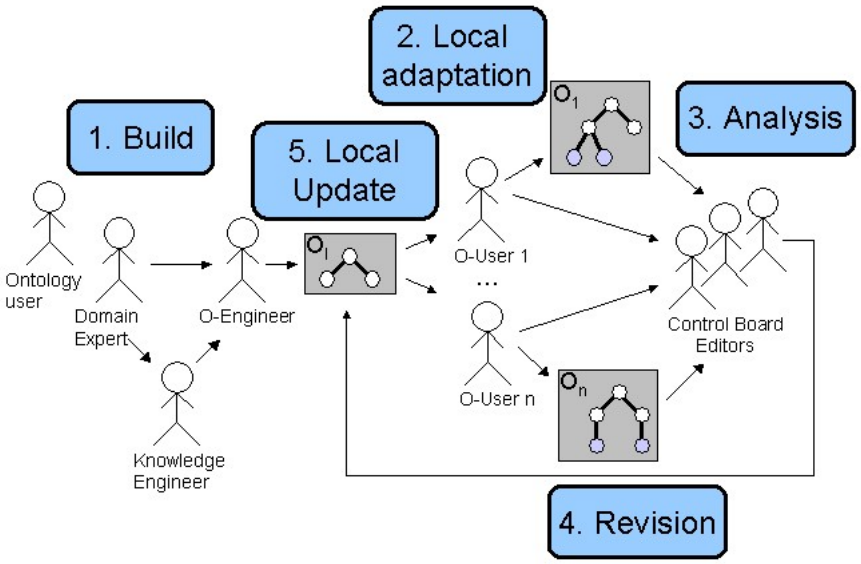


**Figura 3.13:** Modelo de Ciclo de Vida en la metodología DILIGENT (Engler et al. (2006))

1. *Build*. En esta actividad, los expertos del dominio, usuarios, ingenieros de conocimiento e ingenieros de ontologías colaborativamente crean una versión inicial de la ontología. El equipo involucrado en la construcción de la ontología inicial debería ser relativamente pequeño para lograr con mayor facilidad una pequeña y consensuada primera versión de la ontología compartida. No se requiere la completitud de esta ontología inicial con respecto al dominio.

Para la construcción de la versión inicial de la ontología, DILIGENT no propone realizar la actividad de especificación de la ontología ni tiene en cuenta la reutilización y reingeniería de recursos de conocimiento existentes.

2. *Local adaption*. Una vez disponible la ontología compartida, los usuarios pueden comenzar a usarla y adaptarla localmente para sus propósitos específicos. Para ello, podrán cambiar la copia local de la ontología compartida de tal modo que, estos cambios locales no afectan a otros usuarios de la ontología. Todos los cambios locales son recogidos por un comité de control central.

3. *Análisis*. El comité de control analiza en esta fase todos los cambios realizados en la fase 2 por los usuarios en sus copias locales de la ontología compartida. Luego, decide qué cambios locales serán introducidos en la siguiente versión compartida. Para ello, la tarjeta de control intenta identificar similaridades en las ontologías de usuarios ya que no todas estas ontologías deberían ser fusionadas. La meta de esta fase es desarrollar una ontología central compartida tal que, su tamaño no crezca rápidamente y se vuelva insostenible.

En lo referente a la dimensión dinámica, DILIGENT propone la creación de diferentes versiones de la ontología pero no proporciona guías sobre cómo gestionar las versiones, ni sobre cuándo crear diferentes versiones, ni cómo tales cambios pueden afectar a las diferentes versiones.

4. *Revision*. Basada en la fase previa de análisis, se crea y distribuye una nueva revisión de la ontología compartida. Se requieren revisiones regulares de esta ontología para evitar una gran divergencia de las ontologías locales de la ontología compartida. De este modo, es necesaria una decisión equilibrada teniendo en cuenta las diferentes necesidades de los usuarios y sus requerimientos.

5. *Local update*. En este paso final, los usuarios de las ontologías compartidas actualizan sus copias locales teniendo en cuenta la última revisión de la ontología compartida. Ésta pueda contener varios de los cambios que se introdujeron en la fase 2 pero otros cambios podrían haberse perdido ya que el comité de control intenta equilibrar las diferentes necesidades de los usuarios y no siempre asume los cambios cuando éstos se producen. Sin embargo, el usuario debería reutilizar los nuevos conceptos en vez de usar sus conceptos definidos previamente de forma local para beneficiarse del nuevo desarrollo de la ontología compartida.

En general, la metodología propone un modelo de ciclo de vida de ontologías que está basado en la idea de evolución de prototipos.

La principal aportación de la metodología DILIGENT es un marco de discusión que facilita el debate sobre el diseño racional de los cambios que se introducen en las diferentes fases del ciclo de vida. En especial, en las fases 3 y 4 (Analysis y Revision), las discusiones intercambiadas ayudan al comité de control a comprender las razones específicas de los cambios (Engler et al. (2006)). Otros detalles del esquema de discusión de DILIGENT y cómo facilita el desarrollo colaborativo de ontologías pueden verse en (Dellschaft et al. (2008)).

**Metodología NeOn para la construcción de Redes de Ontologías**

Algunas de las aproximaciones metodológicas reconocidas se han presentado previamente: METHONTOLOGY, On-To-Knowledge y DILIGENT. Estas metodologías han proporcionado guías a los investigadores para el desarrollo de ontologías, pero presentan las siguientes limitaciones:

- Las metodologías carecen de guías para construir ontologías reutilizando y/o realizando reingeniería de otras ontologías y los recursos de conocimiento existentes ampliamente consensuados en un dominio particular.

- Las metodologías carecen de guías para contextualizar una ontología existente integrándola con otras ontologías ya existentes, que pudieran estar en continúa evolución.

- Las metodologías no explican el proceso de construcción de ontologías con el mismo estilo y granularidad que las metodologías para desarrollar software.

En el proyecto de investigación NeOn [16], se propone una metodología para construir redes de ontologías denominada NeOn que, como se ha mencionado previamente, intenta superar las limitaciones identificadas en otras metodologías analizadas previamente pero, a la vez, intenta beneficiarse de las ventajas ya existentes en ellas.

[16] http://www.neon-project.org

Una red de ontologías es una colección de ontologías conectadas a través de diversas relaciones tales como mapeo, modularización, versión, y dependencia.

Para construir la metodología se usa la estrategia de divide y vencerás. De este modo, se descompone el problema general a resolver, el desarrollo de una red de ontologías, en diferentes subproblemas tal que, la solución al problema general se obtiene combinando las soluciones a los diferentes subproblemas. En el caso de la metodología, los subproblemas son los procesos y actividades[17] identificadas en los nueve escenarios descritos en Suárez-Figueroa et al. (2007) y en Suárez-Figueroa et al. (2008a), y presentados en la Figura 3.12. Cada uno de estos escenarios en la metodología NeOn son, en definitiva, las diferentes formas alternativas de construir ontologías y redes de ontologías. A continuación se describen brevemente estos escenarios:

- ***Escenario 1***: *Contrucción de redes de ontologías desde el principio sin reutilizar recursos de conocimiento.*
- ***Escenario 2***: *Construcción de redes de ontologías mediante la reutilización y reingeniería de recursos no ontológicos.* En este escenario, los desarrolladores software y profesionales de ontologías deberían analizar si existen recursos no ontológicos que contengan ya terminología consensuada (disponible en glosarios, diccionarios, léxicos, esquemas de clasificación, tesauros, etc.), que pueda ser reutilizada para construir una red de ontologías. Si se decide que uno o varios recursos no ontológicos son útiles para el desarrollo de la ontología, entonces el proceso de reingeniería de recursos no ontológicos debería llevarse a cabo. Esta reutilización de terminología consensuada permite ahorrar tiempo y dinero en el proceso de desarrollo de ontologías y promover la aplicación de buenas prácticas.
- ***Escenario 3***: *Construcción de redes de ontologías mediante la reutilización de recursos ontológicos.* En este escenario, los desarrolladores software y profesionales de ontologías deberían analizar si ya existen recursos ontológicos que puedan ser reutilizados para construir una red de ontologías o no. El principio subyacente, como en el escenario 2, es que la reutilización de estos recursos ahorra tiempo y costes en el desarrollo de las ontologías.
- ***Escenario 4***: *Construcción de redes de ontologías mediante la reutilización y reingeniería de recursos ontológicos.*
- ***Escenario 5***: *Construcción de redes de ontologías mediante la reutilización y combinación (*merging*) de recursos ontológicos.*
- ***Escenario 6***: *Construcción de redes de ontologías mediante la reutilización, combinación y reingeniería de recursos ontológicos.*
- ***Escenario 7***: *Construcción de redes de ontologías mediante la reutilización de patrones de diseño de ontologías*[18]. El análisis así como una descripción de las guías metodológicas propuestas en la metodología NeOn para realizar la reutilización de patrones de diseño pueden verse en Suárez-Figueroa et al. (2008a), Suárez-Figueroa et al. (2008b), y Suárez-Figueroa et al. (2009).

[17] La terminología usada para proceso y actividad en la metología NeOn es la mencionada en la sección 3.3.6. Las actividades, a su vez, pueden estar divididas en cero o más tareas.

[18] La reutilización de patrones de diseño de ontologías se define en Suárez-Figueroa et al. (2008b) como *the activity of using available ontology design patterns in the solution of different modeling problems during the development of new ontologies.* Los patrones de diseño es una técnica emergente para la reutilización de experiencias de codificado y mejores prácticas

- ***Escenario 8***: *Construcción de redes de ontologías mediante la reestructuración de recursos ontológicos.*
- ***Escenario 9***: *Construcción de redes de ontologías mediante localización de recursos ontológicos*[19]. El análisis y descripción de las guías metodológicas propuestas en la metodología NeOn para realizar esta actividad pueden verse en (Suárez-Figueroa et al. (2009)).

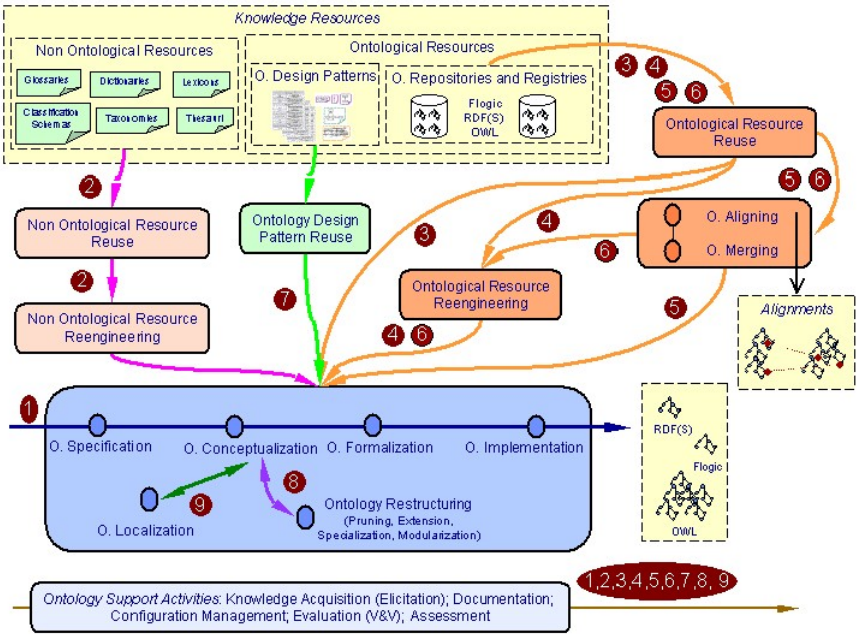


**Figura 3.14:** Escenarios para construir Redes de Ontologías (Suárez-Figueroa et al. (2008a))

Los procesos y actividades en la Figura 3.14 son representados por círculos coloreados o por cajas redondeadas. Las flechas dirigidas con los círculos numerados asociados representan los diferentes escenarios de la metodología mencionados anteriormente. Las cajas con líneas discontinuas representan en la figura los recursos de conocimiento existentes para ser reutilizados y las posibles salidas (redes de ontologías implementadas y mapeos) que resultan de la ejecución de alguno de los escenarios presentados.

Las actividades de *acquisition* y *elicitation*, *documentation*, *configuration management*, *evaluation* y *assessment* (mostradas en la parte inferior de la Figura 3.14) serán realizadas durante todo el desarrollo de la red de ontologías.

Los escenarios de la metodología pueden combinarse de diferentes formas. Por ejemplo, el escenario 2 (reutilización y reingeniería de recursos no ontológicos) puede combinarse con los escenarios 3-8; y el escenario 9 (localizar ontologías) puede realizarse o no con los escenarios 1-8. De todos modos, el conjunto de los nueve escenarios considerados como las formas más plausibles de construcción de redes de ontologías no tiene porqué ser considerados exhaustivos.

De todos los escenarios considerados, el escenario 1 es el más típico para construir ontologías y redes de ontologías desde el comienzo sin reutilizar recursos de conocimiento existentes.

[19] *Ontology Localization* se define en (Suarez Figueroa and Gómez-Pérez (2008)) como *the activity that consists in adapting an ontology to a particular language and culture.* La salida de la actividad es una ontología (ontología multilingüe) cuyos términos expresados en un lenguaje natural fuente han sido traducidos a un lenguaje natural destino. Las traducciones resultantes se añaden como etiquetas disponibles de la ontología original

Sin embargo, cada vez más desarrolladores de ontologías construyen ontologías y redes de ontologías por medio de la reutilización de recursos de conocimiento existentes. La metodología NeOn distingue escenarios que implican reutilizar recursos ontológicos de aquéllos que implican la reutilización y reingeniería de recursos no ontológicos.

Basado en la definición de reutilización en ingeniería del software (iee), se puede definir *reutilización de conocimiento* como el uso de cualquier recurso de conocimiento (ontológico o no ontológico) en la solución de diferentes problemas, como por ejemplo, en la construcción de nuevas ontologías o en el desarrollo de aplicaciones basadas en ontologías.

La reutilización de patrones de diseño de ontologías también fue añadida como un escenario diferente en la metodología NeOn (escenario 7), porque estos patrones son elementos claves durante el diseño de la ontología, que destacan de otros tipos de recursos ontológicos.

### Crítica a las metodologías propuestas

Las metodologías METHONTOLOGY y On-To-Knowledge pueden considerarse hasta la aparición de la metodología NeOn como las metodologías más completas para construir ontologías desde su comienzo, proporcionando guías para la construcción de ontologías sencillas desde su especificación a su implementación.

**Tabla 3.3:** Análisis comparativo de las metodologías METHONTOLOGY, On-To-Knowledge, DILIGENT y NeOn (Suárez-Figueroa (2010))

| | METHONTOLOGY | On-To-Knowledge | DILIGENT | NeOn Methodology |
|---|---|---|---|---|
| **Dimension** | | | | |
| ***Collaboration*** | Not mentioned | Not mentioned | Treated | *Mentioned, but not treated in detail* |
| **Detailed Guidelines for Processes and Activities** | | | | |
| ***Ontology Requirements Specification*** | Not provided<br>Only Competency Questions are proposed | Not provided<br>Only Competency Questions are proposed | This activity is not proposed by the methodology | *Provided* |
| ***Scheduling*** | Not provided | Not provided | Not provided | *Provided* |
| ***Reusing Ontological Resources*** | Not provided<br>Only a list of activities to be carried out is proposed | Not provided<br>Only recommendation of identifying ontologies to be reused is given | Not provided, neither explicitly mentioned | *Provided for ontologies and ontology statements* |
| **Audience** | | | | |
| ***Targeted to Software Developers and Ontology Practitioners*** | Targeted to ontology engineers and researchers | Targeted to ontology engineers and researchers | Intended to domain experts and users | *Targeted to ontology practitioners, general public, and software engineers* |

La tabla 3.3 resume las metodologías presentadas (METHONTOLOGY, On-To-Knowledge, DILIGENT y NeOn) de acuerdo a las siguientes características:

- *Collaboration*: las metodologías METHONTOLOGY y On-To-Knowledge no consideran la ingeniería de ontologías distribuida entre los grupos heterogéneos y distribuidos geográficamente de expertos del dominio y profesionales de ontologías. DILIGENT sí lo trata, pero sólo proporciona un marco de argumentación amplio y rico para construir rápidamente una ontología sencilla y mantener la pista de todas las discusiones relevantes sobre la actividad de conceptualización (Engler et al. (2006)).

- Grado de cobertura de algunos de los procesos o actividades considerados en la metodología NeOn (ontology requirements specification, scheduling y ontological resource reuse), de acuerdo a si proporcionan o no guías detalladas para ellos. Como se observa en la tabla 3.3, ninguna de las tres primeras metodologías (METHONTOLOGY, On-To-Knowledge y DILIGENT) proporciona guías detalladas para los procesos o actividades. Se puede decir que son metodologías más descriptivas que preceptivas porque no proporcionan instrucciones de cómo realizar los procesos o actividades.

  - Las tres metolodologías METHONTOLOGY, On-To-Knowledge y DILIGENT proponen métodos sencillos para llevar a cabo la actividad de especificación de requerimientos de ontologías, que consiste en pasos de alto nivel. Por lo tanto, estas metodologías no proporcionan guías detalladas que expliquen cómo realizar cada paso y qué es necesario para obtener un buen Documento de Especificación de Requerimientos de Ontología (ORSD -*Ontology Requirements Specification Document*), y cómo el ORSD puede usarse posteriormente en el desarrollo de la ontología (por ejemplo, para buscar los recursos de conocimiento para ser reutilizados o, para verificar el contenido de la ontología). En NeOn, se propone unas guías metodológicas para especificar requerimientos de ontologías (Suárez-Figueroa (2010)) basadas en las denominadas cuestiones de competencias y en una plantilla para escribir el documento ORSD.
  - Las actividades de planificación son extremadamente importantes para establecer el orden y el tiempo de los procesos y actividades implicados en el desarrollo de redes de ontologías. Ninguna de las tres primeras metodologías: METHONTOLOGY, On-To-Knowledge, y DILIGENT, proporcionan unas guías prescriptivas detalladas para seleccionar un modelo de ciclo de vida específico para crear un ciclo de vida de ontologías particular, como parte de la actividad de planificación. Sin embargo, NeOn incluye: a) las bases metodológicas para establecer un ciclo de vida concreto para una red de ontologías como parte de la actividad de planificación (basándose en el Glosario de Procesos y Actividades de NeOn, en el conjunto de escenarios identificados en la metodología NeOn, y en el conjunto de modelos de ciclo de vida), b) una herramienta llamada gOntt que soporta la planificación y ejecución de proyectos de desarrollo de redes de ontología, y c) las guías metodológicas para planificar proyectos de desarrollo de redes de ontologías usando gOntt.
  - Sólo METHONTOLOGY considera que las actividades realizadas durante el desarrollo de una ontología pueden implicar llevar a cabo otras actividades en otras ontologías ya construidas o en construcción (Fernández López et al. (2000)). DILIGENT no tiene en cuenta la reutilización de la información y recursos de conocimiento disponibles para acelerar el proceso de construcción de ontologías, On-To-Knowledge sólo considera el uso de métodos de aprendizaje de ontologías a partir de recursos textuales para reducir los esfuerzos en el desarrollo de ontologías, y METHONTOLOGY propone un método general para reusar ontologías, pero no incluye guías metodológicas prescriptivas. Así pues, de estas tres metodologías, sólo METHONTOLOGY proporciona una definición y unas guías iniciales para reingeniería de ontologías (Fernández López et al. (2000), Gómez Pérez and Rojas Amaya (1999)).

    Sin embargo, la metodología NeOn incluye guías metodológicas para reutilizar recursos ontológicos (Suárez-Figueroa (2010)), entre otros, que intentan resolver la fal-

ta de guías prescriptivas detalladas para reutilizar recursos ontológicos en diferentes niveles de granularidad (como un todo, por módulos o por estados).

- Sólo METHONTOLOGY de las tres primeras metodologías, como se observa en la tabla 3.3, se describe dirigida a desarrolladores software y profesionales de ontologías, y no sólo hacia investigadores de ontologías. Sin embargo, son necesarias guías que ayuden a los desarrolladores de software y a los profesionales de ontologías a seleccionar un modelo de ciclo de vida específico, y así a crear un ciclo de vida de ontologías particular, como parte de la actividad de planificación. Estas guías dirigidas a profesionales de ontologías, público en general, y a ingenieros de sofware, como ya se ha mencionado previamente, sí son proporcionadas por NeOn.

La metodología NeOn, actualmente en su tercera versión (Suárez-Figueroa et al. (2010)), incluye los beneficios proporcionados por DILIGENT sobre colaboración desde el punto de vista de sus dimensiones. Además, se tienen en cuenta las propuestas proporcionadas por METHONTOLOGY y On-To-Knowledge sobre el uso de preguntas de competencias para la actividad de especificación de la ontología y respecto a la reutilización de ontologías, se mejoran y se extienden la lista de actividades propuesta por METHONTOLOGY en las líneas metodológicas de la primera versión de la metodología NeOn.

CAPÍTULO 4

# PLANTEAMIENTO DEL PROBLEMA Y SUPUESTOS DEL TRABAJO

En el presente capítulo se describen, en primer lugar, las necesidades generales que han impulsado la línea de investigación de este trabajo, los objetivos específicos que se pretenden, así como los requisitos previos que se han fijado para alcanzarlos. Una vez planteado el problema, a continuación se han de marcar las hipótesis y premisas del trabajo que deben ser asumidas en su resolución.

## 4.1 Planteamiento del problema

El problema a tratar en esta tesis se origina

### 4.1.1 Objetivos del trabajo

El objetivo de este trabajo busca avanzar en el estado del arte del campo de los SITs, especialmente en el área del Modelado del Estudiante, desarrollando un ME basado en un enfoque pedagógico, que se caracterice por ser fácilmente adaptable, extensible, y reutilizable para su aplicación a diferentes dominios de aprendizaje, y con un método de diagnóstico pedagógico-cognitivo no monótono.

Del objetivo general enunciado se han extraído los siguientes objetivos específicos:

- **O1**: *Desarrollar un ME con una amplia taxonomía de conocimiento que posibilite expresar muchos tipos de conocimientos sobre el estudiante.*

  Esta información permitirá al módulo de tutoría realizar una tutoría más personalizada, dependiendo de las características intrínsecas del estudiante y estados temporales, así como de su estado de conocimiento. Una importante nueva característica, incluida ex novo en el modelo, es la representación explícita de los objetivos de aprendizaje que el estudiante debería alcanzar en diferentes dominios del aprendizaje (cognitivo, psicomotor y

afectivo). La nueva taxonomía pretende facilitar el diseño de un nuevo método de diagnóstico basado, no sólo en el modelo del conocimiento del estudiante, sino también en sus características personales, su estado emocional, sus acciones, el grado de consecución de los objetivos, y contribuir, al mismo tiempo, a proporcionar mejores explicaciones y ayudas durante el proceso de aprendizaje. Además, la taxonomía incluirá la valoración de diversos aspectos del aprendizaje del estudiante. Esta información será muy útil, tanto para el proceso de diagnóstico como para el módulo de tutoría.

- **O2**: *Desarrollar un ME con un potente formalismo de representación.*

  La finalidad no es otra sino desarrollar un ME con un formalismo que permita una representación de los conceptos con diferentes niveles de abstracción, y que sirva de soporte para compartir y reutilizar conocimientos.

- **O3**: *Desarrollar un nuevo método de diagnóstico pedagógico-cognitivo.*

  En este trabajo se ha considerado, como parte clave del diagnóstico cognitivo, lo que denominamos diagnóstico pedagógico (sección 1.2), de tal modo que, el ME a desarrollar debe ser capaz de inferir el estado de los objetivos de aprendizaje (alcanzados o no por el estudiante) a partir de su comportamiento. A partir de este diagnóstico pedagógico, además, el método debe ser capaz de proporcionar también un diagnóstico cognitivo, es decir, un diagnóstico que sea capaz de inferir el estado de los conocimientos del estudiante a partir de su comportamiento.

- **O4**: *Desarrollar un nuevo método de diagnóstico con capacidades de razonamiento no monótono*, acorde con la naturaleza, también no monótona, del razonamiento humano.

  Para realizar su tarea, el tutor maneja información incompleta: la propia que se extrae del estado de los conocimientos del estudiante (es decir, lo que se supone que sabe o no sabe en cada instante el estudiante). El responsable de proporcionar esta información debe ser el método de diagnóstico incluido en el modelo del estudiante. No obstante, el conocimiento del estudiante en el momento de abordar un problema es frecuentemente incompleto, a pesar de lo cuál debe continuar el razonamiento durante su aprendizaje. Por ello, el método de diagnóstico debería también considerar tal laguna formulando hipótesis que le permitan avanzar en dicho razonamiento. Junto con ello, también debería manejar la naturaleza no monótona del razonamiento del estudiante que interactúa con el SIT; el estudiante aprende nuevos conceptos u olvida otros ya adquiridos durante su proceso de aprendizaje, lo que podría implicar que surjan contradicciones, y que alguna de la suposiciones realizadas previamente resulten equivocadas, así como lo inferido a partir de ellas. A este razonamiento se le denomina razonamiento no monótono y el mecanismo capaz de detectar estas contradicciones y deshacer sus inferencias es la revisión de creencias.

  De este modo, el método podrá manejar el conocimiento incompleto propio del estado cognitivo del alumno.

- **O5**: *Desarrollar un ME flexible y fácilmente adaptable* a diferentes SITs.

  Este objetivo se ha de lograr con la especialización del ME, en concreto, para su uso en Entornos Virtuales de Aprendizaje y, más específicamente aún, en Entornos Virtuales de Entrenamiento/Formación procedimental.

Esta especialización conllevará no sólo la incorporación de nuevos tipos de conocimientos intrínsecos a este tipo de entornos, sino también el desarrollo de un método de diagnóstico más rico, que sea capaz de inferir más información sobre el estado del estudiante.

A continuación, se establecen los requisitos derivados, por un lado, del análisis del estado del arte de los SITs clásicos (apartado 2.1) y, por otro lado, de la necesidad de integración del modelado del estudiante en un EVIE, en concreto, en MAEVIF (2.4.8).

### 4.1.2 Requisitos derivados del análisis de SITs clásicos

Tras el análisis del estado del arte de los MEs y el DC, descrito en el capítulo 2, "Estado de la Cuestión", se ha podido comprobar que la mayoría de las aproximaciones no parten de una taxonomía que ahonde en los diferentes tipos de conocimientos sobre el estudiante (objetivos de aprendizaje; trazas de su comportamiento; objetos de conocimiento sobre la materia a enseñar; estado de los conocimientos sobre el estudiante, tales como el estado de objetivos -alcanzados o no-, características individuales del estudiante, etc.) que, a la vez, sirva de soporte para realizar un diagnóstico cognitivo profundo y con capacidades de razonamiento no monótono.

Estas limitaciones observadas en el estado del arte conforman una de las razones que lleva a que la presente plantee el desarrollo de un ME que intente mejorar las insuficiencias de los MEs previos. De este modo, y para alcanzar los objetivos del trabajo (sección 4.1.1), se han definido los siguientes requisitos:

- **R1**: *Representación de una taxonomía amplia de objetos de conocimiento en el ME.*

  Los *objetos de conocimiento* constituyen uno de los tipos de conocimientos acerca del estudiante más ampliamente incorporados por los modelados del estudiante surgidos en el campo de los SITs (apartado 2.5.1). El método de diagnóstico de modelado del estudiante debe proporcionar al tutor información sobre lo que el estudiante sabe o no sabe en cada momento, lo que implica que en el ME se deban considerar objetos de conocimiento comunes, tanto de tipo estructural (conceptos, relaciones, etc.), como de tipo procedimental (acciones, planes, reglas, etc.). Acorde con la idea de granularidad (Wielinga et al. (1992)) aplicada en diferentes propuestas de MEs, la taxonomía de objetos de conocimiento a incluir deberá también permitir diferentes niveles de abstracción de conocimientos, lo que, a su vez, facilitará el diagnóstico cognitivo en diferentes niveles (grano más fino o más grueso).

  Este requisito está relacionado con los objetivos O1, O3, O4, y O5.

- **R2**: *Representación de una taxonomía de objetivos de aprendizaje en el ME.*

  Una características fundamental a incluir en el ME es la representación explícita de los tipos de *objetivos de aprendizaje* que puede llegar a alcanzar el alumno en diferentes ámbitos (cognitivo, afectivo y psicomotor).

  Hasta la actualidad se han realizado escasas aproximaciones que procuren la integración de este tipo de conocimiento en el modelado del estudiante (una excepción es, por ejemplo, la de (Dongming Xu and Wang (2005), en la que, no obstante, no se especifican diferentes tipos de objetivos). Sin embargo, dicha integración permitirá realizar un diagnóstico a nivel de objetivos (alcanzados o no por el estudiante), y por lo tanto, el desarrollo de un *ME funcional* con más ventajas frente a los modelos de simulación de comportamiento (véase sección ).

La flexibilidad del método de diagnóstico se verá mejorada con la incorporación de objetivos en otros ámbitos distintos al cognitivo, afectivo y psicomotor, lo que permitirá aplicar el modelo en SITs relacionados con la enseñanza de materias cuyo diseño incluya este tipo de objetivos. Asimismo, será especialmente útil en el diseño de materias que impliquen establecer objetivos especialmente en esos dos dominios (por ejemplo, a nivel afectivo, enseñar a diferentes profesionales/estudiantes: médicos, bomberos, etc., a controlar ciertas situaciones críticas, de elevado estrés o peligro o, a nivel psicomotor, enseñar en áreas relacionadas como la danza, el teatro, la educación física, etc.) Todo ello ampliaría sustancialmente el ámbito de aplicación del modelado del estudiante propuesto.

Este requisito está relacionado con los objetivos O1, O3, O4, y O5.

- **R3**: *Representación de una taxonomía del perfil del estudiante en el ME.*

  La representación del conocimiento sobre el *perfil del estudiante* permitirá al módulo de tutoría realizar una tutoría más adaptable, en función de las características propias de cada alumno y de sus estados transitorios en un momento dado. Tal y como se indicó en el análisis crítico de las taxonomías utilizadas por los MEs realizados hasta la fecha (sección 2.5.1), este tipo de conocimiento se ha incorporado ya en diversos MEs propuestos, si bien limitadamente en su representación y, por lo tanto, en su posterior explotación por parte del diagnóstico cognitivo del estudiante.

  Este requisito está relacionado con los objetivos O1, O3, O4, y O5.

- **R4**: *Representación de una taxonomía del estado de los conocimientos del estudiante en el ME.*

  El diagnóstico cognitivo implica inferir el *estado de los conocimientos del estudiante* a partir de su comportamiento.

  En el análisis del estado del arte del ME no se han encontrado demasiadas propuestas que consideren esta parcela de conocimiento en profundidad (una excepción, por ejemplo, es la ontología OMNIBUS (Mizoguchi et al. (2007))). Sin embargo, consideramos de especial relevancia el registro de este conocimiento, el cuál contiene información diversa sobre un estudiante: el estado o grado de finalización del curso en términos de unidades de diseño formativas (fases, actividades, etc.); algunos datos acumulativos derivados de la actuación del estudiante (número de errores, número de pistas proporcionadas, etc.), que representan el estado de ejecución de una cierta actividad; su estado emocional; el estado de los objetivos (adquiridos o no por el estudiante); el estado de los objetos de conocimiento (el estudiante puede estar en posesión de ellos o no); etc.

  El diagnóstico de este tipo de conocimiento sobre el alumno facilitará también la tarea al tutor, quien podrá así proporcionar a cada estudiante particular ayudas, explicaciones adicionales, etc., mejor adaptadas al estado de sus conocimientos en un momento determinado de su aprendizaje.

  Este requisito está relacionado con los objetivos O1, O3, O4, y O5.

- **R5**: *Representación de una taxonomía del estado de la traza del estudiante en el ME.*

  Otro tipo de conocimiento que deberá considerarse en la taxonomía de conocimientos del ME es la *traza del estudiante* o registro temporal de la historia pasada del comportamiento del estudiante. Este registro engloba la traza de las acciones llevadas a cabo o la del estado de los objetivos, entre otras.

La información acerca de la traza del estudiante puede contribuir al mismo tiempo a que el módulo de tutoría proporcione un soporte más adaptable al estudiante durante su aprendizaje. Por ejemplo, la estrategia de tutoría que se debería proporcionar a un estudiante que se ha equivocado, no será probablemente la misma si es la primera vez -o no- que realiza una cierta actividad y comete el mismo error. En el primer caso, es decir, si es la primera vez que comete el error, la estrategia del tutor, con toda probabilidad, tenderá a proporcionar niveles de pistas, explicaciones o refuerzos distintos que en el segundo caso, en el que, incluso, podría recomendar al alumno abandonar la sesión con el fin de que pueda repasar lecciones previas aparentemente olvidadas.

Este requisito está relacionado con los objetivos O1, O3, O4, y O5.

- **R6**: *Uso de ontologías* que ayude a formalizar el proceso de construcción de un ME fácilmente extensible, adaptable y reutilizable.

  Los beneficios, en este aspecto, son múltiples (véase apartado 3.3) (por ejemplo, existencia de lenguajes de ontologías estandarizados), así como los avances, que en general, surgen en el ámbito de la ingeniería ontológica (metodologías; herramientas como editores, razonadores, ...; etc.). Todas estas características ayudarán a que el modelo a desarrollar sea, en un futuro, fácilmente extensible y adaptable a diferentes SITs, y facilitarán compartir y reutilizar el conocimiento del ME en diferentes entornos de aprendizaje. Asimismo, el uso de ontologías permitirá representar explícitamente diferentes niveles de granularidad del conocimiento (Wielinga et al. (1992)).

  Este requisito está relacionado con el objetivo O2 y O5.

- **R7**: *Proporcionar un nuevo método de diagnóstico pedagógico-cognitivo fácilmente adaptable.*

  El diagnóstico del estado del conocimiento del estudiante habrá de enfocarse desde una perspectiva pedagógica, realizándose, en una primera fase, el diagnóstico de los objetivos de aprendizaje alcanzados -o no- por el estudiante durante su aprendizaje a partir de su comportamiento. Será después, a partir de los resultados obtenidos por el método de diagnóstico pedagógico, cuando se podrá obtener un diagnóstico cognitivo (diagnóstico de los objetos de conocimiento adquiridos -o no- por el estudiante a partir de su comportamiento).

  El cumplimiento de este requisito permitirá dotar al método de diagnóstico de mayor flexibilidad, no limitándolo exclusivamente a un diagnóstico cognitivo (el más habitual en los modelados analizados). Un tutor podrá basar la estrategia de tutoría con un cierto alumno, en un diagnóstico combinado de objetivos alcanzados -o no-, y de ciertos objetos de conocimiento adquiridos -o no-, por el estudiante (diagnóstico pedagógico, de grano más grueso, y; diagnóstico cognitivo, de grano más fino).

  Adoptar prioritariamente un diagnóstico pedagógico tiene las siguientes ventajas principales:

  - Las conclusiones obtenidas por el diagnóstico pueden ser utilizadas directamente por un Agente de Tutoría, toda vez que están expresadas en términos de estados de objetivos (es decir, objetivos alcanzados o no alcanzados por el estudiante). Estos objetivos, definidos en el diseño formativo de la materia que se está enseñando/aprendiendo, y registrados en la ontología del ME, son los que, precisamente,

tendrá en cuenta el tutor al decidir cuál va a ser la siguiente actividad que debe realizar el estudiante o qué nivel de explicación debería mostrarle.

- A partir de este enfoque pedagógico también se obtendrá un diagnóstico cognitivo. El diagnóstico pedagógico, junto con el diagnóstico cognitivo, permitirán obtener mucha más información, y en diferentes niveles de detalle o granularidad, sobre el estado de los conocimientos del alumno.

Dadas las ventajas anteriores, y puesto que no existen hasta la actualidad MEs que puedan realizar ambos tipos de diagnóstico, se considera altamente beneficioso realizar este tipo de diagnóstico, lo que facilitaría al Agente de Tutoría llevar a cabo una tutoría más personalizada y más eficaz.

Además, el método de diagnóstico deberá caracterizarse por su fácil adaptabilidad a otros SITs, es decir, la aplicación del ME a otro SIT no deberá ser compleja ni de elevado coste.

Este requisito está relacionado con los objetivos O3 y O5.

- **R8**: *Proporcionar un nuevo método de diagnóstico pedagógico con capacidades de razonamiento no monótono.*

  A pesar de que la no monotonía en el modelado del estudiante no ha sido tratada ampliamente (véase el apartado 2.5.2), consideramos que, como ya se vio en la sección 1.2, es un aspecto clave para desarrollar un modelado del estudiante adaptativo y flexible.

  Entre las aproximaciones a la revisión de creencias, los Sistemas de Mantenimiento de la Verdad de tipo ATMS, tal y como se vio en la sección 3.1.5, son muy adecuados para su aplicación al DC. Todos los trabajos al respecto (Martins (1991)) han demostrado ser muy válidos en diferentes ámbitos (monitorización en el lanzamiento de naves espaciales Semmel et al. (2005) o análisis de imágenes médicas Rake and Smith (1988), así como, también, en el campo educativo (Kono et al. (1994), Jones et al. (2000), entre otros). Por esta razón, el ATMS es la aproximación elegida para tratar la no monotonía en el método de DC.

  Es importante señalar que los tipos de conocimiento del estudiante mencionados en los requisitos R1, R2, R3, R4, y R5, están estrechamente relacionados con el método de diagnóstico no monótono, especialmente: el *estado de conocimiento del estudiante* y la *traza del estudiante*. El papel de estos dos tipos de conocimientos es esencial, para poder detectar y resolver los diferentes tipos de contradicciones que pueden surgir durante el proceso de diagnóstico del conocimiento del estudiante. Por ejemplo, permitirán discernir errores debidos a olvidos del estudiante (un olvido puede surgir cuando el estudiante logra alcanzar un objetivo en una sesión de aprendizaje que transcurrió hace mucho tiempo, y en la actualidad no lo consigue) o inconsistencias debidas a cambios en su mente (por ejemplo, el estudiante recibe una pista del tutor, que le ha conducido al logro de un cierto objetivo que anteriormente no había alcanzado por sí solo). También son importantes los rasgos de personalidad (conocimiento que forma parte del perfil del estudiante), tales como si el estudiante es olvidadizo, despistado, etc.

  Este requisito está relacionado con el objetivo O3 y O4.

### 4.1.3 Requisitos derivados de la integración en un EVIE (MAEVIF)

Con el fin de demostrar su aplicabilidad y su validez para su uso en Entornos Virtuales de Aprendizaje -más en concreto, en Entornos Virtuales de Entrenamiento/Formación de tipo procedimental-, el mecanismo de modelado del estudiante a desarrollar se integrará en MAEVIF (véase apartado 2.4.8) y, específicamente, en un agente ya previsto en la plataforma, el *Agente de Modelado del Estudiante*. Por lo tanto, los requisitos que se impongan sobre este agente, serán también requisitos para el modelo propuesto. Además, la aplicación a estos entornos más complejos implica nuevas necesidades añadidas a las ya mencionadas para SITs clásicos, entre las que se pueden destacar las siguientes:

- El conocimiento intrínseco a este tipo de entornos, relacionado con su estructura y su naturaleza espacial, debe ser añadido al ME -escenarios, subescenarios y conexiones entre ellos-, así como la consideración de objetos virtuales que pueden hallarse en él (manipulables o no manipulables) y sus posiciones (absolutas, relativas a la posición de otros objetos o relativas a un subescenario).

- En el EV el estudiante puede realizar acciones de diferentes tipos, tales como coger objetos, pulsar botones, etc. Todas estas acciones son enviadas en MAEVIF al Agente de Tutoría, y deberán ser registradas por el Agente de Modelado del Estudiante en forma de una traza de la actividad del estudiante. Adicionalmente, cada vez que el estudiante ejecuta una acción en el EV (o simplemente intenta llevarla a cabo), el Agente de Modelado del Estudiante debe actualizar sus creencias sobre el estado de los conocimientos del estudiante, para poder informar al Agente de Tutoría acerca de los objetivos de aprendizaje que asume ya alcanzados -o no- por el estudiante. Por ejemplo, imaginemos que el estudiante tiene que realizar un experimento en un EV que represente un laboratorio de química: si en un instante determinado el estudiante intenta añadir ácido sulfúrico a un vaso de agua, y no se ha puesto guantes, entonces el Agente de Modelado del Estudiante puede inferir que el estudiante no sabe que es obligatorio ponerse guantes cuando maneja, como en este caso, ácido sulfúrico.

- Una de las posibles acciones en el EV puede ser moverse de una posición origen a una posición destino. El estudiante debería seguir una trayectoria razonablemente buena; ello implica que la calidad de las trayectorias seguidas por un estudiante destaquen como otra importante cuestión que el Agente de Modelado del Estudiante debe tener en consideración a lo largo de la sesión de aprendizaje. Una mala trayectoria puede revelar, por ejemplo, que el estudiante ha construido un mapa cognitivo equivocado de una cierta área del EV, lo que en una situación de emergencia en el mundo real podría ser peligroso. El Agente de Modelado del Estudiante debe ser el encargado de registrar las trayectorias del estudiante y, como parte del diagnóstico, evaluar su calidad teniendo en cuenta las características de la trayectoria óptima calculada por el Agente de Trayectorias (figura 2.55).

Aunque en el estado del arte referente al modelado del estudiante en EVIEs la mayoría de las propuestas no consideran en conjunto las nuevas posibilidades de interacción en estos entornos, éstas deben también ser tenidas en cuenta como fuentes de información relevante en el ME, extendiendo así los tipos de información ya considerados en los requisitos derivados de los SITs clásicos. Esto permitirá diagnosticar mejor el estado de conocimiento del estudiante y, consecuentemente, permitirá al tutor software seleccionar con mayor eficacia la estrategia de

tutoría más apropiada durante el proceso de aprendizaje. Por lo tanto, del análisis previo de necesidades en un EV como MAEVIF, surgen nuevos requerimientos respecto a la información que debe incluir el ME a desarrollar:

- **R9**: *Representación de objetos de conocimiento relativos a un EV en el ME.*

  Es necesario contemplar en el ME objetos relacionados con la estructura del EV; con los objetos virtuales del entorno, así como sus posiciones. Asimismo, también es necesario añadir objetos de conocimiento de tipo procedimental, tales como las acciones del estudiante que implican movimiento o los planes de aprendizaje.

  Este requisito está relacionado con los objetivos O1, O3, O4, y O5.

- **R10**: *Representación de la traza de la actividad del estudiante en el ME.*

  Se impone como requisito la necesidad de poder representar en el ME el registro de acciones (que incluyan movimientos); interacciones con objetos o avatares; nuevos tipos de preguntas que puede plantear el estudiante al módulo de tutoría; diferentes niveles de pistas o instrucciones directamente ligadas con el entorno que le rodea; etc.

  Este requisito está relacionado con los objetivos O1, O3, O4, y O5.

- **R11**: *Representación del estado de las trayectorias seguidas por el estudiante en el ME.*

  Este registro en el modelado del estudiante ya fue considerado como requisito necesario para un SIT clásico. Sin embargo, surge la necesidad, en un EV, de complementar este tipo de información con otros datos acumulativos, derivados de la traza de las trayectorias seguidas por el estudiante durante la ejecución de una actividad. Estos nuevos registros representarán el estado de esas trayectorias, permitirán evaluar la calidad de las mismas, y complementarán la valoración de la ejecución de la actividad realizada por el estudiante. Como ya se ha analizado anteriormente, esta información adicional es interesante a fin de que el módulo de tutoría pueda proporcionar al estudiante más y mejor ayuda.

  Este requisito está relacionado con los objetivos O1, O3, O4, y O5.

- **R12**: *El mecanismo de diagnóstico pedagógico a implementar, deberá ser ampliado con base en los nuevos elementos de conocimientos derivados de la integración en un EVIE.*

  Esta ampliación permitirá actualizar sus creencias sobre el conocimiento del estudiante y podría suponer ampliar la taxonomía de conocimientos considerada en el ME.

  Este requisito está relacionado con los objetivos O1, O3 y O5.

En la tabla siguiente se muestran, por filas, los requisitos de este trabajo, y, por columnas, los objetivos a alcanzar. Cada celda marcada (i, j) representa que el requisito Ri se define para alcanzar el objetivo Oj.

Tabla 4.1: Relación entre requisitos y objetivos establecidos

| Objetivos / Requisitos | O1 | O2 | O3 | O4 | O5 |
|---|---|---|---|---|---|
| R1 | √ | | √ | √ | √ |

Tabla 4.1: Relación entre requisitos y objetivos establecidos (cont.)

| Objetivos / Requisitos | O1 | O2 | O3 | O4 | O5 |
|---|---|---|---|---|---|
| R2 | √ | | √ | √ | √ |
| R3 | √ | | √ | √ | √ |
| R4 | √ | | √ | √ | √ |
| R5 | √ | | √ | √ | √ |
| R6 | | √ | | | √ |
| R7 | | | √ | | √ |
| R8 | | √ | √ | √ | |
| R9 | √ | | √ | √ | √ |
| R10 | √ | | √ | √ | √ |
| R11 | √ | | √ | √ | √ |
| R12 | √ | | √ | | √ |

## 4.2 Supuestos del trabajo

Una vez planteados los objetivos específicos, así como los requisitos que debe cumplir la solución al problema planteado, a continuación se establecen las principales hipótesis, premisas, y restricciones del trabajo.

- **H1**: *Los EVIEs a los que se aplique serán de tipo esencialmente procedimental.* El aprendizaje en este tipo de entornos se basa en la realización de procedimientos/tareas, en los que el alumno tiene que ejecutar una secuencia de acciones cuya meta es finalizar una actividad satisfactoriamente. Existe al menos un plan de actuación o secuencia de acciones, que el estudiante puede realizar para finalizar una actividad con éxito.

- **H2**: *El ME se aplicará a entornos que sigan un modelo de aprendizaje constructivista.* A diferencia de la mayoría de SITs e EVIEs anteriores, el ME se aplicará a entornos donde los tutores son más bien intermediarios al proporcionar ayuda al estudiante, que es quien

realmente debería controlar su proceso de aprendizaje, descubriendo elementos de conocimiento por sí mismo más que a través de la instrucción. El tutor, en estos entornos, proporciona pistas, instrucciones, explicaciones, etc., al estudiante, que le ayuden y le encaminen en su aprendizaje.

- **H3**. *El ME propuesto estará destinado a EVIEs monousuario o multiusuario (como MAE-VIF)*. En este último caso, podrá haber varios estudiantes a la vez participando en una cierta actividad de aprendizaje, en la que cada usuario/alumno tenga su propio plan y su seguimiento también sea individual. Pueden existir, además, planes de varios estudiantes interdependientes (por ejemplo, que el plan de un estudiante requiera en un momento determinado que otro estudiante haya dejado un objeto específico en una cierta posición).

- **H4**. A pesar de que en los MEs propuestos hasta la actualidad no se representa explícitamente, y en detalle, una amplia taxonomía acerca de los conocimientos del estudiante, que dé mayor soporte a su método de diagnóstico, es posible desarrollar un ME con un formalismo de representación, como las ontologías, que permita abarcar e interrelacionar todos los conocimientos especificados en los requisitos ya mencionados. La información de este modelo puede ser compartida con otros sistemas y, a la vez, ser reutilizada en un futuro, todo ello con el fin de extender este modelo y adaptarlo para poder ser aplicado a diferentes entornos de aprendizaje.

- **H5**. El método de diagnóstico pedagógico-cognitivo no monótono propuesto facilitará el aprendizaje del estudiante, proporcionando amplia información sobre el estado de los conocimientos de aquél al tutor.

- **H6**. Es posible desarrollar un ME fácilmente adaptable a diferentes SITs.

Las hipótesis H1, H2, y H3, están relacionadas con el objetivo O5. H4 está relacionada con O1, O2, y O5. H5 está relacionada con O3 y O4. H6 está relacionada con el objetivo O5.

A continuación, se presentan las premisas del trabajo:

- **P1**. El diseño de cualquier curso a impartir mediante el SIT deberá partir de un enfoque formativo, lo que supondrá proporcionar la descomposición jerárquica del curso en diferentes unidades de aprendizaje (en fases, y cada fase, a su vez, en actividades), así como los objetivos de aprendizaje asociados a cada unidad. El curso incluirá actividades que el estudiante debe llevar a cabo en sesiones de aprendizaje, bajo la supervisión del tutor.

- **P2**. Aunque existen otras alternativas de modelado, se partirá del hecho de que la descomposición jerárquica del curso se supone se encontrará registrada en una ontología de Tutoría, junto con el plan inicial asociado a cada actividad. Esta ontología está fuera del ámbito del trabajo presente; pertenece al Módulo de Tutoría con la que estará interrelacionada la ontología del ME.

- **P3**. Los objetivos de aprendizaje se registran en la ontología del ME, junto con el plan solución que el estudiante va intentando construir para concluir con éxito una actividad de aprendizaje (un estudiante puede resolver una actividad con un plan distinto al plan del tutor; por lo tanto, el plan seguido por el estudiante para resolver una actividad no tiene porqué coincidir con el plan inicial del tutor).

- **P4**. La ontología del ME deberá ser iniciada con información necesaria antes de que un estudiante comience un curso de aprendizaje. Entre esta información cabe destacar la siguiente:

  - Información sobre el perfil del estudiante.
  - Información referente al curso: operadores asociados a las actividades del curso.
  - Información sobre los objetivos asociados a las actividades del curso, los objetos de conocimiento que deben ser adquiridos para alcanzar esos objetivos, así como la interrelación entre ambos.
  - Información sobre el estado inicial asumido para los objetivos asociados a cada actividad del curso (adquirido, no adquirido, o desconocido).

  El estado de los objetivos durante la ejecución de una actividad habrá sido modificado acorde con el comportamiento del estudiante en cada momento, arrastrando toda la información previamente citada de una sesión a la siguiente. Al inicio de cada actividad se registra en la ontología del ME el plan inicial de esta actividad. La información del plan permitirá ir comparando la ejecución de las acciones del estudiante con la siguiente acción del plan. Asimismo, cada vez que el estudiante realice una acción se registrará en términos de la ontología. Si la acción implica un desplazamiento del estudiante, se actualizan sus coordenadas en la ontología. Esta hipótesis es satisfecha por el diseño de MAEVIF (2.4.8).

- **P5**. El diseño de estrategias de tutoría se considera fuera del ámbito de esta tesis, y son responsabilidad del Módulo de Tutoría.

La premisa P2 está relacionada con el objetivo O1 y O2, la premisa P3 está relacionada con los objetivos O1 y O2, la premisa P4 está relacionada con los objetivos O1, O2, y O3, y las premisas P1 y P5 son generales.

Finalmente, se definen el conjunto de restricciones del trabajo que podrán determinar algunos de los objetivos de investigaciones futuras:

- **R1** La descomposición jerárquica de un curso de otra forma distinta a la planteada en la premisa P1 no se considera en el trabajo actual.

- **R2** El ME propuesto no se aplica a entornos multiusuario en los que se se considere el desempeño de acciones colaborativas entre un grupo de estudiantes.

- **R3**. Los objetivos de aprendizaje asociados a las actividades de un curso se consideran definidas en la ontología de ME de acuerdo siempre a tres taxonomías concretas ya existentes para cada dominio: para el dominio cognitivo (taxonomía de Bloom), para el dominio afectivo (taxonomía de Krathwohl), y para el dominio psicomotor (taxonomía de Simpson) (véase sección ). Sin embargo, existen numerosas aportaciones en este aspecto que son evoluciones de las anteriores, u otras propuestas completamente distintas. En este aspecto, es importante hacer hincapié en que no se defiende a ultranza estas taxonomías, sino que se pretende demostrar que partiendo de la definición de los objetivos y la restante información de la Ontología del Estudiante, se puede obtener un diagnóstico pedagógico que enriquezca la información aportada por el Módulo del Estudiante al Módulo de Tutoría.

El uso de las ontologías permitirá en un futuro añadir repositorios de taxonomías de objetivos o crear una jerarquía de taxonomías de objetivos a incluir en el modelado propuesto, de tal modo que se puedan definir objetivos utilizando alguna de las taxonomías más representativas en este campo, no necesariamente las taxonomías específicas asumidas en este trabajo.

La restricción R1 es general, la restricción R2 se refiere al objetivo O5, y la restricción R3 se refiere a los objetivos O1 y O2.

CAPÍTULO 5

# SOLUCIÓN ADOPTADA

PARA dar respuesta al problema del modelado del estudiante expuesto anteriormente, se presenta en este capítulo la solución propuesta. En primer lugar, se muestra la arquitectura para modelar al estudiante en un SIT con los requisitos establecidos en el capítulo 4. Posteriormente, se describe el desarrollo de la ontología de modelado del estudiante, que engloba la aplicación de la metodología elegida para su construcción y el propio desarrollo de la ontología. Finalmente, tomando como soporte la ontología desarrollada, se detalla el funcionamiento del método de diagnóstico no monótono inmerso en el modelo del estudiante así como una descripción en profundidad de sus módulos.

## 5.1 Arquitectura y funcionamiento básico

El desarrollo de la solución propuesta para modelar al estudiante en un SIT se engloba en el enfoque de diseño pedagógico que se muestra esquemáticamente en la figura 5.1. La aplicación de este enfoque a la enseñanza de una cierta materia se ha dividido en los siguientes pasos:

- Diseño del plan de estudios para la materia a enseñar (X). Supone, en general, definir un conjunto de actividades para facilitar el proceso de aprendizaje incremental de la materia y, además, para cada actividad, definir los objetivos que debe alcanzar el estudiante.

- El Módulo del Experto, a través del planificador, define los pasos (acciones) que se pueden realizar para finalizar con éxito la actividad. Cada una de estas acciones consistirá en la aplicación de un cierto operador, con sus precondiciones y postcondiciones. A diferencia del plan de estudios, de carácter estático, el planificador es dinámico porque al inicio de la sesión de aprendizaje, construye el plan o secuencia de acciones que deberá realizar el estudiante para llevar a cabo con éxito la actividad propuesta. Asimismo, si el estudiante en algún momento se aparta de este plan calculado inicialmente, el planificador puede obtener, si existe, un plan alternativo que todavía le permita completar con éxito la actividad.

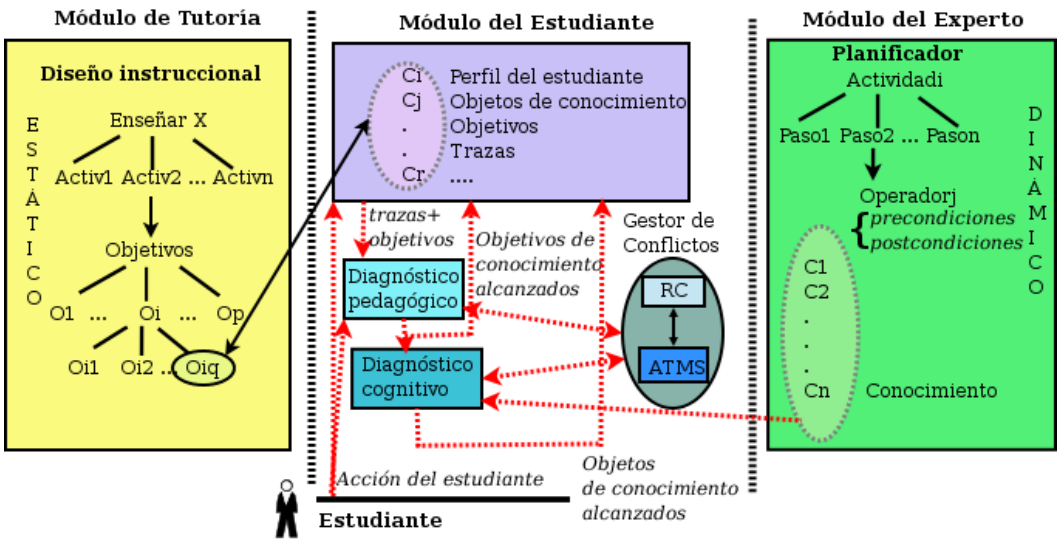


**Figura 5.1:** Arquitectura propuesta para el módulo del estudiante

- Cuando el estudiante ejecuta una cierta acción, esta ejecución se registra en la ontología del ME, que no sólo contiene diversos tipos de conocimientos sino también la interrelación entre ellos (tales como los que relacionan los objetivos de aprendizaje, definidos por el módulo de tutoría, y los objetos de conocimiento, proporcionados por el módulo experto, que el estudiante llegaría a alcanzar si lograse cumplir dichos objetivos).

- El diagnóstico en el ME se divide en dos módulos: el *Diagnóstico Pedagógico* (DP) y el *Diagnóstico Cognitivo* (DC). El DP se encargará de determinar los nuevos objetivos alcanzados o no por el estudiante teniendo en cuenta la última acción ejecutada por el estudiante (registrada en la ontología) y en los objetivos ya alcanzados o no por el estudiante cuando la acción se ejecutó. Para este propósito, el DP usa un conjunto de reglas de diagnóstico. Por otro lado, el DC infiere el estado de conocimiento concreto del estudiante basado en los objetivos alcanzados y en los objetos de conocimiento asociados a ellos.

  Durante el proceso de diagnóstico pueden surgir diversos tipos de contradicciones que el *Gestor de Conflictos* deberá resolver. Para realizar esta funcionalidad, se basará en un sistema ATMS y en un *Módulo de Resolución de Conflictos* (RC).

### 5.1.1 Integración de la arquitectura propuesta en MAEVIF

La arquitectura propuesta para el modelado del estudiante (Figura 5.1), ha sido implementada e integrada en la plataforma software para el desarrollo de EVIEs llamada MAEVIF, cuya arquitectura se describió en la sección 2.4.8.1. Como consecuencia de esta integración, ha surgido la necesidad de establecer un conjunto de requisitos en los que se han tenido en cuenta las características de un EVIE como MAEVIF, y que se detallaron en la sección 7.

En un EVIE, las nuevas capacidades de interacción con el SIT, relativas a una interfaz más inmersiva, deben ser consideradas como nuevas fuentes de información, de gran relevancia en todo el proceso educativo (en la evaluación de la ejecución del estudiante, en una tutoría personalizada adecuada a este tipo de entornos, etc.). El tutor software puede considerar esta nueva

información: los objetos que manipula el estudiante en los entornos 3D, la trayectoria que sigue el estudiante para alcanzar un cierto punto en el escenario virtual, etc. En definitiva, esto implica que la estructura clásica de un SIT, vista en la sección 1.1.1, sea insuficiente para tratar estas nuevas posibilidades características de los EVIEs de tipo 3D.

Para abordar el problema anterior, en el desarrollo de MAEVIF se ha introducido un nuevos módulo independiente, el *Módulo Mundo*, capaz de gestionar el conocimiento acerca del mundo virtual. En su diseño e implementación, uno de los objetivos ha sido que los diversos tipos de conocimiento y procesos que los manejan debían aparecer claramente difererenciados en el sistema, para mantener así la esencia de los SITs; y, a la vez, que todos sus módulos colaboraran con el fin de alcanzar el objetivo común de enseñar al estudiante. Además, esta tarea es facilitada por el uso de una aproximación basada en agentes software.

A continuación, se identifican los módulos en la arquitectura de MAEVIF reflejados, en su mayoría, en la arquitectura propuesta para el modelado del estudiante propuesto (Figura 5.1); y, en cursiva, los agentes de MAEVIF que forman parte de cada módulo. Su descripción e interrelación entre ellos puede verse en la sección (2.4.8.1):

- **Módulo del Estudiante**: Contiene información sobre el estudiante. De esta tarea se encarga el *Agente de Modelado del Estudiante*, en el que está integrado el modelado del estudiante propuesto en este trabajo, y que consta de dos elementos fundamentales: la ontología de Modelado del Estudiante, que registra la información sobre los conocimientos del estudiante; y el método de diagnóstico pedagógico-cognitivo, que permite inferir el estado del conocimiento del estudiante a partir de su comportamiento. Toda esta información será utilizada por el Módulo de Tutoría para guiar y ayudar al estudiante durante su aprendizaje de forma más adaptable, acorde a las características individuales del alumno.

- **Módulo de Tutoría**: A este módulo pertenece el *Agente de Tutoría*, cuyas principales misiones son: proponer actividades a los estudiantes según el conocimiento que posean de la materia, y realizar el seguimiento del estudiante.

- **Módulo Experto**: Debe representar el conocimiento relevante en el dominio o materia que se enseña. Para esta tarea el *Agente Experto* recoge la información relacionada con las acciones que los estudiantes pueden realizar en el EV. Además, debe resolver problemas como lo haría un experto. Esta tarea es llevada a cabo por el *Agente de Planificación*, que permite la construcción dinámica de planes-solución, y el *Agente de Simulación* que, principalmente, ayuda al *Agente Experto* en el proceso de validación de las acciones del estudiante, al tiempo que simula las consecuencias de estas acciones.

- **Módulo de Comunicación**: Permite la comunicación entre los estudiantes y el sistema. De esta tarea se encargan el *Agente de Comunicación Global* y el *Agente de Comunicación del Estudiante*.

- **Módulo del Mundo**: Es un nuevo módulo introducido en MAEVIF como consecuencia de las nuevas capacidades de interacción que suponen los EVs 3D. Este módulo, no existente en la estructura clásica de los SITs, se encarga de gestionar el conocimiento acerca del mundo virtual. Es responsable fundamentalmente de: a) mantener el estado del escenario virtual, los objetos posicionados en él, y los personajes virtuales que están inmersos en él; b) proporcionar las capacidades perceptivas o sintéticas para los personajes virtuales introducidos en el EV para soportar el proceso de aprendizaje, y; c) Representar virtualmen-

te al tutor (tutor virtual) y otros miembros del equipo. De estas tareas son responsables el *Agente Mundo*, el *Agente Percepción*, y el *Agente Tutor Virtual*, respectivamente.

Por todo lo anterior, se puede decir que la arquitectura de MAEVIF es una extensión de la arquitectura de un SIT clásico para EVIEs.

## 5.2 Ontología de Modelado del Estudiante

La solución adoptada para representar el conocimiento del ME está soportada por el formalismo de las ontologías justificado ya en el apartado 4. Una vez elegido el formalismo de representación, se seleccionó como lenguaje de definición de ontologías OWL[1], que es una recomendación de la W3C ampliamente utilizada. Además, este lenguaje es lo suficientemente expresivo para el propósito de esta tesis, y existe abundante documentación así como herramientas para trabajar con ontologías expresadas en OWL.

Elegido el lenguaje de ontologías, se escogió Protégé como herramienta de construcción de ontologías (3.3.4). La expresividad de Protégé permite crear una ontología en la que prácticamente todos los componentes de modelado se representan en ella o se pueden representar. Uno de los componentes que no admite representación son las reglas. No obstante, la capacidad de razonamiento con la ontología del ME presentada en este trabajo se consigue con un motor de inferencia complementario. Otros de los motivos fundamentales para esta elección es que la herramienta Protégé es *open source* y tiene la posibilidad de exportar una ontología a formato OWL. Otras utilidades que incorpora como *plug-ins* han sido también usados en la tesis, por ejemplo, el editor gráfico de ontologías OWLViz[2].

La Ontología de Modelado del Estudiante representa los diversos tipos de conocimientos del estudiante que se consideran necesarios para modelar adecuadamente al estudiante en un SIT y que permitirán un aprendizaje adaptativo, que se adecúe a las características individuales de cada alumno. Esta ontología ha sido diseñada con vistas a ser utilizada en la enseñanza de procedimientos, por ejemplo, en el aprendizaje de actividades referentes a una interfaz gráfica de usuario o en Entornos Virtuales de Aprendizaje para Entrenamiento de tipo procedimental. En ambos tipos de Entornos de Entrenamiento/Formación existe un plan de actuación o secuencia de acciones que el estudiante debe realizar para finalizar una tarea con éxito.

En primer lugar, se describirá la metodología para construir la Ontología de Modelado del Estudiante. En segundo lugar y de forma genérica, se muestran las ontologías modulares en las que se ha descompuesto la Ontología general de Modelado del Estudiante. Posteriormente, se presenta la modificación que debe realizarse para adaptarla a entornos específicos, en concreto, para dos tipos de entornos de aprendizaje: aprendizaje de un GUI y aprendizaje en Entornos Virtuales de Entrenamiento.

### 5.2.1 Metodología de desarrollo de la Ontología general de Modelado del Estudiante

Tras el análisis del estado del arte referente a las metodologías existentes para la construcción de ontologías (ver apartado 3.3.6.1), se ha seleccionado la metodología NeOn para el desarrollo de la ontología del estudiante presentada en este trabajo. La metodología NeOn no sólo se considera que proporciona unas líneas metodológicas más completas para la construcción

[1] http://www.w3.org/2007/OWL/wiki/OWL_Working_Group
[2] http://protegewiki.stanford.edu/index.php/OWLViz

de la ontología sino que, además, el estudio preliminar de los conocimientos a incluir en la ontología del estudiante así como su interrelación, ha conducido a elegir esta metodología como la más adecuada y la única para construir la red de ontologías de Modelado del Estudiante, seleccionada como la mejor alternativa para el desarrollo de la ontología. Es más, los escenarios y las guías metodológicas de NeOn, se han considerado muy adecuadas para la evolución de la propia ontología en el futuro.

El escenario de construcción de la Ontología de Modelado del Estudiante, acorde a la metodología NeOn, no es único sino una combinación de varios de los especificados (véase el apartado 3.3.6.1). Durante su desarrollo, se ha realizado la construcción de la ontología general desde la fase de especificación hasta su implementación (escenario 1 de la metodología NeOn). Sin embargo, para la adaptación de la Ontología general a dos entornos concretos de aprendizaje: entornos GUIs y Entornos Virtuales de Entrenamiento procedimental, como se describirá más adelante en el capítulo 6, ha sido necesario considerar el escenario 2 de construcción de ontologías en la metodología NeOn. Asimismo, la propia evolución de la ontología ha motivado la conveniencia de realizar una modularización de la ontología, no realizada a priori, su extensión y su especialización. Estas actividades forman parte del Escenario 8 en la metodología NeOn.

Los dos apartados siguientes se centran fundamentalmente en el Escenario 1 para la construcción de la ontología ya que incluye las actividades principales que tienen que ser realizadas en cualquier desarrollo de ontologías, sea cual sea la combinación de escenarios para su construcción. Se va a dedicar especial atención a los puntos más importantes, no exhaustivamente, de la tarea de especificación de la Ontología general de Modelado del Estudiante siguiendo las líneas propuestas en la metodología NeOn.

### 5.2.2 Especificación de las necesidades de la Ontología de Modelado del Estudiante

La actividad de especificación establece porqué la ontología es construida, qué uso se espera de ella y quiénes son los usuarios finales. Para especificar los requerimientos de la ontología se usará la técnica de cuestiones de competencia propuestas en Grüninger and Fox (1995).

#### 5.2.2.1 Uso esperado de la Ontología de Modelado del Estudiante

El propósito de la construcción de la Ontología de Modelado del Estudiante es proporcionar amplia y valiosa información consensuada sobre el estado de conocimiento del estudiante, trazas del comportamiento del estudiante a lo largo de las diferentes sesiones de aprendizaje, etc., de tal forma que el módulo tutor pueda tomar decisiones de tutoría razonables así como proporcionar a los estudiantes una adecuada realimentación en cada momento de su aprendizaje. Además, la solución propuesta se beneficiará de las ventajas del formalismo de las ontologías, fundamentalmente facilidad de reutilización y extensión a diferentes SITs. Esto significa que, aunque la ontología del estudiante se va a implementar como parte de MAEVIF, plataforma para el desarrollo de EVIEs y, específicamente, dentro de su agente denominado Agente de Modelado del Estudiante, podrá ser fácilmente reutilizada para otros SITs (no sólo para Entornos Virtuales Inteligentes para la Formación/Entrenamiento de tipo procedimental) gracias a la extensibilidad y grado de generalidad con que se pretende dotar a esta ontología. En concreto, el propósito es que la ontología pueda ser usada también para SITs en entornos 2D para aprendizaje/entrenamiento del manejo de interfaces gráficas de usuario (GUIs) en aplicaciones de ordenador (por ejemplo, manejo de un editor de textos).

Otro propósito fundamental es permitir un fácil diagnóstico basado en reglas a través de esta ontología teniendo en cuenta además como fuente de información las posibilidades de interacción derivadas del uso de los EVs en EVITs.

#### 5.2.2.2 Usuarios finales previstos de la Ontología de Modelado del Estudiante

Como se ha mencionado previamente, la Ontología de Modelado del Estudiante formará parte de la plataforma para el desarrollo de EVIEs denominada MAEVIF y se espera su uso en entornos para manejo de GUIs en ciertas aplicaciones de ordenador. Por lo tanto, los usuarios esperados de esta ontología son los siguientes:

- Estudiantes que usen SITs en general para el aprendizaje de alguna materia. En especial, el uso de la ontología está orientado a los siguientes grupos de estudiantes:

    1. Estudiantes que utilizan los EVIEs como MAEVIF para la formación/entrenamiento en materias en las que estos entornos son especialmente apropiados, por ejemplo, para formación/entrenamiento en contextos de alto riesgo como centrales nucleares, laboratorios de química donde la peligrosidad se derive de la manipulación de sustancias peligrosas y/o nocivas para la saludo como los ácidos, pesticidas, etc., así como para formación/entrenamiento en escenarios de difícil reproducción como entrenamiento de astronautas, etc. En particular, los estudiantes recibirán en estos entornos una formación de tipo procedimental.
    2. Estudiantes que usen entornos 2D para el aprendizaje del manejo de GUIs en aplicaciones de computador como, por ejemplo, un editor de texto.

#### 5.2.2.3 Cuestiones de competencia

Las cuestiones de competencias son preguntas en lenguaje natural usadas para determinar el ámbito de la ontología que se va a desarrollar y se utilizan para extraer los principales conceptos y sus propiedades, relaciones y axiomas formales de la ontología. Por lo tanto, estas cuestiones y sus respuestas pueden considerarse como un tipo especificación de requerimientos base para que la ontología pueda ser evaluada.

Las cuestiones de competencia pueden componerse (*cuestiones simples*) formando *cuestiones compuestas*. Estas últimas se responden mediante la composición de las respuestas asociadas a las cuestiones de competencia simples.

En la generación de las cuestiones de competencia para la construcción de la Ontología de Modelado del Estudiante se han agrupado éstas en cinco grandes bloques. Estos bloques son considerados claves dentro de la taxonomía de conocimientos del estudiante: perfil del estudiante, objetos de conocimiento, objetivos de conocimiento, estado del estudiante y traza del estudiante.

Por su extensión, se incluye a continuación cinco tablas. Cada una de ellas contiene un fragmento importante, pero no exhaustivo, de los cinco bloques de cuestiones de competencia especificadas para la construcción de la ontología de Modelado del Estudiante. Es importante hacer hincapié en que las cuestiones de competencia no se han obtenida en una única fase. En primer lugar, se obtuvieron las cuestiones de competencia para la ontología de Modelado del Estudiante general. Más tarde, para cada uno de los prototipos de demostración de aplicación del método propuesto en este trabajo -aprendizaje de un GUI (editor de texto), y aprendizaje en un Entorno

Virtual-, tal y como se detallará en el capítulo 6, ha sido necesario realizar una extensión y especialización de la ontología general desarrollada para cada uno de estos entornos de aprendizaje (escenario 8 de construcción de ontologías de la metodología NeOn).

Tabla 5.1: Cuestiones de competencias específicas relacionadas con el *perfil del estudiante*

| IDENTIFICADOR | CUESTIÓN DE COMPETENCIA |
|---|---|
| **CC1.** | *¿Cuál es su nombre?* |
| **CC2.** | *¿Cuál es su identificador?* |
| **CC3.** | *¿Cuáles son sus apellidos?* |
| **CC4.** | *¿Cuál es su fecha de nacimiento?* |
| **CC5.** | *¿Cuál es su dirección?* |
| **CC6.** | *¿Cuál es su estado civil?* |
| **CC7.** | *¿Cuál es su sexo?* |
| **CC8.** | *¿Cuál es su ciudad?* |
| **CC9.** | *¿Cuál es su provincia?* |
| **CC10.** | *¿Cuál es su país?* |
| **CC11.** | *¿Cuál es su código postal?* |
| **CC12.** | *¿Cuál es su teléfono?* |
| **CC13.** | *¿Cuál es su dirección de correo electrónico?* |
| **CC14.** | *¿Cuáles son sus dimensiones corporales?* |
| **CC15.** | *¿Cuáles son sus discapacidades?* |
| **CC16.** | *¿Cuáles son sus estilos de aprendizaje preferidos?* |
| **CC17.** | *¿Cuáles son sus preferencias en dispositivos de entrada y salida?* |
| **CC18.** | *¿Cuáles son sus áreas de experiencia?* |
| **CC19.** | *¿Cuál es su nivel de experiencia en un cierto área?* |
| **CC20.** | *¿Cuáles son sus áreas de experiencia con computadores?* |

Tabla 5.1: Cuestiones de competencias específicas relacionadas con el *perfil del estudiante* (cont.)

| IDENTIFICADOR | CUESTIÓN DE COMPETENCIA |
|---|---|
| **CC21.** | *¿Cuál es su experiencia concreta en un cierto área de computadores?* |
| **CC22.** | *¿Cuál es su nivel de experiencia previa en un cierto área relacionado con los computadores?* |
| **CC23.** | *¿Cuáles son sus perspectivas en un cierto área relacionado con los computadores?* |
| **CC24.** | *¿Cuál ha sido su ocupación en un cierto área relacionado con los computadores?* |
| **CC25.** | *¿Tiene un especial rechazo o manía a los contenidos que va a aprender?* |
| **CC26.** | *¿Cuál es su interés a priori sobre los contenidos a aprender?* |
| **CC27.** | *¿Cuál es su interés actual por los contenidos que aprende?* |
| **CC28.** | *¿Cómo se define su personalidad?* |
| **CC29.** | *¿Cuál es su estado psicológico en la situación actual?* |
| **CC30.** | *¿En qué grado se encuentra en este instante en un cierto estado psicológico?* |

Tabla 5.2: Cuestiones de competencias específicas relacionadas con un *objetivo de aprendizaje*

| IDENTIFICADOR | CUESTIÓN DE COMPETENCIA |
|---|---|
| **CC31.** | *¿Cuál es su identificador?* |
| **CC32.** | *¿Cuál es su descripción?* |
| **CC33.** | *¿Cuáles son las actividades que requieren este objetivo como prerrequisito para ser iniciadas?* |
| **CC34.** | *¿Cuáles son las actividades que deben alcanzar este objetivo tras su finalización?* |
| **CC35.** | *¿Cuáles son los objetos de conocimiento requeridos por él?* |

Tabla 5.2: Cuestiones de competencias específicas relacionadas con un *objetivo de aprendizaje* (cont.)

| IDENTIFICADOR | CUESTIÓN DE COMPETENCIA |
|---|---|
| **CC36.** | *¿De qué objetivos de aprendizaje consta un objetivo didáctico?* |
| **CC37.** | *¿Qué fases en un determinado curso tienen definido como objetivo a ser alcanzado un cierto objetivo didáctico?* |
| **CC38.** | *¿Qué fases en un determinado curso requieren un cierto objetivo didáctico como prerrequisito para ser iniciadas?* |
| **CC39.** | *¿Cuál es el nivel de fialibilidad de un cierto objetivo específico a partir del cuál se considerará éste alcanzado?* |

Tabla 5.3: Cuestiones de competencias específicas relacionadas con un *objeto de aprendizaje*

| IDENTIFICADOR | CUESTIÓN DE COMPETENCIA |
|---|---|
| **CC40.** | *¿Cuál es su identificador?* |
| **CC41.** | *¿Cuál es su descriptor?* |
| **CC42.** | *¿A qué objeto(s) de conocimiento(s) pertenece una cierta propiedad?* |
| **CC43.** | *¿Es una propiedad funcional?* |
| **CC44.** | *¿Es una propiedad funcional inversa?* |
| **CC45.** | *¿Es una propiedad simétrica?* |
| **CC46.** | *¿Cuál es el valor de una propiedad?* |
| **CC47.** | *¿Cuál es el valor por defecto de una propiedad?* |
| **CC48.** | *¿Cuál es el tipo de valor de una propiedad?* |
| **CC49.** | *¿Cuál es la posición de un cierto objeto?* |
| **CC50.** | *¿De qué escenarios consta un cierto mapa virtual?* |
| **CC51.** | *¿Cuál es la posición origen a partir de la que un estudiante sabe construir su ubicación en el mapa mental?* |

Tabla 5.3: Cuestiones de competencias específicas relacionadas con un *objeto de aprendizaje* (cont.)

| IDENTIFICADOR | CUESTIÓN DE COMPETENCIA |
|---|---|
| **CC52.** | *¿Cuál es la posición destino hasta la que un estudiante sabe construir su ubicación en el mapa mental?* |
| **CC53.** | *¿Cuáles son sus dimensiones corporales?* |
| **CC54.** | *¿A qué subescenario pertenece un cierto objeto geométrico?* |
| **CC55.** | *¿Es manipulable o no un cierto objeto geométrico?* |
| **CC56.** | *¿De qué objetos simples geométricos se compone un cierto objeto compuesto geométrico?* |
| **CC57.** | *¿Con qué otros escenarios está conectado un cierto escenario virtual?* |
| **CC58.** | *¿De qué subescenarios consta un cierto escenario virtual?* |
| **CC59.** | *¿Qué objetos geométricos contiene un cierto subescenario virtual?* |
| **CC60.** | *¿Con qué otros subescenarios está conectado un cierto subescenario virtual?* |
| **CC61.** | *¿Cuál es la coordenada x de la posición absoluta de un objeto?* |
| **CC62.** | *¿Cuál es la coordenada y de la posición absoluta de un objeto?* |
| **CC63.** | *¿Cuál es la coordenada z de la posición absoluta de un objeto?* |
| **CC64.** | *¿Cuál es el área (subescenario) asociado a la posición de un objeto (posición relativa a área)?* |
| **CC65.** | *¿Cuál es el objeto geométrico asociado a la posición de otro objeto (posición relativa a objeto)?* |
| **CC66.** | *¿Qué concepto está representado con una cierta variable?* |
| **CC67.** | *¿Cuál es el valor de una cierta variable?* |
| **CC68.** | *¿Cuál es el nombre que identifica una cierta relación?* |
| **CC69.** | *¿Cuál es la cardinalidad de una relación?* |

Tabla 5.3: Cuestiones de competencias específicas relacionadas con un *objeto de aprendizaje* (cont.)

| IDENTIFICADOR | CUESTIÓN DE COMPETENCIA |
|---|---|
| **CC70.** | *¿Cuál es el rango de una relación?* |
| **CC71.** | *¿Cuál es el dominio de una relación?* |
| **CC72.** | *¿Es una cierta relación reflexiva?* |
| **CC73.** | *¿Es una cierta relación simétrica?* |
| **CC74.** | *¿Es una cierta relación transitiva?* |
| **CC75.** | *¿Cuál es el nombre del operador asociado a una acción puntual?* |
| **CC76.** | *¿Cuáles son los roles que se pueden desempeñar al ejecutar una acción puntual?* |
| **CC77.** | *¿Cuáles son las precondiciones asociadas a una acción puntual?* |
| **CC77.** | *¿Cuáles son las consecuencias (postcondiciones) asociadas a una acción puntual?* |
| **CC78.** | *¿Cuáles son las metas alcanzadas al ejecutar una acción puntual?* |
| **CC79.** | *¿Cuál es el nombre de la función que permite obtener el modo de ejecución de una acción puntual?* |
| **CC80.** | *¿Cuál es el nombre de la función que permite obtener la calidad de la ejecución de una acción puntual?* |
| **CC80.** | *¿Qué objetos se usan como herramientas en una cierta acción puntual?* |
| **CC81.** | *¿Qué acción puntual está asociada a una cierta aplicación de acción?* |
| **CC82.** | *¿Cuándo se inicia una cierta aplicación de acción?* |
| **CC83.** | *¿Cuándo finaliza una cierta aplicación de acción?* |
| **CC84.** | *¿De qué elementos de un plan se compone una acción compuesta?* |
| **CC85.** | *¿Cuál es la posición relativa en el plan de uno de sus elementos?* |

Tabla 5.3: Cuestiones de competencias específicas relacionadas con un *objeto de aprendizaje* (cont.)

| IDENTIFICADOR | CUESTIÓN DE COMPETENCIA |
|---|---|
| **CC86.** | *¿De qué secuencia de aplicaciones de acción se compone un cierto procedimiento?* |
| **CC87.** | *¿De qué condición se compone un elemento de una determinada acción puntual?* |
| **CC88.** | *¿Sobre qué entidad se establece una condición sobre el estado de una determinada acción puntual?* |
| **CC89.** | *¿De qué elementos de plan, aplicaciones de acción y/o acciones compuestas, se compone un plan?* |
| **CC90.** | *¿Cuál es el coste de la solución óptima asociada al plan?* |
| **CC91.** | *¿Cuál es el nombre asociado a una cierta regla?* |
| **CC92.** | *¿Cuál es la posición de origen de un tramo asociado a una trayectoria?* |
| **CC93.** | *¿Cuál es la posición final de un tramo asociado a una trayectoria?* |
| **CC94.** | *¿De qué tramos se compone una trayectoria?* |
| **CC95.** | *¿Cuál es la posición origen de una trayectoria?* |
| **CC96.** | *¿Cuál es la posición final de una trayectoria?* |
| **CC97.** | *¿Cuál es la distancia a recorrer en una trayectoria?* |
| **CC98.** | *¿Cuál es la velocidad asumida en una trayectoria?* |
| **CC99.** | *¿Cuál es el posición de origen de un camino?* |
| **CC100.** | *¿Cuál es el posición final de un camino?* |

Tabla 5.4: Cuestiones de competencias específicas relacionadas con la *traza de un estudiante*

| IDENTIFICADOR | CUESTIÓN DE COMPETENCIA |
|---|---|
| **CC101.** | *¿A qué estudiante pertenece una cierta traza?* |

Tabla 5.4: Cuestiones de competencias específicas relacionadas con la *traza de un estudiante* (cont.)

| IDENTIFICADOR | CUESTIÓN DE COMPETENCIA |
|---|---|
| **CC102.** | *¿Cuándo comienza una cierta traza?* |
| **CC103.** | *¿Cuándo finaliza una cierta traza?* |
| **CC104.** | *¿Cuál es la acción específica asociada a una cierta traza de acción?* |
| **CC105.** | *¿Cuál es el estado de ejecución de la acción correspondiente a una cierta traza de acción?* |
| **CC106.** | *¿Cuál es el identificador de la actividad asociada a una cierta traza de actividad?* |
| **CC107.** | *¿A qué traza de sesión pertenece una cierta traza de actividad?* |
| **CC108.** | *¿De qué trazas de procedimientos consta una cierta traza de actividad?* |
| **CC109.** | *¿Cuál es el estado de ejecución de actividad asociado a una cierta traza de actividad?* |
| **CC110.** | *¿Cuál es el estado emocional asociado a una cierta traza emocional de un estudiante?* |
| **CC111.** | *¿De qué trazas de variables consta una cierta traza emocional de un estudiante?* |
| **CC112.** | *¿De qué trazas de objetivos consta el registro de trazas de objetivos de un estudiante?* |
| **CC113.** | *¿Cuál es el estado de objetivo asociado a una cierta traza de objetivo de un estudiante?* |
| **CC114.** | *¿Cuál es el estado del objetivo asociado a registrar en una cierta traza de objetivo?* |
| **CC115.** | *¿A qué traza de ejecución de acción corresponde una cierta traza de objetivo?* |
| **CC16.** | *¿De qué trazas de ejecución de acciones consta una cierta traza de procedimiento?* |

Tabla 5.4: Cuestiones de competencias específicas relacionadas con la *traza de un estudiante* (cont.)

| IDENTIFICADOR | CUESTIÓN DE COMPETENCIA |
|---|---|
| **CC117.** | *¿Cuál es el estado de sesión asociado a una cierta traza de sesión?* |
| **CC118.** | *¿De qué trazas de actividad consta una cierta traza de sesión?* |
| **CC119.** | *¿Cuál es el identificador de sesión asociado a una cierta traza de sesión?* |
| **CC120.** | *¿Cuál es la variable trazada en una cierta traza de variable?* |

Tabla 5.5: Cuestiones de competencias específicas relacionadas con el estado del *conocimiento del estudiante*

| IDENTIFICADOR | CUESTIÓN DE COMPETENCIA |
|---|---|
| **CC121.** | *¿A qué estudiante pertenece un cierto estado de conocimiento?* |
| **CC122.** | *¿Cuál es la valoración de la velocidad de aprendizaje de un cierto estudiante tras realizar una actividad?* |
| **CC123.** | *¿Cuál es la valoración del nivel de memoria de un cierto estudiante tras realizar una actividad?* |
| **CC124.** | *¿Cuál es la valoración del nivel de razonamiento de un cierto estudiante tras realizar una actividad?* |
| **CC125.** | *¿Cuál es la traza emocional a partir de la cual se obtiene un cierto estado emocional?* |
| **CC126.** | *¿Qué valoración del estado emocional está asociado a un cierto estado de un estudiante?* |
| **CC127.** | *¿Con qué intensidad se ha obtenido un cierto estado emocional de un estudiante?* |
| **CC128.** | *¿Qué nivel de experiencia ha logrado un estudiante en un cierto estado?* |

Tabla 5.5: Cuestiones de competencias específicas relacionadas con el *estado del conocimiento del estudiante* (cont.)

| IDENTIFICADOR | CUESTIÓN DE COMPETENCIA |
|---|---|
| **CC129.** | *¿A qué objeto de conocimiento corresponde un cierto estado de objeto de conocimiento?* |
| **CC130.** | *¿Cuál es la valoración obtenida de un cierto objeto conocimiento (estructural o procedimental) por el estudiante en un determinado estado?* |
| **CC131.** | *¿Cuál es el tiempo de aplicación de un cierto conocimiento procedimental en un cierto estado de un estudiante?* |
| **CC132.** | *¿Cuántas veces se aplicó bien un cierto conocimiento procedimental en un determinado estado de un estudiante?* |
| **CC133.** | *¿Cuántas veces se aplicó mal un cierto conocimiento procedimental en un determinado estado de un estudiante?* |
| **CC134.** | *¿Cuántas veces un estudiante ha aplicado bien un operador pero la acción asociada no estaba en el plan ?* |
| **CC135.** | *¿Cuántas veces un estudiante no ha aplicado un operador porque no correspondía el objeto al que se puede aplicar el operador?* |
| **CC136.** | *¿Cuántas veces un estudiante no ha aplicado un operador porque no se cumplía alguna de sus precondiciones?* |
| **CC137.** | *¿Cuántas veces un estudiante ha aplicado un operador determinado fuera de secuencia pero estando en el plan de aprendizaje y coincidiendo operador y argumentos?* |
| **CC138.** | *¿Cuántas veces un estudiante ha aplicado un operador determinado fuera de secuencia pero estando en el plan de aprendizaje, coincidiendo operador pero no alguno de sus argumentos?* |
| **CC139.** | *¿Cuántas veces un estudiante ha aplicado un operador coincidiendo operador y argumentos con la siguiente que corresponde ejecutar en el plan asociado a una determinada actividad?* |
| **CC149.** | *¿Cuántas veces un estudiante ha aplicado un operador coincidiendo operador pero no argumentos con la siguiente que corresponde ejecutar en el plan de una determinada actividad?* |

Tabla 5.5: Cuestiones de competencias específicas relacionadas con el *estado del conocimiento del estudiante* (cont.)

| IDENTIFICADOR | CUESTIÓN DE COMPETENCIA |
|---|---|
| **CC141.** | *¿Cuál es la traza de actividad asociada a un cierto estudiante a la que pertenece ciertos estados de acciones?* |
| **CC142.** | *¿De qué conjunto de estados de acción se compone un cierto estado de acciones asociado a la traza de una actividad?* |
| **CC143.** | *¿Cuál es la valoración (factor) del estado de todas las acciones realizadas durante la ejecución de una actividad para un estudiante?* |
| **CC144.** | *¿Cuál es el estado de adquisición de un cierto objetivo para un estudiante en un cierto instante?* |
| **CC145.** | *¿Cuál es el objetivo asociado a un cierto estado de objetivo?* |
| **CC146.** | *¿A qué traza de objetivo corresponde un cierto estado de objetivo?* |
| **CC147.** | *¿Es el estado de un cierto objetivo asumido o, por el contrario, es deducido?* |
| **CC148.** | *¿Cuál es la causa por la que se infirió un cierto estado de objetivo?* |
| **CC149.** | *¿Cuál es el nivel de fiabilidad hasta el momento conseguido por un estudiante para un cierto estado de objetivo?* |
| **CC150.** | *¿Cuál es la nota media obtenida por un estudiante en los diferentes niveles (curso, fase y actividad) de un plan de estudios?* |
| **CC151.** | *¿Se ha completado o no por un estudiante un cierto nivel de aprendizaje de un plan de estudios?* |
| **CC152.** | *¿Cuántas veces se ha completado por un estudiante un cierto nivel de aprendizaje de un plan de estudios?* |
| **CC153.** | *¿Cuál es la media de preguntas avanzadas realizadas por un estudiante durante la ejecución de una actividad?* |
| **CC154.** | *¿Cuál es la media de preguntas generales realizadas por un estudiante durante la ejecución de una actividad?* |

Tabla 5.5: Cuestiones de competencias específicas relacionadas con el *estado del conocimiento del estudiante* (cont.)

| IDENTIFICADOR | CUESTIÓN DE COMPETENCIA |
|---|---|
| **CC155.** | *¿Cuál es la media de preguntas básicas realizadas por un estudiante durante la ejecución de una actividad?* |
| **CC156.** | *¿Cuál es el promedio de éxitos logrados por el estudiante durante la ejecución de una actividad?* |
| **CC157.** | *¿A qué actividad corresponde un cierto estado de actividad?* |
| **CC158.** | *¿A partir de qué estados de ejecución de una actividad se obtiene un cierto estado de esta actividad?* |
| **CC159.** | *¿Cuál es la valoración del conocimiento de una cierta actividad obtenido para un cierto estudiante?* |
| **CC160.** | *¿Cuál es el promedio de replanificaciones realizadas hasta el momento al realizar un estudiante una actividad?* |
| **CC161.** | *¿Qué valoración cualitativa del nivel de experiencia en una cierta actividad tiene un estudiante?* |
| **CC162.** | *¿A qué curso está asociado un cierto estado de curso?* |
| **CC163.** | *¿De qué estados de fases consta un cierto estado de curso para un estudiante?* |
| **CC164.** | *¿A qué fase está asociada un cierto estado de fase de un estudiante?* |
| **CC165.** | *¿De qué estados de actividad consta un cierto estado de fase de un estudiante?* |
| **CC166.** | *¿A qué curriculum de un cierto estudiante corresponde el estado de un cierto plan de estudios?* |
| **CC167.** | *¿De qué estados de cursos está compuesto el estado de un cierto plan de estudios para un estudiante?* |
| **CC168.** | *¿Cuál es el estado de las precondiciones asociadas al estado de ejecución de una acción?* |
| **CC169.** | *¿Cuál es el estado de las postcondiciones asociadas al estado de ejecución de una acción?* |

Tabla 5.5: Cuestiones de competencias específicas relacionadas con el *estado del conocimiento del estudiante* (cont.)

| IDENTIFICADOR | CUESTIÓN DE COMPETENCIA |
|---|---|
| **CC170.** | *¿A qué estado de ejecución de actividad pertenece un cierto estado de ejecución de una acción?* |
| **CC171.** | *¿A partir de qué traza de acción ejecutada por el estudiante se obtiene un cierto estado de ejecución de acción?* |
| **CC172.** | *¿El operador asociado a una acción se aplicó finalmente, es decir, se cumplían las precondiciones asociadas a esa acción cuando se intentó aplicar?* |
| **CC173.** | *¿Cuál es la valoración cualitativa de la ejecución de una cierta actividad para un estudiante?* |
| **CC174.** | *¿A qué traza de actividad pertenece un cierto estado de ejecución de actividad?* |
| **CC175.** | *¿De qué estados de ejecución de acciones consta un cierto estado de ejecución de actividad?* |
| **CC176.** | *¿De qué estados de ejecución de acciones consta un cierto estado de ejecución de actividad?* |
| **CC177.** | *¿Qué coste asociado tiene un cierto estado de ejecución de actividad?* |
| **CC178.** | *¿Qué tiempo se ha invertido en la ejecución de una determinada actividad?* |
| **CC179.** | *¿Cuál es el factor o valoración cuantitativa asociado a la ejecución de una actividad por un estudiante?* |
| **CC180.** | *¿Cuál es el índice de errores en una cierta ejecución de una actividad?* |
| **CC181.** | *¿Cuál es el índice de éxitos en una cierta ejecución de una actividad?* |
| **CC182.** | *¿Cuál es en la ejecución de una actividad la siguiente acción que corresponde ejecutar de acuerdo al plan correspondiente?* |
| **CC183.** | *¿Cuántas preguntas avanzadas ha realizado un estudiante durante la ejecución de una cierta actividad?* |

Tabla 5.5: Cuestiones de competencias específicas relacionadas con el *estado del conocimiento del estudiante* (cont.)

| IDENTIFICADOR | CUESTIÓN DE COMPETENCIA |
|---|---|
| **CC184.** | *¿Cuántas preguntas generales ha realizado un estudiante durante la ejecución de una cierta actividad?* |
| **CC185.** | *¿Cuántas preguntas básicas ha realizado un estudiante durante la ejecución de una cierta actividad?* |
| **CC186.** | *¿Cuántas respuestas incorrectas ha dado el estudiante durante la ejecución de una cierta actividad?* |
| **CC187.** | *¿Cuántos ejemplos ha proporcionado el tutor durante la ejecución de una cierta actividad?* |
| **CC188.** | *¿Cuántos problemas ha proporcionado el tutor durante la ejecución de una cierta actividad?* |
| **CC189.** | *¿Cuántas replanificaciones se han producido durante la ejecución de una cierta actividad?* |
| **CC190.** | *¿Cuántos intentos fallidos se han producido durante la ejecución de una cierta actividad?* |
| **CC191.** | *¿De qué estados de acciones se ha obtenido un cierto estado de ejecución de una actividad?* |
| **CC192.** | *¿A qué elemento de acción puntual (condición -post-condición o precondición) corresponde un cierto estado de una condición?* |
| **CC193.** | *¿Con qué calidad se obtuvo un determinado estado de una meta asociada a una acción?* |
| **CC194.** | *¿El estado de una cierta precondición asociada a la ejecución de una acción es satisfecha o no?* |
| **CC195.** | *¿Cuál es el grado de realización o alcance asociado a un cierto estado de sesión?* |
| **CC196.** | *¿Cuánto tiempo ha durado un cierto estado de sesión?* |
| **CC197.** | *¿A qué traza de sesión está asociado un cierto estado de sesión?* |
| **CC198.** | *¿El estado de una sesión es ya iniciada o no?* |

Las cuestiones de competencia en cada uno de los grupo presentados en las tablas precedentes y entre grupos, se pueden componer formando cuestiones de competencia más generales. Algunas de estas cuestiones establecidas para la construcción de la ontología de Modelado del Estudiante se muestran como ejemplo en la tabla siguiente:

Tabla 5.6: Cuestiones de competencias compuestas

| IDENTIFICADOR | TIPO DE INFORMACIÓN | CUESTIÓN DE COMPETENCIA |
|---|---|---|
| **CC199.** | **Traza del estudiante**<br>**Estado del estudiante** | *Dada una cierta acción ejecutada por el estudiante en una actividad, ¿Cuál es la siguiente acción que corresponde ejecutar en el plan de esa actividad?* |
| **CC200.** | **Traza del estudiante**<br>**Estado del estudiante** | *Dada la traza de una acción ejecutada por el estudiante, ¿Qué precondiciones asociadas a su acción se cumplen* |
| **CC201.** | **Estado del estudiante**<br>**Objetivo de aprendizaje** | *Dado un cierto estado de objetivo (alcanzado o no) para un estudiante, ¿Cuál es el nivel de fiabilidad del objetivo asociado?* |
| **CC202.** | **Perfil del estudiante**<br>**Traza del estudiante** | *Dada el identificador de un estudiante, ¿Cuáles son las acciones ejecutadas por él durante la sesión actual?* |
| **CC203.** | **Objetivo de aprendizaje**<br>**Estado de estudiante**<br>**Perfil del estudiante** | *Dada un objetivo de aprendizaje, ¿Cuáles es la información personal de los estudiantes que han alcanzado dicho objetivo?* |
| **CC204.** | **Perfil de estudiante**<br>**Objetivo de aprendizaje**<br>**Estado de estudiante**<br>**Objeto de aprendizaje** | *¿Un determinado estudiante ha alcanzado un objetivo que requiere unos objetos de conocimiento concretos?* |

A partir de las cuestiones de competencias establecidas, de las que se han detallado sólo una parte importante en las tablas anteriores, se ha extraído la terminología que ha sido representada en la ontología de Modelado del Estudiante por medio de conceptos, atributos y relaciones. De este modo, hemos identificado los términos (conocidos también como predicados) y los objetos en el universo de discurso (instancias).

### 5.2.3 Reutilización y re-ingeniería de recursos ontológicos y no ontológicos

Las necesidades identificadas para la construcción de la ontología general de Modelado del Estudiante no se han podido cubrir con ningún tipo de recurso ontológico o no ontológico por lo que, la ontología ha sido creada desde cero siguiendo el escenario 1 de desarrollo de la metodología NeOn. Sin embargo, como veremos en el capítulo siguiente, ha sido necesario el escenario 2 de la metodología NeOn al extender esta ontología general para el entorno de aprendizaje de GUIs (editor de texto). En este sentido, es importante destacar en la aplicación de la metodología para el desarrollo de la ontología de este trabajo los dos aspectos claves siguientes: En primer lugar, el peso representativo de los expertos del dominio así como el consenso al desarrollar la metodología y, en segundo lugar, la evolución de la ontología dada su envergadura. En este último aspecto, también es necesario volver a destacar la necesidad de la modularización de la ontología y su extensión y especialización para los dos entornos de aprendizaje a los que se ha aplicado y que se muestran en las secciones siguientes.

### 5.2.4 Descripción general de la Ontología de Modelado del Estudiante

La ontología está compuesta por las siguientes ontologías modulares (Figura 5.2):

- *Ontología Student_Information*. Esta ontología representa la información específica de cada estudiante.

- *Ontología Student_Profile*. Esta ontología representa la información personal de un estudiante: datos personales, características físicas y psicológicas, preferencias de interacción, preferencias de estilos de aprendizaje, rasgos de personalidad, etc.

- *Ontología Learning_Objective*. Esta ontología posibilita la especificación de objetivos en varios dominios (afectivo, psicomotor y cognitivo) y con diferentes niveles de abstracción (objetivos de mayor nivel de abstracción o didácticos, y objetivos de menor nivel de abstracción u objetivos específicos).

  La descripción de los objetivos de aprendizaje en el Modelado del Estudiante es uno de los aspectos claves y más novedosos de esta ontología. Esta característica da soporte a un nuevo mecanismo de diagnóstico pedagógico presentado en el apartado 5.3, que permite inferir a partir del comportamiento del estudiante el estado de los objetivos (alcanzados o no, por el alumno durante su aprendizaje), a diferencia de otros modelados del estudiante propuestos hasta el momento, que infieren el estado de los objetos de conocimiento del estudiante a partir de su comportamiento. Sin embargo, la interrelación existente entre las ontologías modulares de objetivos de conocimiento y objetos de conocimiento permite también, indirectamente, realizar a posteriori este último tipo de diagnóstico denominado diagnóstico cognitivo.

- *Ontología Learning_Object*. Esta ontología describe los principales tipos de elementos de conocimiento que se pueden aprender en una determinada actividad educativa. Los objetos de aprendizaje se han clasificado en objetos que describen conocimiento estructural y objetos que describen conocimiento procedimental.
- *Ontología Student_State*. Esta ontología describe el estado del conocimiento actual del estudiante, su actuación acumulada (respecto a la ejecución de sesiones, actividades, acciones y cumplimientos de sus precondiciones asociadas), su estado pedagógico (grado de realización del plan de aprendizaje, cursos, actividades, etc.), el estado actual de los objetivos de aprendizaje, su estado emocional, el estado de las capacidades generales y competencias del estudiante (nivel de atención, de memoria, etc.).
- *Ontología Student_Trace*. Esta ontología contiene un registro detallado del comportamiento del estudiante durante una sesión de aprendizaje. A partir de este registro se obtienen ciertas informaciones acumulativas en la Ontología del Estado del Estudiante. Las trazas especificadas en la ontología mantienen un registro tanto de las acciones del estudiante como de las intervenciones del tutor, tales como instrucciones o pistas. Asimismo, incluye un registro de cómo han cambiado los estados de los objetivos de aprendizaje a lo largo del tiempo.
- *Ontología Student_Monitoring*. Esta ontología describe las características relativas al procedimiento de seguimiento de ciertas variables durante el aprendizaje del estudiante.

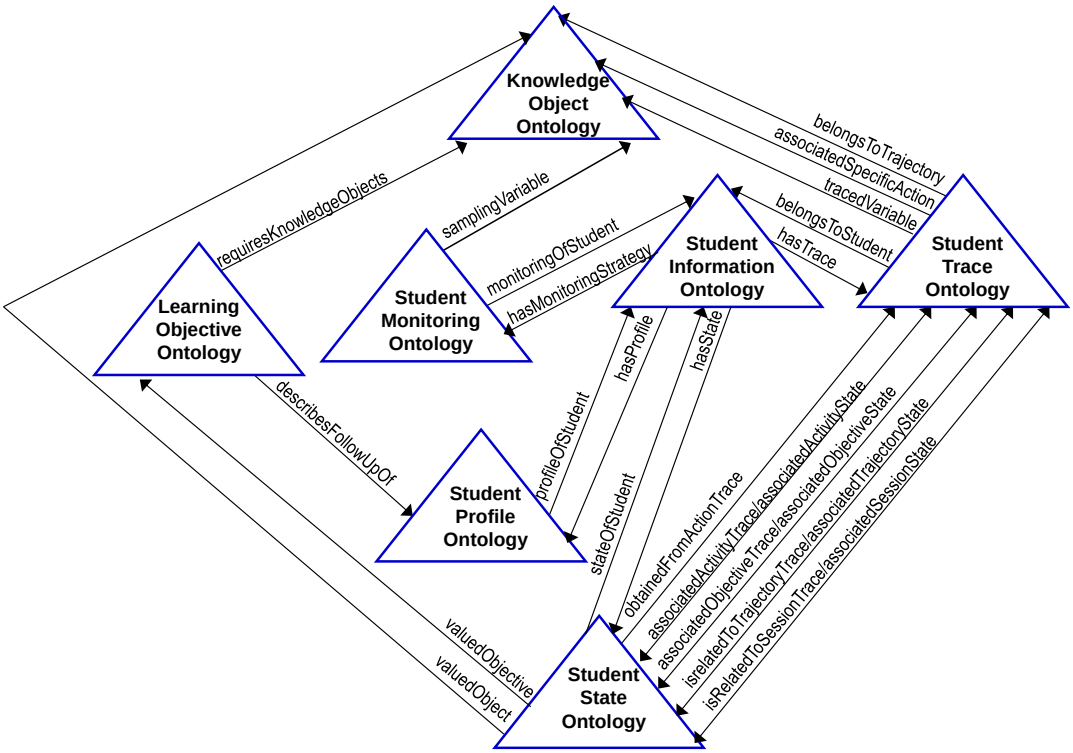


**Figura 5.2:** Principales relaciones ad-hoc entre las ontologías modulares

### 5.2.5 Descripción detallada de la Ontología de Modelado del Estudiante

Las ontologías antes reseñadas engloban diversos aspectos del conocimiento del estudiante. A continuación, se detallan las jerarquías de subclases añadidas a cada una de las ontologías que componen la ontología de Modelado del Estudiante.

**Student_Profile**. La jerarquía de clases consideradas se muestran en la Figura 5.3.

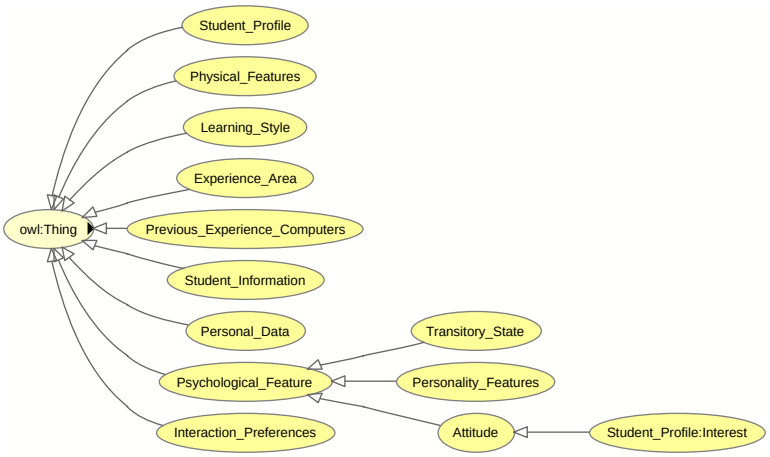


**Figura 5.3:** Jerarquía de clases de la ontología **Student_Profile** en el ME

En la tabla A.1 del apéndice, se describen cada una de las clases de la ontología **Student_Profile**, su interrelación así como las propiedades asociadas.

**Student_Trace**. Las clases de esta ontología describen aspectos de la traza de lo que hace el estudiante durante una actividad pedagógica. Esta información, como se verá en el apartado 5.3, es clave para realizar un diagnóstico cognitivo adecuado del estado de conocimiento del estudiante. A continuación, se muestra la jerarquía relativa a la traza del estudiante (Figura 5.4) y la descripción detallada de sus clases en la tabla A.6 del apéndice.

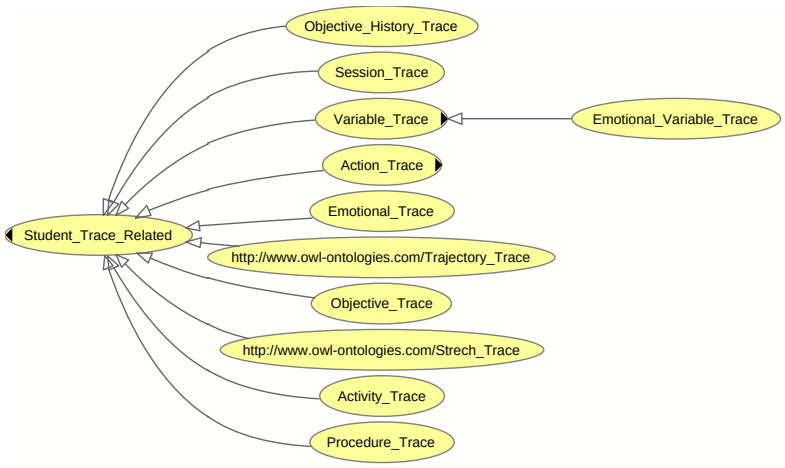


**Figura 5.4:** Jerarquía de la ontología **Student_Trace** en el ME

**Student_Monitoring**. Las clases de esta ontología describen aspectos que caracterizarán el procedimiento de seguimiento del estudiante en su aprendizaje. A continuación, se muestra su jerarquía relativa (Figura 5.5) y la descripción detallada de sus clases en la tabla A.7 del apéndice.

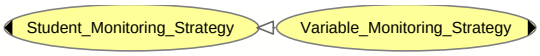


**Figura 5.5:** Jerarquía de la ontología **Student_Monitoring** en el ME

**Student_State**. Las clases de esta ontología describen, para un estudiante, datos obtenidos dinámicamente durante el desarrollo de la actividad y tras su finalización. Estos datos son usados principalmente por el módulo de tutoría del sistema y permitirán establecer el estado general del estudiante durante su aprendizaje. A continuación, se muestra la jerarquía relativa al estado del estudiante (Figura 5.6). La descripción detallada de sus clases se muestra en la tabla A.5 del apéndice.


**Figura 5.6:** Jerarquía de la ontología **Student_State** en el ME

Una ontología adicional, la ontología denominada **Student_Information**, se crea como un conjunto de toda la información específica para un estudiante, y que incluye las cuatro ontologías anteriores.

Además de las cinco ontologías previas, se han identificado tres ontologías más de soporte: Learning_Objective y Knowledge_Object, descritas a continuación. Estas ontologías se pueden considerar independientes del estudiante, en el sentido de que permiten definir instancias de conceptos que pueden ser compartidas por varios estudiantes, en contraposición a las restantes, que son dependientes del estudiante.

**Learning_Objective**. Uno de los aspectos importantes de esta ontología es que da la posibilidad de especificar objetivos en varios dominios y con diferentes niveles de abstracción.

El diseño de cualquier materia, desde el punto de vista del módulo de tutoría, se ha considerado descompuesto en cursos. A su vez, cada curso estará formado por una o más fases y cada fase estará constituida por una o más actividades. Asimismo, desde el punto de vista de los objetivos, también se han descompuesto en varios niveles: los objetivos más generales en la jerarquía de objetivos se consideran pertenecientes a las fases en las que se dividen los cursos (denominados *Didactic_Objective*) y los objetivos de menor nivel de abstracción o "grano más fino" se consideran pertenecientes a las actividades (denominados *Specific_Objective*). Un objetivo específico se puede definir en varios niveles de dominio (cognitivo, afectivo o psicomotor).

De acuerdo al tipo de actividad, los objetivos considerados pueden pertenecer a uno o varios de los dominios citados previamente. Por ejemplo, en una actividad perteneciente a un curso virtual de entrenamiento en una central nuclear donde puede existir riesgo de radiación deberían definirse objetivos a nivel afectivo del tipo *Stress_Control*, *Fear_Control*, etc. Asimismo, en un curso virtual de formación de bailarines sería necesario definir, para determinadas actividades, entre otros, objetivos a nivel psicomotor del tipo *Objective_Basis_Fundamental_Movement*, *Objective_Skilled_Movement*, *Objective_Communication_Non_Discursive*, etc. Algunos ejemplos de instancias de objetivos pueden verse en los dos prototipos de demostración incluidos más adelante en este apartado.


**Figura 5.7:** Jerarquía de clases de la ontología **Learning_Objective** en el ME

La descripción detallada de la jerarquía de objetivos de aprendizaje (Figura 5.7) se presenta en la tabla A.2 del apéndice.

**Knowledge_Object**. Los objetos de aprendizaje en esta ontología se han clasificado en función del tipo de conocimiento subyacente: objetos que describen un conocimiento estructural (clase **Structural_Knowledge** en la jerarquía de objetos) y objetos que describen un conocimiento procedimental (clase **Procedural_Knowledge** en la jerarquía de objetos).

A continuación, se muestra la jerarquía principal de **Structural_Knowledge** (Figura 5.8) y la descripción detallada de sus clases en la Tabla A.3 del apéndice.


**Figura 5.8:** Jerarquía de *Structural_Knowledge* en la ontología **Knowledge_Object** del ME

Del mismo modo, en la Figura 5.9 se muestra la jerarquía principal de **Procedural_Knowledge** y la descripción detallada de sus clases en la Tabla A.4 del apéndice.


**Figura 5.9:** Jerarquía de *Procedural_Knowledge* en la ontología **Knowledge_Object** del ME

## 5.3 Método de diagnóstico

El método de diagnóstico presentado a continuación está soportado por un motor de inferencia encargado de realizar el proceso de razonamiento deductivo. El motor de inferencia elegido ha sido Jena por todas las capacidades ya mencionadas no sólo en cuanto a su capacidad de procesamiento del lenguaje OWL, en el que está implementada la ontología del ME propuesta en esta tesis (apartado 5.2), sino también por sus características tanto de encadenamiento como de almacenaje de tripletas (apartado 3.3.5).

A diferencia de anteriores propuestas, el método de diagnóstico propuesto llevará a cabo una fase de diagnóstico pedagógico, encargada de determinar los objetivos alcanzados o no por el estudiante a partir de su comportamiento. Esta fase de diagnóstico es previa al diagnóstico cognitivo propiamente dicho.

Asimismo, el método de diagnóstico propuesto debe ser no monótono. Para ello, entre las técnicas de IA de razonamiento no monótono ya estudiadas y aplicadas con éxito por otros autores nos hemos decantado por el ATMS como componente del método a implementar. El motivo de esta elección ha sido la adaptación de las características de esta técnica al diagnóstico cognitivo del ME: a) permite manejar conocimiento incompleto (estado cognitivo del alumno) por medio de la formulación de hipótesis para que el proceso de razonamiento pueda continuar, b) permite invalidar eventualmente algunas hipótesis asumidas si son éstas rechazadas a lo largo del proceso de razonamiento, así como retraer las inferencias realizadas a partir de ellas, y c) permite tener en cuenta la naturaleza del alumno que interactúa con el SIT durante su aprendizaje

(el estudiante puede aprender nuevos conceptos u olvidar otros ya adquiridos).

El algoritmo de inferencia debe abordar el problema de la aparición de contradicciones. Dado el enfoque pedagógico presentado en este método (véase apartado 5.1), es posible la coexistencia en un instante determinado en el ME de estados contradictorios de un mismo objetivo definido en la actividad de aprendizaje que realiza el estudiante. Como se describirá más adelante, hay diversos motivos por los que surgen datos inconsistentes (descuidos del estudiante, cambios en su mente tras una tutoría, etc.). Estas causas serán fundamentales para establecer, en cada caso, las pautas más adecuadas a seguir en la estrategia de tutoría a aplicar a continuación. En la Figura 5.10, se muestra un diagrama del método de diagnóstico presentado en este trabajo.

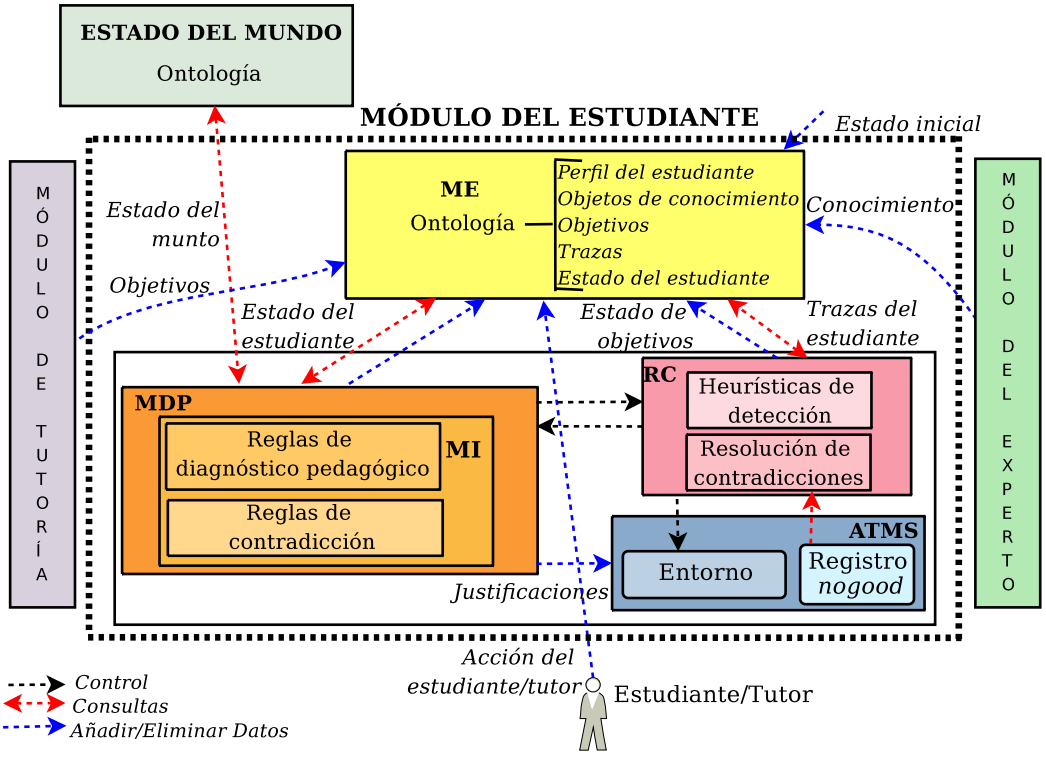


**Figura 5.10:** Esquema del método de diagnóstico

Los principales módulos que interactúan entre sí para realizar el diagnóstico pedagógico son los siguientes:

**MDP** es el módulo de diagnóstico pedagógico que, soportado por un conjunto de reglas de diagnóstico pedagógico, y realizando consultas a la ontología del ME y a la ontología que describe el estado del mundo en cada instante, es el responsable de inferir qué objetivos son adquiridos o no adquiridos por el estudiante. Se ha elegido para ello como motor de razonamiento Jena y las inferencias realizadas son comunicadas en forma de justificaciones a un TMS de tipo ATMS.

**ATMS** se encarga de controlar la consistencia de un conjunto de suposiciones usadas por el MDP y determinar las creencias que se sostienen y cuáles no.

**RC** es el gestor de conflictos, que se encarga de dos tareas principales: a) determinar las causas posibles de las contradicciones que aparezcan. Para ello, usa ciertas heurísticas independientes (o dependientes) del dominio y, b) resolver las contradicciones buscando un entorno nuevo consistente con el que el MDP pueda seguir su proceso de inferencia. Este módulo también debe realizar diversas consultas a la ontología del ME y modificar, si fuera necesario, ciertos estados de objetivos para obtener un entorno consistente con el que pueda seguir trabajando el MDP.

De acuerdo al esquema propuesto, los pasos generales del método son los que se enumeran a continuación:

1. Se establece el estado inicial del ME con supuestos sobre el estado de los objetivos diseñados para la actividad de aprendizaje a realizar por un estudiante. Para cada objetivo, el supuesto sobre su estado inicial puede ser *adquirido*, *no-adquirido* o *desconocido*.

2. El MDP debe informar a ATMS de los supuestos referentes a los estados de los objetivos. Esta comunicación se lleva a cabo en forma de nodos asumidos, tal y como se definen en ATMS (véase apartado 3.1.5).

3. El estudiante realiza una acción como parte de la actividad de aprendizaje que está llevando a cabo. También es posible que esta acción proceda del tutor cuando, durante el aprendizaje proporcione una orden al estudiante (puede entenderse como tal también el dar una pista), si así lo decidiera.

4. La realización de la acción previa, provoca el disparo de algunas de las reglas de diagnóstico definidas en el MDP usando un encadenamiento hacia delante del razonador Jena y que precisará de diversas consultas acerca de parte del estado actual de dos ontologías: la ontología del ME y la ontología sobre el estado del mundo.

5. El MDP comunica a ATMS el proceso de razonamiento realizado (paso 4) mediante una justificación que será registrada por el sistema de razonamiento no monótono. Si, en el proceso de razonamiento, el MDP detectara una contradicción (mediane el disparo de reglas definidas con este fin), se realizan las siguientes tareas:

   5.1 La contradicción también es comunicada a ATMS en forma de justificación. Este sistema obtiene el entorno (conjunto de suposiciones que causan la inconsistencia) que sostiene esta contradicción mediante un encadenamiento hacia atrás a partir de la red de dependencias entre nodos que posee hasta ese instante del proceso. El entorno obtenido se almacena en el registro denominado *nogood*.

   5.2 El MDP también invoca al módulo RC para que detecte los diferentes tipos de inconsistencias de acuerdo a las causas que las provocan. Esto se realizará en base a ciertas heurísticas definidas. Asimismo, el módulo RC es el responsable de buscar candidatos de entornos consistentes (según la causa de la contradicción y revisando las suposiciones que sostienen la inconsistencia) e invoca con ellos a ATMS.

   5.3 ATMS, como consecuencia de lo anterior, comprueba la consistencia de los nuevos entornos chequeando el registro *nogood*. Si uno de los entornos candidatos es consistente, ATMS sustituye el entorno actual por éste, de tal forma que el MDP pueda

seguir razonando con datos derivados del entorno consistente seleccionado. La resolución de la contradicción supondrá que el módulo RC lleve a cabo la modificación del estado de algún objetivo en la ontología del ME.

### 5.3.1 Estado inicial del Modelo del Estudiante

Para establecer el estado inicial del Modelo del Estudiante hay dos alternativas extremas: a) considerar, como en HSMIS (Ikeda et al. (1993)), para el modelo inicial del estudiante, que se trata de un estudiante excelente, es decir, comprende el material de aprendizaje muy bien. Para ello, tal y como se vio en el apartado 2.3.6.4, este método introduce un módulo que genera oráculos virtuales, es decir, respuestas del estudiante basadas en la fiabilidad del ME actual, de tal modo que el tutor plantea menos preguntas, y la información necesaria se sustituye con respuestas correctas y, b) considerar un modelo "vacío", es decir, el profesor no sabe nada de antemano sobre el estudiante. Con esta alternativa, el sistema tendría que realizar muchas preguntas o plantear muchos problemas al estudiante para poder obtener su estado de conocimiento sobre el material concreto de aprendizaje.

Las dos alternativas previas, desde nuestro punto de vista, no son muy razonables en la tutoría real. Por esta razón, a diferencia de ellas, en el método planteado se considerará una alternativa intermedia. El estudiante, de acuerdo a la actividad que va a desarrollar dentro de un determinado plan de enseñanza, podrá tener asociado un modelo inicial que suponga ciertos objetivos ya alcanzados, requeridos por la actividad a realizar, otros no alcanzados, y otros cuya adquisición se desconozca a priori.

El estado inicial de los objetivos, por lo tanto, ya no es fijo y la propiedad *acquired* asociada a la clase *Objective_State* en la ontología del ME permitirá tres valores: *true* (el sistema sabe que el estudiante alcanzó el objetivo), *false* (el sistema sabe que el estudiante no alcanzó el objetivo) y *unknown* (el sistema no sabe nada acerca del alcance del objetivo por el estudiante).

Un objetivo se considerará ya adquirido cuando se haya alcanzado un número determinado de veces. Este número se fija mediante la propiedad *levelReliability* en la clase *Specific_Objective*, y mediante la propiedad *levelCurrentReliability* se establecerá el número de veces que se ha alcanzado ese objetivo hasta la actualidad para un determinado estudiante. Hasta que no se alcance el nivel de fiabilidad establecido para los objetivos específicos, no se considerará el objetivo completamente adquirido (un alumno puede alcanzar en el desarrollo de una actividad un objetivo concreto por simple azar).

Con respecto al resto del conocimiento en la ontología del ME, se considera inicialmente la información referente al curso (objetivos de aprendizaje y sus objetos asociados, así como la interrelación entre ellos, el plan de la actividad a desarrollar del curso y sus operadores asociados a esa actividad, etc.), información sobre el perfil del estudiante, etc. Las instancias de la traza del estudiante y el estado del estudiante se irán paulatinamente añadiendo a la ontología, en función de las acciones realizadas por el estudiante durante el desarrollo de las diferentes actividades de aprendizaje. Esta información se arrastra de una sesión a la siguiente. Del mismo modo, el razonamiento realizado por el módulo de diagnóstico pedagógico irá modificando las instancias en ciertas jerarquías de la ontología: estado de objetivos, cierta información del perfil del estudiante, etc.

### 5.3.2 Módulo de diagnóstico pedagógico

De acuerdo al enfoque de diseño adoptado, es necesario definir un conjunto de reglas para realizar la primera fase de diseño que se ha denominado Diagnóstico Pedagógico. Estas reglas

inferirán los nuevos objetivos alcanzados a partir de las acciones realizadas por el estudiante y, en algunas ocasiones, de los objetivos ya alcanzados inferidos a partir del comportamiento previo del estudiante. Además, ciertas reglas pueden inferir que el estudiante no ha logrado ciertos objetivos; en este caso la información que el ME proporciona acerca de la traza del estudiante estudiante será imprescindible para determinar si el estudiante ha olvidado en un momento determinado de su aprendizaje algún conocimiento o, si realmente no ha alcanzado nunca esos objetivos.

Para definir las reglas de este módulo, se ha desarrollado la siguiente taxonomía de criterios de diagnóstico en la que se han simplificado algunas jerarquías para mayor claridad (Figura 5.11).

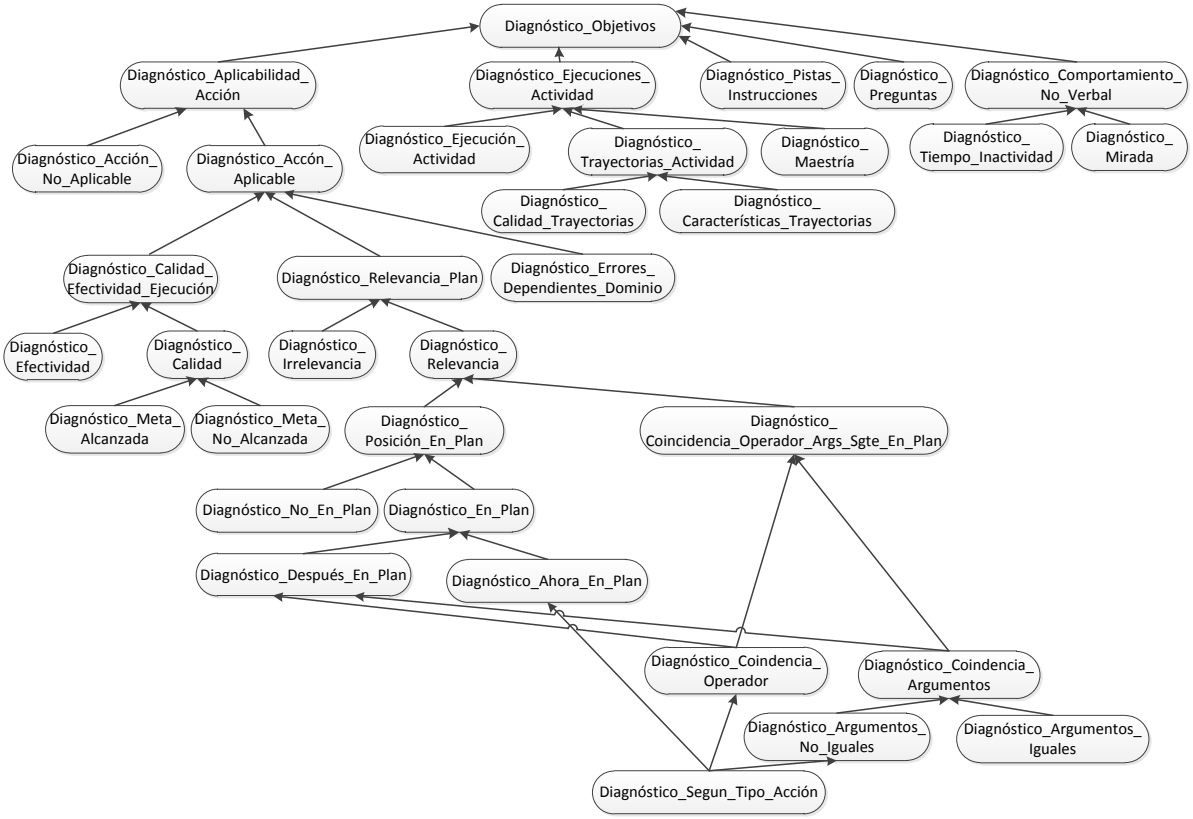


**Figura 5.11:** Taxonomía de criterios de diagnóstico de objetivos alcanzados

La taxonomía mostrada servirá de soporte para la definición posterior de un conjunto de patrones con los que se equiparen las reglas de diagnóstico dando lugar a las siguientes categorías de reglas: *diagnóstico de acuerdo al tipo de acción que el estudiante ejecuta, diagnóstico de las ejecuciones de la actividad, diagnóstico basado en el número y tipo de preguntas formuladas por el estudiante, diagnóstico basado en las pistas e instrucciones proporcionadas por el tutor*, y *diagnóstico basado en el comportamiento no verbal del estudiante*. A continuación se describen cada una de estas categorías.

#### 5.3.2.1 Diagnóstico de acuerdo al tipo de acción que el estudiante ejecuta

Esta categoría se subdivide en varios tipos de diagnósticos y sus reglas asociadas, cuyo estudio se describe en los siguientes apartados. Esta división se origina del análisis de los tipos de acciones que el estudiante puede llevar a cabo desde distintas perspectivas: *según la aplicabilidad de la acción, según la efectividad/calidad de la ejecución de la acción, según la relevancia de la acción respecto al plan*, y *acciones erróneas dependientes del dominio*.

**Diagnóstico de acciones basado en su aplicabilidad**

Estas reglas inferirán los objetivos de aprendizaje que pueden asumirse si el estudiante simplemente intenta realizar una acción.

- Para cualquier tipo de acción (operador) que el estudiante intente ejecutar (acción no aplicable porque no cumple alguna precondición o acción aplicable), se define la siguiente regla de diagnóstico:

$$R_1: \text{SI } \text{Intenta\_Aplicar}(accx) \rightarrow \text{Sabe}(\text{Existe}(accx)) \tag{5.1}$$

$$\begin{aligned} R_2: \text{SI } & \text{Requiere\_Objeto}(accx, objy) \wedge \\ & \text{Intenta\_Aplicar\_Sobre\_Objeto}(accx, objy) \rightarrow \\ & \text{Sabe}(\text{Existe}(objy)) \wedge \text{Sabe}(\text{Donde\_Esta}(objy)) \end{aligned} \tag{5.2}$$

- Para cualquier tipo de acción aplicable (acción ejecutada por el estudiante) y, además, que se aplica a un objeto, se define la siguiente regla de diagnóstico:

$$R_3: \text{SI } \text{Aplicar\_A\_Objeto}(accx, objy) \rightarrow \text{Sabe}(\text{Es\_Aplicable}(accx, objy)) \tag{5.3}$$

- Para acciones aplicables y que establecen relación entre varios objetos (estas acciones se identifican en la ontología mediante *Modifica_Relacion_Entre_Objetos*), se define la siguiente regla:

$$\begin{aligned} R_4: \text{SI } & \text{Aplicar}(accx) \wedge \text{Modifica\_Relacion\_Entre\_Objetos}(accx, objx, objy) \rightarrow \\ & \text{Sabe}(\text{Son\_Relacionables}(objx, objy)) \end{aligned} \tag{5.4}$$

- Diagnóstico de acciones no aplicables, es decir, diagnóstico de acciones que al intentar el estudiante ejecutarlas alguna de las precondiciones asociadas a estas acciones no se cumplen:

$$\begin{aligned} R_5: \text{SI } & \text{Intenta\_Aplicar}(accx) \wedge \text{No\_Cumple}(accx, precondy) \wedge \\ & \text{Sgte\_Accion\_En\_Plan}(accy) \wedge \neg\text{Eq}(accx, accy) \rightarrow \\ & \neg\text{Sabe}(\text{Requiere\_Precond}(accx, precondy)) \wedge \\ & \neg\text{Sabe}(\text{Sgte\_Accion\_En\_Plan}(accy)) \end{aligned} \tag{5.5}$$

$$
\begin{aligned}
R_6: \text{SI } & \text{Intenta\_Aplicar}(accx) \wedge \text{No\_Cumple}(accx, precondy) \wedge \\
& \text{Sgte\_Accion\_En\_Plan}(accy) \wedge \\
& \neg\text{Eq}(\text{Operador}(accx, opx), \text{Operador}(accy, opy)) \rightarrow \\
& \neg\text{Sabe}(\text{Requiere\_Precond}(accx, precondy)) \wedge \\
& \neg\text{Sabe}(\text{Sgte\_Accion\_En\_Plan}(accy)) \wedge \neg\text{Sabe}(plan)
\end{aligned} \tag{5.6}
$$

- Acciones de movimiento no aplicables con objeto y con restricciones de objetos. Este tipo de acciones no pertenecen al caso de estudio (actividad de programación de lavadora) ya que el traslado de objetos (detergentes, cesto de ropa, etc.) en el escenario de esta actividad no tiene asociada estos tipos de restricciones.

**Diagnóstico de acciones de acuerdo a la efectividad/calidad de su ejecución**

Estas reglas inferirán los objetivos de aprendizaje que pueden asumirse si el estudiante ejecuta una acción, de acuerdo al logro (o no) de las metas asociadas a la acción (*diagnóstico basado en la efectividad de ejecución de la acción*), y cómo estas metas han sido alcanzadas (*diagnóstico de acuerdo a la calidad de la ejecución de la acción*).

- *Diagnóstico basado en la efectividad de ejecución de la acción*

  - Para cualquier tipo de acción que aplique el estudiante, se definen las siguientes reglas de diagnóstico:

$$
\begin{aligned}
R_7: \text{SI } & \text{Aplicar}(accx) \wedge \text{Obtener\_Meta}(accx, metay) \rightarrow \\
& \text{Es\_Capaz\_De\_Obtener}(accx, metay) \wedge \\
& \text{Es\_Capaz\_De\_Aplicar}(accx) \wedge \text{Sabe}(\text{Usada\_Para}(accx, metay))
\end{aligned} \tag{5.7}
$$

$$
\begin{aligned}
R_8: \text{SI } & \text{Aplicar}(accx) \wedge \text{Puede\_Obtener\_Meta}(accx, metay) \wedge \\
& \neg\text{Obtener\_Meta}(accx, metay) \rightarrow \\
& \neg\text{Sabe}(\text{Se\_Ejecuta}(accx, modoEjecuciony))
\end{aligned} \tag{5.8}
$$

- *Diagnóstico basado en la calidad de ejecución de la acción.*

  Este diagnóstico es fundamental para las acciones denominadas *Actions_With_Partial_Level_Fullfilment* incluidas en la ontología del estudiante, tales como acciones de movimiento del estudiante en un EV, donde la longitud del movimiento puede ser menor, igual o mayor que la trayectoria óptima desde un punto al otro en el EV.

  - Según el tipo de acción ejecutada y el escenario de la actividad, el estudiante puede llegar a la meta de la acción con un nivel de calidad u otro. Hay acciones como trasladarse con un objeto, moverse de un punto de origen a un punto de destino, acciones que se realizan en el dominio de la cirugía (por ejemplo, suturar), etc., en las

que es necesario diagnosticar el grado de calidad en la obtención de la meta asociada a la ejecución de la acción. Otras, sin embargo, implican sólo el diagnóstico del cumplimiento de su meta (generalmente, abrir, cerrar, coger, etc.).

$$\begin{aligned} R_9: \text{SI } & \text{Aplicar(accx)} \wedge \text{Accion\_Cumplimiento\_Parcial(accx)} \wedge \\ & \text{Puede\_Obtener\_Meta\_Con\_Calidad(accx, metay, calidadx)} \wedge \\ & \text{Calidad\_Optima\_Meta(accx, metay, calidady)} \wedge \\ & \text{MayorEq(calidadx, calidady)} \rightarrow \\ & \text{Es\_Capaz\_De\_Aplicar(accx, Calidad\_Optima)} \end{aligned} \tag{5.9}$$

$$\begin{aligned} R_{10}: \text{SI } & \text{Aplicar(accx)} \wedge \text{Accion\_Cumplimiento\_Parcial(accx)} \wedge \\ & \text{Puede\_Obtener\_Meta\_Con\_Calidad(accx, metay, calidadx)} \wedge \\ & \text{Calidad\_Optima\_Meta(accx, metay, calidady)} \wedge \\ & \text{Calidad\_Pesima\_Meta(accx, metay, calidadz)} \wedge \\ & \text{Menor(calidadx, calidady)} \wedge \text{Mayor(calidadx, calidadz)} \rightarrow \\ & \text{Es\_Capaz\_De\_Aplicar(accx, Calidad\_Media)} \end{aligned} \tag{5.10}$$

$$\begin{aligned} R_{11}: \text{SI } & \text{Aplicar(accx)} \wedge \text{Accion\_Cumplimiento\_Parcial(accx)} \wedge \\ & \text{Puede\_Obtener\_Meta\_Con\_Calidad(accx, metay, calidadx)} \wedge \\ & \text{Calidad\_Pesima\_Meta(accx, metay, calidady)} \wedge \\ & \text{Menor(calidadx, calidady)} \rightarrow \\ & \text{Es\_Capaz\_De\_Aplicar(accx, Calidad\_Pesima)} \end{aligned} \tag{5.11}$$

**Diagnóstico de acciones aplicadas según su relevancia con respecto al plan**

Distinguimos dos subcategorías: *Diagnóstico de acciones aplicadas con coincidencia de operador y/o argumentos con la siguiente acción en el plan* y *Diagnóstico de acciones aplicadas de acuerdo a su posición en el plan.*

- Diagnóstico de acciones aplicadas que coinciden con la siguiente acción en el plan.[3].

$$\begin{aligned} R_{12}: \text{SI } & \text{Aplicar(accx)} \wedge \text{Sgte\_Accion\_Plan(accx)} \rightarrow \\ & \text{Sabe(Sgte\_Accion\_Plan(accx))} \end{aligned} \tag{5.12}$$

- Diagnóstico de acciones aplicadas que coinciden con el operador de la siguiente en el plan y alguno de sus argumentos también coinciden (por ejemplo, el estudiante coge un objeto, *objx*, de una determinada posición, y coincide la acción y su argumento con la siguiente acción en el plan y con el objeto sobre el que se aplica, *objx*. También, el argumento puede

[3] *accx* no es aquí una acción instanciada, sino genérica: mover, coger, pulsar, etc.

ser un objeto destino, *objy*, de la acción como, por ejemplo, coger un objeto situado en la posición del objeto *objy*).

$$\begin{aligned} R_{13}: \text{SI } & \text{Aplicar\_A\_Objeto(accx, objx)} \wedge \text{Sgte\_Accion\_En\_Plan(accx)} \wedge \\ & \text{Obj\_Sgte\_Accion\_Plan(objx)} \rightarrow \text{Sabe(Donde\_Esta(objx))} \\ & \text{Sabe(Obj\_Sgte\_Accion\_Plan(objx))} \wedge \text{Sabe(Reconocer(objx))} \end{aligned} \tag{5.13}$$

- Diagnóstico de acciones aplicadas, que coincide con el operador de la siguiente acción en el plan y alguno de sus argumentos, *objx*. Este objeto es parte del objeto *objy* (por ejemplo, si el estudiante echa detergente de lavado en la cubeta de detergente del cajón de detergentes -el cajón consta de 3 cubetas: cubeta de lejía, cubeta de suavizante y cubeta de detergente de lavado). En este caso, se define la siguiente regla:

$$\begin{aligned} R_{14}: \text{SI } & \text{Aplicar\_A\_Objeto(accx, objx)} \wedge \text{Sgte\_Accion\_En\_Plan(accx)} \wedge \\ & \text{Obj\_Sgte\_Accion\_Plan(objx)} \wedge \text{Parte\_De(objx, objy)} \wedge \\ & \text{Parte\_De(objz, objy)} \rightarrow \\ & \text{Sabe(Parte\_De(objx, objy))} \wedge \\ & \text{Sabe\_Elegir(Parte\_De(objx, objy))} \end{aligned} \tag{5.14}$$

- Diagnóstico de acciones aplicadas que modifican la relación entre el estudiante y el objeto sobre el que se aplica, *objx* situado en la posición *posx* donde se encuentran todos los objetos de una clase particular, *clasex* (por ejemplo, el estudiante coge un detergente, el detergente de lavado, de la mesa de detergentes), y se trata de una acción que coincide con la siguiente acción del plan. En este caso, se definen las siguientes reglas:

$$\begin{aligned} R_{15}: \text{SI } & \text{Aplicar(accx)} \wedge \text{Sgte\_Accion\_En\_Plan(accx)} \wedge \\ & \text{Modifica\_Relacion\_Estudiante\_Objeto(accx, objy)} \wedge \\ & \text{Pos\_Relativa\_Objetos\_De\_Clase(clasex, posx)} \wedge \\ & \text{Pos\_En\_Sgte\_Accion\_Plan(posx)} \wedge \\ & \text{Subclase\_De(objy, clasex)} \wedge \text{Esta\_Situado\_En(objy, posx)} \rightarrow \\ & \text{Sabe\_Elegir(Objetos\_De\_Clase(objy))} \wedge \\ & \text{Sabe(Instancia\_De(objy, clasex))} \end{aligned} \tag{5.15}$$

$$\begin{aligned} R_{16}: \text{SI } & \text{Aplicar\_A\_Posicion(accx, posx)} \wedge \\ & \text{Sgte\_Accion\_En\_Plan(accx)} \wedge \text{Pos\_En\_Sgte\_Accion\_Plan(posx)} \wedge \\ & \text{Pos\_Relativa\_Objetos\_De\_Clase(clasex, posx)} \wedge \\ & \text{Sabe(Obj\_Sgte\_Accion\_Plan(objx))} \wedge \neg\text{Sabe(Donde\_Esta(objx))} \wedge \\ & \text{Sabe(Subclase\_De(objx, clasex))} \rightarrow \\ & \text{Sabe(Donde\_Esta(Objetos\_De\_Clase(clasex)))} \end{aligned} \tag{5.16}$$

Si el estudiante sabe la finalidad del siguiente objeto que debe usar para continuar la actividad y usa ese objeto como herramienta entonces, sabe para qué sirve el objeto utilizado.

$$\begin{aligned} R_{17}: \text{SI } & \text{Aplicar\_A\_Objeto(accx, objy)} \wedge \text{Sgte\_Accion\_En\_Plan(accx)} \wedge \\ & \text{Sabe(Obj\_Sgte\_Accion\_Plan(objy))} \wedge \\ & \text{Usa\_Objeto\_Como\_Herramienta(accx, objy)} \rightarrow \\ & \text{Sabe(Para\_Que\_Sirve(objy))} \end{aligned} \tag{5.17}$$

- Diagnóstico de acciones aplicadas y que coinciden con la siguiente acción en el plan pero no coinciden sus argumentos. Este diagnóstico depende del tipo de acción.

  *Acciones que implican interacción con un objeto.* Para este tipo de acciones que se aplican a un objeto, en las que coincide el operador aplicado con el siguiente de acuerdo al plan pero sin coincidir los objetos a los que se aplica el operador, se puede suponer que *El estudiante no sabe el objeto que se debe usar en la siguiente acción.* Por ejemplo, el estudiante *Echa el detergente de lavado en la cubeta del suavizante de la lavadora* y la siguiente acción de acuerdo al plan es *Echar el detergente de lavado en la cubeta de detergente.*

$$\begin{aligned} R_{18}: \text{SI} & \text{Aplicar\_A\_Objeto(accionx, objx)} \wedge \text{Obj\_En\_Sgte\_Accion(objy)} \wedge \\ & \neg\text{Eq(objx, objy)} \rightarrow \\ & \text{Annadir\_ME(Know(}\neg\text{Sabe(Objeto\_En\_Sgte\_Accion(objy))))} \end{aligned} \tag{5.18}$$

  *Acciones que implican interacción con un objeto pero que no implican modificar su posición.* Por ejemplo, una acción que implique la modificación de la relación entre el estudiante y el objeto como la acción *coger*, o una acción que modifique el estado del objeto pero que no sea su posición, por ejemplo, la acción pulsar un botón. Se considerarán dos tipos de acciones: acciones respecto a una posición concreta (*pdestino*) y acciones respecto a una posición relativa a una clase de objetos (*posobjy*).

  Si no hay coincidencia de argumentos con la acción que correspondía ejecutar en el plan porque hay varias clases de objetos en una posición relativa a una clase de objetos y varias posibles ocurrencias de cada tipo de objeto, el estudiante realiza la acción sobre un objeto que no es el que correspondía en ese momento, y ni siquiera pertenece a la misma clase de objetos (por ejemplo, el estudiante coge un detergente con lejía cuando había que coger alguno de los detergentes de lavado existentes). Para este caso, se define la siguiente regla

de diagnóstico:

$$
\begin{aligned}
R_{19}: \text{SI } & \text{Aplicar\_A\_Objeto}(accx,\ objx) \wedge \\
& \neg\text{Es\_De\_Tipo}(accx, \text{Modifies\_Object\_Position}) \wedge \\
& \text{Sgte\_Accion\_En\_Plan}(accx) \wedge \\
& \text{Pos\_Relativa\_Objetos\_De\_Clase}(clasex, posx) \wedge \\
& \text{Pos\_En\_Sgte\_Accion\_Plan}(posx) \wedge \\
& \text{Obj\_Sgte\_Accion\_Plan}(objx') \wedge \\
& \text{Esta\_Situado\_En}(objx', posx) \wedge \\
& \text{Esta\_Situado\_En}(objx, posx) \wedge \\
& \neg\text{Eq}(\text{Subclase\_De}(objx, clasey),\ \text{Subclase\_De}(objx', clasez)) \rightarrow \\
& \neg\text{Sabe}(\text{Reconocer}(\text{Subclase\_De}(objx', clasez)) \vee \\
& \neg\text{Sabe\_Elegir}(\text{Objetos\_De\_Clase}(clasez))
\end{aligned}
\tag{5.19}
$$

Si no hay coincidencia de argumentos con la acción que correspondía ejecutar en el plan porque hay varias clases de objetos en una posición relativa a una clase de objetos, hay varias posibles ocurrencias de cada tipo de objeto, y el estudiante realiza la acción sobre un objeto que no es el que correspondía en ese momento pero sí pertenece a la misma clase de objetos (por ejemplo, aunque no es el caso de la actividad, coger un detergente de lavado de entre varios existentes en la mesa de lavado y que no era el que correspondía coger). Para este caso, se define la siguiente regla de diagnóstico:

$$
\begin{aligned}
R_{20}: \text{SI } & \text{Aplicar\_A\_Objeto}(accx,\ objx) \wedge \\
& \neg\text{Es\_De\_Tipo}(accx, \text{Modifies\_Object\_Position}) \wedge \\
& \text{Sgte\_Accion\_En\_Plan}(accx) \wedge \\
& \text{Pos\_Relativa\_Objetos\_De\_Clase}(clasex, posx) \wedge \\
& \text{Pos\_En\_Sgte\_Accion\_Plan}(posx) \wedge \\
& \text{Obj\_Sgte\_Accion\_Plan}(objx') \wedge \\
& \text{Esta\_Situado\_En}(objx', posx) \wedge \\
& \text{Esta\_Situado\_En}(objx, posx) \wedge \\
& \text{Eq}(\text{Subclase\_De}(objx, clasey),\ \text{Subclase\_De}(objx', clasey)) \rightarrow \\
& \text{Sabe}(\text{Reconocer}(\text{Subclase\_De}(objx', clasey)) \wedge \\
& \text{Sabe\_Elegir}(\text{Objetos\_De\_Clase}(clasey)) \wedge \\
& \neg\text{Sabe\_Elegir}(objx')
\end{aligned}
\tag{5.20}
$$

*Acciones que modifican la posición del estudiante* (denotadas como *Modifies_Student_Position* en la ontología), *o que modifican la posición de un objeto* (denotadas en la ontología como *Modifies_Object_Position*). Son acciones como, por ejemplo, el desplazamiento de una persona de una posición a otra en el escenario de la actividad (denotada como *Move* en la ontología), o que implican movimiento del objeto, por ejemplo, el traslado de un

objeto de una posición a otra del escenario (denotada como *Move_3D_Object* en la ontología), o el simple movimiento de un objeto como arrastrar el ratón en un entorno GUI (denotada como *Drag_Object* en la ontología). El destino de estas acciones puede venir dado por una posición concreta (*pdestino*), una posición relativa a una clase de objetos (*posobjy*) o una posición relativa a un área dentro del escenario de la actividad (*parea*). En función de la forma en que se exprese dicho destino, los objetivos alcanzados difieren.

Para este tipo de acciones ejecutadas por el estudiante, si hay coincidencia de operador y argumentos (con cualquier forma de definición de la posición de destino), se define la siguiente regla de diagnóstico:

$$
\begin{aligned}
R_{21}: \text{SI } & \text{Aplicar\_A\_Pos}(\text{accx}, \text{pfinal}) \wedge \text{Sgte\_Accion\_En\_Plan}(\text{accx}) \wedge \\
& \text{Pos\_En\_Sgte\_Accion\_Plan}(\text{pfinal}) \rightarrow \\
& \text{Sabe}(\text{Donde\_Esta}(\text{pfinal})) \wedge \\
& \text{Conoce}(\text{Subespacios\_Y\_Conexiones}(\text{porigen}, \text{pfinal})) \wedge \\
& \text{Es\_Capaz\_De\_Construir}( \\
& \quad \text{Ubicacion\_En\_Mapa\_Mental}(\text{porigen}, \text{pfinal})) \wedge \\
& \text{Es\_Capaz\_De\_Construir}(\text{Camino}(\text{porigen}, \text{pfinal}))
\end{aligned}
\tag{5.21}
$$

Para este tipo de acciones, si se ejecutan por el estudiante y hay coincidencia de operador, no hay coincidencia de destino con la acción que corresponde ejecutar en el plan, pero sí coinciden el área en la que se encuentran ambas posiciones de destino, se definen dos reglas. Una para el caso de que la posición de destino venga dada de la forma *pdestino*, y otra para el caso en que dicha posición venga dada por *parea*. Ambas, dada su similitud, se expresan a continuación conjuntamente:

$$
\begin{aligned}
R_{22}: \text{SI } & \text{Aplicar\_A\_Pos}(\text{accx}, \text{posx/area}) \wedge \text{Sgte\_Accion\_En\_Plan}(\text{accx}) \wedge \\
& \text{Pos\_En\_Sgte\_Accion\_Plan}(\text{posx}'/\text{area}') \wedge \\
& \neg\text{Eq}(\text{posx/area}, \text{posx}'/\text{area}')) \wedge \\
& \text{Eq}(\text{Subespacio}(\text{posx/area}), \text{Subespacio}(\text{posx}'/\text{parea}')) \rightarrow \\
& \text{Sabe}(\text{Subespacio}(\text{posx}'/\text{area}')) \wedge \\
& \text{Es\_Capaz\_De\_Construir}( \\
& \quad \text{Ubicacion\_En\_Mapa\_Mental}( \\
& \qquad \text{porigen}, \text{Pos}(\text{Subespacio}(\text{posx}'/\text{area}')))) \wedge \\
& \text{Es\_Capaz\_De\_Construir}( \\
& \quad \text{Camino}(\text{porigen}, \text{Pos}(\text{Subespacio}(\text{posx}'/\text{area}')))) \wedge \\
& \text{Sabe}(\text{Donde\_Esta}(\text{Subespacio}(\text{posx}'/\text{area}'))) \wedge \\
& \text{Sabe}(\text{Reconocer}(\text{Subespacio}(\text{posx}'/\text{area}')))
\end{aligned}
\tag{5.22}
$$

Las dos reglas anteriores difieren un poco para el caso en que el destino del movimiento se exprese relativo a la posición de un objeto (*objy*). Por ejemplo, en una determinada actividad y en el subescenario de la cocina, si se moviera el estudiante a la posición donde

se encuentran los utensilios de cocina en vez de a la posición donde se encuentran los detergentes. A continuación, se añaden para este caso las siguientes reglas de diagnóstico:

$$
\begin{aligned}
R_{23}: \text{SI } & \text{Aplicar\_A\_Pos}(accx, objy) \wedge \text{Sgte\_Accion\_En\_Plan}(accx) \wedge \\
& \text{Pos\_En\_Sgte\_Accion\_Plan}(objy') \wedge \\
& \neg\text{Eq}(objy, objy')) \wedge \\
& \text{Eq}(\text{Subespacio}(objy), \text{Subespacio}(objy')) \rightarrow \\
& \text{Sabe}(\text{Subespacio}(objy')) \wedge \\
& \text{Es\_Capaz\_De\_Construir}( \\
& \quad \text{Ubicacion\_En\_Mapa\_Mental}( \\
& \qquad porigen, \text{Pos}(\text{Subespacio}(objy')))) \wedge \\
& \text{Es\_Capaz\_De\_Construir}( \\
& \quad \text{Camino}(porigen, \text{Pos}(\text{Subespacio}(objy')))) \wedge \\
& \text{Sabe}(\text{Donde\_Esta}(\text{Subespacio}(objy'))) \wedge \\
& \text{Sabe}(\text{Reconocer}(\text{Subespacio}(objy'))) \wedge \\
& \text{Sabe}(\text{Subespacio}(objy)) \wedge \\
& (\neg\text{Sabe}(\text{Reconocer}(objy')) \vee \\
& \neg\text{Sabe}(\text{Pos\_En\_Sgte\_Accion\_Plan}(objy')))
\end{aligned}
\tag{5.23}
$$

$$
\begin{aligned}
R_{24}: \text{SI } & \text{Aplicar\_A\_Pos}(accx, objy) \wedge \text{Sgte\_Accion\_En\_Plan}(accx) \wedge \\
& \text{Pos\_En\_Sgte\_Accion\_Plan}(objy') \wedge \\
& \neg\text{Eq}(objy, objy')) \wedge \\
& \text{Sabe}(\text{Pos\_En\_Sgte\_Accion\_Plan}(objy')) \wedge \\
& \text{Eq}(\text{Subespacio}(objy), \text{Subespacio}(objy')) \rightarrow \\
& \text{Sabe}(\text{Subespacio}(objy')) \wedge \\
& \text{Es\_Capaz\_De\_Construir}( \\
& \quad \text{Ubicacion\_En\_Mapa\_Mental}( \\
& \qquad porigen, \text{Pos}(\text{Subespacio}(objy')))) \wedge \\
& \text{Es\_Capaz\_De\_Construir}( \\
& \quad \text{Camino}(porigen, \text{Pos}(\text{Subespacio}(objy')))) \wedge \\
& \text{Sabe}(\text{Donde\_Esta}(\text{Subespacio}(objy'))) \wedge \\
& \text{Sabe}(\text{Reconocer}(\text{Subespacio}(objy'))) \\
& \text{Sabe}(\text{Subespacio}(objy)) \wedge \\
& \neg\text{Sabe}(\text{Reconocer}(objy'))
\end{aligned}
\tag{5.24}
$$

Igualmente, se definen las siguientes tres reglas (expresadas conjuntamente aquí en una) para el caso en que ni siquiera coincidan los subespacios de las posiciones finales expre-

sadas éstas como *posx*, *objy* o *area* respectivamente:

$$
\begin{aligned}
R_{25}: \text{SI } & \text{Aplicar\_A\_Pos}(\text{accx}, \text{posx/objy/area}) \wedge \\
& \text{Sgte\_Accion\_En\_Plan}(\text{accx}) \wedge \\
& \text{Pos\_En\_Sgte\_Accion\_Plan}(\text{posx}'/\text{objy}'/\text{area}') \wedge \\
& \neg\text{Eq}(\text{posx/objy/area},\ \text{posx}'/\text{objy}'/\text{area}')) \wedge \\
& \neg\text{Eq}(\text{Subespacio}(\text{posx/objy/area}), \\
& \quad \text{Subespacio}(\text{posx}'/\text{objy}'/\text{area}')) \rightarrow \\
& \neg\text{Es\_Capaz\_De\_Construir}( \\
& \quad \text{Camino}(\text{porigen},\ \text{Pos}(\text{Subespacio}(\text{posx}'/\text{objy}'/\text{area}')))) \wedge \\
& (\neg\text{Sabe}(\text{Subespacio}(\text{posx}'/\text{objy}'/\text{area}')) \vee \\
& \neg\text{Es\_Capaz\_De\_Construir}( \\
& \quad \text{Ubicacion\_En\_Mapa\_Mental}( \\
& \quad \text{porigen},\ \text{Pos}(\text{Subespacio}(\text{posx}'/\text{objy}'/\text{area}'))))) \wedge
\end{aligned}
\tag{5.25}
$$

- Diagnóstico de acciones aplicadas fuera de secuencia.

  Diagnóstico de acciones ejecutadas por el estudiante que no es la que corresponde ejecutar pero están en el plan posteriormente y coincide al menos el operador de la acción:

$$
\begin{aligned}
R_{26}: \text{SI } & \text{Aplicar}(\text{accx}) \wedge \text{Sgte\_Accion\_En\_Plan}(\text{accy}) \wedge \\
& \neg\text{Eq}(\text{accx}, \text{accy}) \wedge \text{Operador}(\text{accy}, \text{opx}) \wedge \\
& \text{Op\_Sgte\_Accion\_Plan}(\text{opy}) \wedge \neg\text{Eq}(\text{opx}, \text{opy}) \wedge \\
& \text{Op\_En\_Plan\_Post}(\text{opy}) \rightarrow \\
& \text{Sabe}(\text{Act\_En\_Plan\_Post}(\text{accx})) \wedge \\
& \neg\text{Sabe}(\text{Sgte\_Accion\_En\_Plan}(\text{accy}))
\end{aligned}
\tag{5.26}
$$

  Diagnóstico de acciones ejecutadas por el estudiante que se aplican a uno o más objeto, no es la acción que corresponde ejecutar pero están en el plan posteriormente y coincide al menos uno de los objetos sobre el que se aplica. Por ejemplo, el estudiante pulsa el botón de comienzo antes de cerrar el cajón de los detergentes tras echar el detergente de lavado en la cubeta de detergentes. Para realizar este diagnóstico se añade la siguiente regla:

$$
\begin{aligned}
R_{27}: \text{SI } & \text{Aplicar\_A\_Objeto}(\text{accx},\ \text{objx}) \wedge \\
& \text{Sgte\_Accion\_En\_Plan}(\text{accy}) \wedge \neg\text{Eq}(\text{accx}, \text{accy}) \wedge \\
& \text{Operador}(\text{accx}, \text{opx}) \wedge \\
& \text{Op\_Obj\_En\_Plan\_Post}(\text{opx},\ \text{objx}) \rightarrow \\
& \text{Sabe}(\text{Requiere\_Objeto}(\text{accx},\ \text{objx})) \wedge \\
& \neg\text{Sabe}(\text{Sgte\_Accion\_En\_Plan}(\text{accy}))
\end{aligned}
\tag{5.27}
$$

Diagnóstico de acciones ejecutadas por el estudiante, no es la acción que corresponde ejecutar pero están en el plan posteriormente coincidiendo operador y argumentos.

$$\begin{aligned} R_{28}: \text{SI } & \text{Aplicar}(accx) \wedge \text{Sgte\_Accion\_En\_Plan}(accy) \wedge \\ & \neg\text{Eq}(accx, accy) \wedge \text{Acc\_En\_Plan\_Post}(accx) \rightarrow \\ & \text{Sabe}(\text{Acc\_En\_Plan\_Post}(accx)) \wedge \\ & \neg\text{Sabe}(\text{Sgte\_Accion\_En\_Plan}(accy)) \end{aligned} \tag{5.28}$$

Diagnóstico de acciones ejecutadas por el estudiante y relevantes pero que no están en el plan. Por ejemplo, cuando el estudiante tiene que pulsar ya el botón de comienzo y abre la puerta de la lavadora (si esto fuera posible):

$$\begin{aligned} R_{29}: \text{SI } & \text{Aplicar}(accx) \wedge \text{Sgte\_Accion\_En\_Plan}(accy) \wedge \\ & \neg\text{Eq}(accx, accy) \wedge \neg\text{Acc\_En\_Plan\_Post}(accx) \rightarrow \\ & \neg\text{Sabe}(\text{Sgte\_Accion\_En\_Plan}(accy)) \end{aligned} \tag{5.29}$$

**Diagnóstico de acciones erróneas dependientes del dominio**

Representan los errores típicos en el dominio.

- Si el estudiante aplica una acción que crea una tupla de una relación entre varios objetos, que implica exactamente el mismo operador y los mismos objetos que la siguiente acción en el plan, pero el orden de los objetos en la tupla es inadecuado, se puede asumir que él/ella no sabe que el orden de los objetos es relevante, y no sabe cuál es el orden correcto:

$$\begin{aligned} R_{30}: \text{SI } & \text{Aplicar}(accx) \wedge \\ & \text{Es\_De\_Tipo}(accx, \text{Modifies\_Relation\_Among\_Objects}) \wedge \\ & \text{Sgte\_Accion\_En\_Plan}(accx') \wedge \text{Operador}(accx, opx) \wedge \\ & \text{Operador}(accx', opx) \wedge \text{Eq}(\text{Args}(accx, lobjx), \text{Args}(accx', lobjx')) \wedge \\ & \neg\text{Eq}(\text{Orden\_Args}(lobjx), \text{Orden\_Args}(lobjx')) \rightarrow \\ & \neg\text{Sabe}(\text{Orden\_Args\_Es\_Relevante}(accx')) \wedge \\ & \neg\text{Sabe}(\text{Orden\_Args\_En\_Sgte\_Accion}(accx')) \wedge \end{aligned} \tag{5.30}$$

#### 5.3.2.2 Diagnóstico de las ejecuciones de la actividad

Esta categoría está relacionada con la evaluación de la sesión que se llevará a cabo una vez que el estudiante concluya una actividad dada. A su vez, dividimos sus reglas en tres subcategorías que se detallan a continuación: *Diagnóstico de la ejecución de la actividad*, *Diagnóstico de las trayectorias de la actividad*, y *Diagnóstico del nivel de maestría*.

**Diagnóstico de la ejecución de la actividad**

Las reglas asociadas permiten, basándose en los parámetros obtenidos a lo largo de la ejecución de la actividad, y una vez finalizada ésta, diagnosticar si la solución obtenida por el estudiante es de igual, menor o mayor coste que la solución óptima obtenida por el planificador. El escenario de solución de menor coste es posible porque el estudiante puede realizar un conjunto de acciones completamente correctas, sin ningún tipo de fallos, con la misma trayectoria que la trayectoria de la solución óptima pero que requiera menor número de acciones. En este caso, si realiza correctamente todas las acciones, sin necesidad de pistas, preguntas, etc., el estudiante habrá obtenido una solución con menor coste que la solución óptima. Así pues, las reglas concretas añadidas para este tipo de diagnóstico son:

- Diagnóstico de soluciones de igual coste que la solución planificada:

$$
\begin{aligned}
R_{31}: \text{SI } & \text{Calidad\_Ejecucion\_Actividad(estActividad, factorCalidadActuacion)} \wedge \\
& \text{Eq(factorCalidadActuacion, calidadMax)} \wedge \\
& \text{Plan\_Actividad\_Con\_Coste(planx, costx)} \wedge \\
& \text{Es\_De\_Actividad(estActividad, actividadx)} \wedge \\
& \text{Coste\_Sol\_Optima(actividadx, costy)} \wedge \text{Eq(costx, costy)} \rightarrow \\
& \text{Es\_Capaz\_De\_Construir(Plan\_Solucion(actividadx, Coste\_Optimo))}
\end{aligned}
\tag{5.31}
$$

  En la regla anterior, se tiene en cuenta en ´factorCalidadActuacion el parámetro *executionFactor*, no sólo el *trajectoryFactor*, puesto que el estudiante ha podido realizar la trayectoria óptima pero haber realizado más acciones, menos acciones, o igual número de acciones que en la solución óptima.

- Diagnóstico de soluciones de menor coste que la solución óptima:

$$
\begin{aligned}
R_{32}: \text{SI } & \text{Calidad\_Ejecucion\_Actividad(estActividad, factorCalidadActuacion)} \wedge \\
& \text{Eq(factorCalidadActuacion, calidadMax)} \wedge \\
& \text{Plan\_Actividad\_Con\_Coste(planx, costx)} \wedge \\
& \text{Es\_De\_Actividad(estActividad, actividadx)} \wedge \\
& \text{Coste\_Sol\_Optima(actividadx, costy)} \wedge \text{Menor(costx, costy)} \rightarrow \\
& \text{Es\_Capaz\_De\_Construir(} \\
& \quad \text{Plan\_Solucion(actividadx, Coste\_Mejor\_Planificada))}
\end{aligned}
\tag{5.32}
$$

- Diagnóstico de soluciones de mayor coste que la solución óptima:

$$
\begin{aligned}
R_{33}: \text{SI } & \text{Calidad\_Ejecucion\_Actividad(estActividad, factorCalidadActuacion)} \wedge \\
& \text{Eq(factorCalidadActuacion, Menor(calidadMax))} \rightarrow \\
& \text{Es\_Capaz\_De\_Construir(} \\
& \quad \text{Plan\_Solucion(actividadx, Coste\_Peor\_Planificada))}
\end{aligned}
\tag{5.33}
$$

**Diagnóstico del nivel de maestría**

Para diagnosticar el grado de maestría o dominio demostrado por el estudiante en la resolución del problema (construcción del plan), se han añadido las reglas siguientes. En ellas, se ha tenido en cuenta el valor obtenido para el parámetro *factorActivityKnowledge* de la actividad tras su realización. Este factor, como ya se describió en la ontología (A.1), está basado en la evaluación de las acciones en la ejecución actual de la actividad (valor del parámetro *executionFactor*), en la evaluación de la trayectoria seguida por el estudiante en esta ejecución (valor del parámetro *trajectoryFactor*), y en las evaluaciones ya obtenidas en anteriores ejecuciones de la misma actividad por el estudiante. Los valores de los límites que aparecen en las reglas son parámetros configurables, que dependerán del tipo de actividad y el entorno concreto de aprendizaje.

- Diagnóstico de nivel de maestría de principiante:

$$\begin{aligned} R_{34}: \text{SI } & \text{Calidad\_Actividad(estActividad, factorCalidadActiv)} \wedge \\ & \text{MayorEq(factorCalidadActiv, lim1)} \wedge \\ & \text{Menor(factorCalidadActiv, lim2)} \wedge \\ & \text{Es\_De\_Actividad(estActividad, actividadx)} \rightarrow \\ & \text{Es\_Capaz\_De\_Construir(} \\ & \quad \text{Plan\_Actividad\_Como\_Principiante(actividadx))} \end{aligned} \tag{5.34}$$

- Diagnóstico de nivel de maestría intermedio:

$$\begin{aligned} R_{35}: \text{SI } & \text{Calidad\_Actividad(estActividad, factorCalidadActiv)} \wedge \\ & \text{MayorEq(factorCalidadActiv, lim2)} \wedge \\ & \text{Menor(factorCalidadActiv, lim3)} \wedge \\ & \text{Es\_De\_Actividad(estActividad, actividadx)} \rightarrow \\ & \text{Es\_Capaz\_De\_Construir(} \\ & \quad \text{Plan\_Actividad\_Como\_Intermedio(actividadx))} \end{aligned} \tag{5.35}$$

- Diagnóstico de nivel de maestría avanzado:

$$\begin{aligned} R_{36}: \text{SI } & \text{Calidad\_Actividad(estActividad, factorCalidadActiv)} \wedge \\ & \text{MayorEq(factorCalidadActiv, lim3)} \wedge \\ & \text{Menor(factorCalidadActiv, lim4)} \wedge \\ & \text{Es\_De\_Actividad(estActividad, actividadx)} \rightarrow \\ & \text{Es\_Capaz\_De\_Construir(} \\ & \quad \text{Plan\_Actividad\_Como\_Avanzado(actividadx))} \end{aligned} \tag{5.36}$$

- Diagnóstico de nivel de maestría experto:

$$\begin{aligned} R_{37}: \text{SI } & \text{Calidad\_Actividad(estActividad, factorCalidadActiv)} \wedge \\ & \text{MayorEq(factorCalidadActiv, lim4)} \wedge \\ & \text{Es\_De\_Actividad(estActividad, actividadx)} \rightarrow \\ & \text{Es\_Capaz\_De\_Construir(} \\ & \quad \text{Plan\_Actividad\_Como\_Experto(actividadx))} \end{aligned} \tag{5.37}$$

**Diagnóstico de las trayectorias de la actividad**

Las reglas de esta subcategoría se han dividido, a su vez, en dos subcategorías: *diagnóstico del grado de calidad de la trayectoria* y *diagnóstico de acuerdo a las características de la trayectoria*.

- *Diagnóstico de la calidad de la trayectoria.*

  Para diagnosticar el grado de calidad en la elección del camino durante la ejecución actual de la actividad, se ha tenido en cuenta el valor del parámetro *trajectoryFactor*. En el cálculo de este factor intervienen diversos parámetros y la valoración de su resultado depende del escenario de la actividad en que ésta se ejecute; por ejemplo, del grado de peligrosidad que implique la ejecución de cada trayectoria en un escenario específico. Se establecen por ello unos límites (configurables) obtenidos mediante un conjunto de pruebas previas a las ejecuciones de cada actividad por el estudiante. Las siguientes reglas son añadidas para diagnosticar este aspecto:

  - Diagnóstico de trayectoria con grado de calidad óptimo:

$$\begin{aligned} R_{38}: \text{SI } & \text{Calidad\_Trayectoria(estActividadx, factorCalidadTray)} \wedge \\ & \text{Eq(factorCalidadTray, valorOptimo)} \wedge \\ & \text{Es\_De\_Actividad(estActividadx, actividadx)} \rightarrow \\ & \text{Sabe\_Elegir(Trayectoria\_Con\_Calidad(actividadx, nivel\_optimo))} \end{aligned} \tag{5.38}$$

  - Diagnóstico de trayectoria con grado de calidad muy bueno:

$$\begin{aligned} R_{39}: \text{SI } & \text{Calidad\_Trayectoria(estActividadx, factorCalidadTray)} \wedge \\ & \text{MayorEq(factorCalidadTray, lim1\_trayect)} \wedge \\ & \text{Menor(factorCalidadTray, valorOptimo)} \wedge \\ & \text{Es\_De\_Actividad(estActividadx, actividadx)} \rightarrow \\ & \text{Sabe\_Elegir(Trayectoria\_Con\_Calidad(actividadx, nivel\_muy\_bueno))} \end{aligned} \tag{5.39}$$

- Diagnóstico de trayectoria con grado de calidad bueno:

$$\begin{aligned} R_{40}: \text{SI } & \text{Calidad\_Trayectoria(estActividadx, factorCalidadTray)} \wedge \\ & \text{MayorEq(factorCalidadTray, lim2\_trayect)} \wedge \\ & \text{Menor(factorCalidadTray, lim1\_trayect)} \wedge \\ & \text{Es\_De\_Actividad(estActividadx, actividadx)} \rightarrow \\ & \text{Sabe\_Elegir(Trayectoria\_Con\_Calidad(actividadx, nivel\_bueno))} \end{aligned} \tag{5.40}$$

- Diagnóstico de trayectoria con grado de calidad aceptable:

$$\begin{aligned} R_{41}: \text{SI } & \text{Calidad\_Trayectoria(estActividadx, factorCalidadTray)} \wedge \\ & \text{MayorEq(factorCalidadTray, lim3\_trayect)} \wedge \\ & \text{Menor(factorCalidadTray, lim2\_trayect)} \wedge \\ & \text{Es\_De\_Actividad(estActividadx, actividadx)} \rightarrow \\ & \text{Sabe\_Elegir(Trayectoria\_Con\_Calidad(actividadx, nivel\_regular))} \end{aligned} \tag{5.41}$$

- Diagnóstico de trayectoria con grado de calidad pobre:

$$\begin{aligned} R_{42}: \text{SI } & \text{Calidad\_Trayectoria(estActividadx, factorCalidadTray)} \wedge \\ & \text{MayorEq(factorCalidadTray, lim4\_trayect)} \wedge \\ & \text{Menor(factorCalidadTray, lim3\_trayect)} \wedge \\ & \text{Es\_De\_Actividad(estActividadx, actividadx)} \rightarrow \\ & \text{Sabe\_Elegir(Trayectoria\_Con\_Calidad(actividadx, nivel\_malo))} \end{aligned} \tag{5.42}$$

- Diagnóstico de trayectoria con grado de calidad pobre:

$$\begin{aligned} R_{43}: \text{SI } & \text{Calidad\_Trayectoria(estActividadx, factorCalidadTray)} \wedge \\ & \text{Menor(factorCalidadTray, lim4\_trayect)} \wedge \\ & \text{Es\_De\_Actividad(estActividadx, actividadx)} \rightarrow \\ & \text{Sabe\_Elegir(Trayectoria\_Con\_Calidad(} \\ & \quad \text{actividadx, nivel\_pesimo))} \end{aligned} \tag{5.43}$$

- *Diagnóstico de las características de la trayectoria.* Se trata de diagnosticar acerca del estado de conocimiento del estudiante (como, por ejemplo, su posible desorientación) a partir de ciertas características de la trayectoria tales como: si el estudiante se aleja o se aproxima de/a la trayectoria óptima, si el estudiante sigue justo el sentido opuesto a la trayectoria óptima, si está realizando una trayectoria con ciclos, si está desorientado en función de los tramos que va realizando, etc. Este diagnóstico podría realizarse de dos formas: a posteriori, una vez que el estudiante ha realizado la actividad completa y, por lo tanto, ha llegado a su destino, o durante el desplazamiento. En ambos casos, el diagnóstico podría ayudar en

gran medida al Módulo de Tutoría para guiar al estudiante en su aprendizaje. Dos posibles reglas en un diagnóstico de este tipo podrían ser las siguientes:

$$\begin{aligned} R_{44}: \text{SI } & \text{Mayor(Desviacion(} \\ & \quad \text{trayectoria\_estudiante, trayectoria\_optima, grado))} \wedge \\ & \text{Aplicar\_Plan(planx)} \wedge \text{Accion\_Final\_Plan(planx, apl\_accx)} \\ & \text{Accion\_Puntual(apl\_accx, accx)} \wedge \\ & \text{Sabe(apl\_accx, Requiere\_Precond(} \\ & \quad \text{accx, Pos\_En\_Sgte\_Accion\_Plan(pdest)))} \rightarrow \\ & \neg\text{Sabe(Mantener\_Ubicacion\_En\_Mapa\_Mental(pdest))} \end{aligned} \tag{5.44}$$

$$\begin{aligned} R_{45}: \text{SI } & \text{Mayor(Desviacion(} \\ & \quad \text{trayectoria\_estudiante, trayectoria\_optima, grado))} \wedge \\ & \text{Aplicar\_Plan(planx)} \wedge \text{Accion\_Final\_Plan(planx, apl\_accx)} \\ & \text{Accion\_Puntual(apl\_accx, accx)} \wedge \\ & \neg\text{Sabe(Requiere\_Precond(} \\ & \quad \text{accx, Pos\_En\_Sgte\_Accion\_Plan(pdest)))} \rightarrow \\ & \neg\text{Sabe(Mantener\_Ubicacion\_En\_Mapa\_Mental(pdest))} \vee \\ & \neg\text{Sabe(apl\_accx, Requiere\_Precond(} \\ & \quad \text{accx, Pos\_En\_Sgte\_Accion\_Plan(pdest)))} \end{aligned} \tag{5.45}$$

#### 5.3.2.3 Diagnóstico basado en el número y tipo de preguntas formuladas por el estudiante

El tipo de preguntas formuladas por el estudiante, proporciona también información sobre el grado de conocimiento que el estudiante acerca de los objetos del escenario, de las acciones posibles, o de la actividad.

- *Diagnóstico de grado de conocimiento de un objeto.*

  Un estudiante puede preguntar acerca de algún aspecto del aprendizaje de determinadas formas. Por ejemplo, puede preguntar para qué sirve o para qué se usa un determinado objeto, y el objetivo de esa pregunta sería el mismo.

$$\begin{aligned} R_{46}: \text{SI } & \text{Tipo\_Pregunta(pregunta, Que\_es, objx)} \rightarrow \\ & \neg\text{Sabe(Que\_es(objx))} \end{aligned} \tag{5.46}$$

$$
\begin{aligned}
R_{47}: \text{SI } & \text{Tipo\_Pregunta(pregunta, Que\_es, objx)} \wedge \\
& \text{Sgte\_Accion\_En\_Plan(accx)} \wedge \\
& \text{Requiere\_Precond(accx, Objetos\_De\_Clases(lclasey))} \wedge \\
& \neg\text{Instancia\_De(objx, lclasey))} \rightarrow \\
& \neg\text{Sabe(Reconocer(Objetos\_De\_Clases(lclasey)))} \vee \\
& \neg\text{Sabe(Sgte\_Accion\_En\_Plan(accx))} \vee \\
& \neg\text{Sabe(Requiere\_Precond(} \\
& \qquad \text{accx, Objetos\_De\_Clases(lclasey)))}
\end{aligned}
\tag{5.47}
$$

$$
\begin{aligned}
R_{48}: \text{SI } & \text{Tipo\_Pregunta(pregunta, Para\_que\_sirve, objx)} \rightarrow \\
& \neg\text{Sabe(Para\_Que\_Sirve(objx))}
\end{aligned}
\tag{5.48}
$$

- *Diagnóstico de desconocimiento parcial de la actividad que el estudiante está realizando.* El estudiante ha realizado una o más acciones fallidas, que no forman parte de la actividad, y desconoce por qué. Sería conveniente ir al último punto (en la traza) en el que realizó una acción acorde a la planificación de la actividad para explicarle dónde empezó a desviarse del plan y qué debería haber hecho. Al igual que antes, preguntas como *¿Qué es lo siguiente que tengo que hacer?*, *¿Tengo que manipular objetos?*, *¿Tengo que transportar objetos?*, *¿Tengo que moverme?*, etc., podrían reducirse al tipo de pregunta: `Que_es_lo_siguiente_a_hacer`, tal y como se muestra en las siguientes reglas:

$$
\begin{aligned}
R_{49}: \text{SI } & \text{Tipo\_Pregunta(pregunta, Que\_es\_lo\_siguiente\_a\_hacer)} \wedge \\
& \text{Sgte\_Accion\_En\_Plan(accx)} \rightarrow \\
& \neg\text{Sabe(Sgte\_Accion\_En\_Plan(accx))} \wedge \\
& \text{(Estudiante\_Precavido)}
\end{aligned}
\tag{5.49}
$$

$$
\begin{aligned}
R_{50}: \text{SI } & \text{Tipo\_Pregunta(pregunta, Que\_deberia\_haber\_hecho)} \wedge \\
& \text{Sgte\_Accion\_En\_Plan(accx)} \rightarrow \\
& \neg\text{Sabe(Sgte\_Accion\_En\_Plan(accx))} \wedge \\
& \text{(Estudiante\_Aventurado)}
\end{aligned}
\tag{5.50}
$$

- *Diagnóstico de desconocimiento parcial del operador asociado a la acción siguiente a ejecutar por el estudiante*

$$
\begin{aligned}
R_{51}: \text{SI } & \text{Tipo\_Pregunta(pregunta, Cuales\_son\_los\_objetos)} \wedge \\
& \text{Sgte\_Accion\_En\_Plan(accx)} \wedge \\
& \text{Requiere\_Precond(accx, Objetos\_De\_Clases(lclasey))} \rightarrow \\
& \neg\text{Sabe(Requiere\_Precond(accx, Objetos\_De\_Clases(lclasey)))} \wedge \\
& \text{(Estudiante\_Precavido)}
\end{aligned}
\tag{5.51}
$$

$R_{52}$ : SI Tipo_Pregunta(pregunta, Donde_estoy) ∧
Esta_Situado_En(posx) →
¬Sabe(Ubicacion_En_Mapa_Mental(posx)) ∧
(Estudiante_Precavido) (5.52)

$R_{53}$ : SI Tipo_Pregunta(pregunta, A_que_lugar) ∧
Sgte_Accion_En_Plan(accx) ∧
Pos_En_Sgte_Accion_Plan(posx) → (5.53)
¬Sabe(Requiere_Precond(accx, posx)) ∧
(Estudiante_Precavido)

$R_{54}$ : SI Tipo_Pregunta(pregunta, Por_donde) ∧
Esta_Situado_En(posx) ∧
Pos_En_Sgte_Accion_Plan(pfinal) → (5.54)
¬Es_Capaz_De_Construir(Camino(posx, pfinal)) ∧
(Estudiante_Precavido)

$R_{55}$ : SI Tipo_Pregunta(pregunta, Cual_es_el_efecto, opx) →
¬Sabe(Proporciona_Postcond(opx, postcondx)) ∧ (5.55)
(Estudiante_Precavido)

$R_{56}$ : SI Esta_Objeto(pregunta, objx) ∧
Sgte_Accion_En_Plan(accx) ∧
Requiere_Precond(accx, Obj_Sgte_Accion_Plan(objx)) → (5.56)
Sabe(Requiere_Precond(accx, objx))

$R_{57}$ : SI Esta_Operador(pregunta, opx) ∧
Sgte_Accion_En_Plan(accx) ∧
Op_Sgte_Accion_Plan(opx)) → (5.57)
Sabe(Op_Sgte_Accion_Plan(opx))

$R_{58}$ : SI Esta_Objeto(pregunta, objx) ∧
Sgte_Accion_En_Plan(accx) ∧
Obj_Sgte_Accion_Plan(objx)) → (5.58)
Sabe(Obj_Sgte_Accion_En_Plan(objx))

#### 5.3.2.4 Diagnóstico basado en las pistas e instrucciones proporcionadas al estudiante

Las pistas e instrucciones proporcionadas por el tutor permiten obtener información acerca del grado de conocimiento que el estudiante debería tener de los objetos en el escenario, de las posibles acciones, o sobre la actividad específica de aprendizaje. Esta categoría se ha clasificado, a su vez, en las subcategorías que se describen a continuación:

- *Diagnóstico del conocimiento de la acción que debe realizar el estudiante a continuación.* Este diagnóstico se podrá obtener cuando la pista (en cualquier nivel de pista ofrecido), o instrucción proporcionada al estudiante para realizar la siguiente acción, contienen información respecto a la acción a realizar: muévete, trasládate, coge, inserta, pulsa, etc. Se usa un este predicado `Da_Pista` aplicado a la siguiente acción a realizar en el plan.

$$R_{59}: \text{SI } \text{Da\_Pista}(\text{Sgte\_Accion\_En\_Plan}(\text{accx})) \rightarrow \text{Sabe}(\text{Sgte\_Accion\_En\_Plan}(\text{accx})) \tag{5.59}$$

$$R_{60}: \text{SI } \text{Da\_Pista}(\text{Sgte\_Accion\_En\_Plan}(\text{accx})) \wedge \text{Da\_Pista}(\text{Op\_Sgte\_Accion\_Plan}(\text{opx})) \rightarrow \text{Sabe}(\text{Op\_Sgte\_Accion\_Plan}(\text{opx})) \tag{5.60}$$

$$R_{61}: \text{SI } \text{Da\_Pista}(\text{Sgte\_Accion\_En\_Plan}(\text{accx})) \wedge \text{Da\_Pista}(\text{Obj\_Sgte\_Accion\_Plan}(\text{objx})) \rightarrow \text{Sabe}(\text{Obj\_Sgte\_Accion\_Plan}(\text{objx})) \tag{5.61}$$

- *Diagnóstico del conocimiento del destino si la acción siguiente es de tipo movimiento (acción que modifique la posición del estudiante.* Este diagnóstico se podrá obtener cuando la pista o instrucción proporcionada al estudiante para realizar la siguiente acción contiene información sobre la posición de destino a alcanzar. La regla R61, que a continuación se muestra, obtiene este diagnóstico ampliando el conjunto de predicados para las precondiciones de este tipo de acciones con el siguiente: `Pos_En_Sgte_Accion_Plan(posx)`[4].

  Además, si la posición de destino es relativa a un objeto o relativa a un área del escenario, por ejemplo en pistas tales como *Acércate a la lavadora* o *Acércate a la cocina*, también se obtiene información complementaria acerca de estas precondiciones mediante los predicados `Obj_Pos_En_Sgte_Accion_Plan(objy)` y `Area_Pos_En_Sgte_Accion_Plan(area)` respectivamente.

$$R_{62}: \text{SI } \text{Da\_Pista}(\text{Sgte\_Accion\_En\_Plan}(\text{accx})) \wedge \text{Da\_Pista}(\text{Requiere\_Precond}(\text{accx}, \text{precondy}) \rightarrow \text{Sabe}(\text{Requiere\_Precond}(\text{accx}, \text{precondy})) \tag{5.62}$$

  donde `precondy` en este caso es `Pos_En_Sgte_Accion_Plan(posx)`.

[4] En el diagnóstico descrito no se considera más que un único objeto de cada tipo en el destino. En el caso de que existieran varios objetos de una cierta clase de objetos habría que ampliar las reglas en esta subcategoría para distinguir en la pista/instrucción a qué objeto se refiere (.el", ün", etc.), y diagnosticar acorde a esta información adicional.

- *Diagnóstico del conocimiento de objetos si la siguiente acción es de interacción con objeto(s)*. Este diagnóstico se podrá obtener cuando la pista o instrucción proporcionada al estudiante para realizar la siguiente acción contiene información acerca del tipo de objeto u objetos con los que a continuación el estudiante debe interaccionar (por ejemplo, si la pista o instrucción se refiere a una acción de modificación de la relación con un objeto o de modificación del estado del objeto). La regla R61 previamente descrita obtendría también este diagnóstico ampliando el conjunto de predicados para las precondiciones de este tipo de acciones con el siguiente: `Clase_Obj_Sgte_Accion_Plan(clasex)`.
- *Diagnóstico del conocimiento del lugar donde se encuentra un objeto cuando la siguiente acción modifica la posición de un objeto*. Este diagnóstico se podrá obtener cuando la pista o instrucción proporcionada al estudiante para realizar la siguiente acción contiene información acerca de la posición en la que está el objeto con el que interacciona el estudiante. La regla R61 previamente descrita obtendría también este diagnóstico ampliando el conjunto de predicados para las precondiciones de este tipo de acciones con el siguiente predicado: `Pos_Obj_Sgte_Accion_Plan(posx)`. Además, si la posición del objeto en la siguiente acción es relativa a un objeto o relativa a un área del escenario, por ejemplo en pistas tales como *Coge el detergente de la mesa de lavado* o *Coge el detergente de la cocina*, también se obtiene información complementaria acerca de estas precondiciones mediante el predicado `Pos_Relat_Obj_Sgte_Accion_Plan(objx)`.
- *Diagnóstico basado en pistas acerca de las metas a conseguir*. Por ejemplo, una pista de este tipo es la siguiente: *Necesitas conseguir un martillo*. Si se proporciona esta pista, se puede diagnosticar que el estudiante sabe que necesita construir un plan cuyo último operador tiene una post-condición que permite conseguir esa meta. A continuación, se añaden las siguientes reglas de diagnóstico que lo expresan:

$$\begin{aligned} R_{63}: \text{SI } & \text{Da\_Pista(Meta(postcondx))} \rightarrow \\ & \text{Sabe(Requiere\_Plan\_Que\_Consiga(Meta(postcondx)))} \end{aligned} \tag{5.63}$$

$$\begin{aligned} R_{64}: \text{SI } & \text{Sabe(Requiere\_Plan\_Que\_Consiga(Meta(postcondx))} \wedge \\ & \text{Aplicar\_Plan(planx)} \wedge \text{Consigue\_Estado\_Final(planx, estadofinalx)} \wedge \\ & \text{Incluye(estadofinalx, Meta(postcondx))} \rightarrow \\ & \text{Es\_Capaz\_De\_Construir(planx, Meta(postcondx))} \end{aligned} \tag{5.64}$$

#### 5.3.2.5 Diagnóstico basado en comportamiento no verbal

Esta categoría incluye reglas que permiten inferir los objetivos de aprendizaje que se pueden asumir (o no) alcanzados por el estudiante basado en el tiempo de inactividad del estudiante, la dirección de su mirada, etc.

- *Diagnóstico basado en tiempos de ejecución y orientación de la mirada*

$$\begin{aligned} R_{65}: \text{SI } & \text{Sgte\_Accion\_En\_Plan(accx)} \wedge \\ & \text{Objeto\_Usado\_En\_Accion(accx, objx)} \wedge \\ & \text{Tpo\_Sin\_Mirar(objx, tpoy)} \wedge \text{Mayor(tpoy, UmbralTiempo)} \rightarrow \\ & \neg\text{Sabe(Donde\_Esta(objx))} \end{aligned} \tag{5.65}$$

$$
\begin{aligned}
R_{66}: \text{SI } & \text{Sgte\_Accion\_En\_Plan(accx)} \wedge \\
& \text{Requiere\_Precond(accx, Pos\_En\_Sgte\_Accion\_Plan(posx))} \wedge \\
& \text{Esta\_Situado\_En(posx)} \wedge (( \text{Tiempo\_Inactividad(tinac)} \wedge \\
& \text{Mayor(tinac, TIME\_OUT))} \vee (\text{Se\_Aleja\_De(posx))}) \rightarrow \\
& \neg\text{Sabe(Sgte\_Accion\_Plan(accx))} \wedge \\
& \text{(Estudiante\_Pasivo)}
\end{aligned}
\tag{5.66}
$$

El Módulo de Resolución de Conflictos será el responsable de resolver las contradicciones así como de decidir si la contradicción se ha debido a que el estudiante realmente no sabe los objetivos del consecuente de la regla o bien ha sido un despi ste o descuido, se ha desorientado o se le ha olvidado lo que tenía que hacer a continuación en el plan (basándose, entre otras, en la información sobre la traza del estudiante).

#### 5.3.2.6 Predicados de las reglas de diagnóstico

A continuación se muestran las tablas de predicados implicados en las reglas del módulo de diagnóstico pedagógico. Los predicados están clasificados de acuerdo a la(s) jerarquía(s) a la que pertenecen los conceptos con los que se mapean sus términos. Estos conceptos pueden pertenecer a las ontologías: ontología del ME, ontología del Mundo, y/o Ontología de Tutoría.

Tabla 5.7: Predicados relacionados con las jerarquías de *Learning_Objective* y *Knowledge_Object*

| PREDICADOS | SIGNIFICADO |
|---|---|
| `Sabe(objx)` | El estudiante sabe el objeto de conocimiento *objx*. |
| `Para_Que_Sirve(objx)` | Para qué sirve el objeto *objx*. |
| `Que_Es(objy)` | Conoce el concepto *objy*. |
| `Reconocer(objy)` | Reconocer el objeto *objy*. |
| `Donde_Esta(objy)` | Dónde está el objeto *objy* (puede ser un objeto, un subespacio, una posición, etc.). |
| `Orden_Args_En_Sgte_Accion(accx)` | Orden de argumentos de la siguiente acción *accy*. |
| `Requiere_Plan_Que_Consiga(metax)` | El estudiante sabe que requiere un plan para alcanzar la meta *metax*. |

Tabla 5.7: Predicados relacionados con las jerarquías de *Learning_Objective* y *Knowledge_Object* (cont.)

| PREDICADOS | SIGNIFICADO |
|---|---|
| `Es_Capaz_De_Aplicar(accx)` | El estudiante es capaz de aplicar la acción *accx*. |
| `Es_Capaz_De_Aplicar(accx, calidady)` | El estudiante es capaz de aplicar el operador *accx* con *calidady*. |
| `Sabe_Elegir(objx)` | El estudiante sabe elegir el objeto de conocimiento *objx*. |
| `Es_Capaz_De_Obtener(objx)` | El estudiante es capaz de obtener el objeto de conocimiento *objx*. |
| `Es_Capaz_De_Obtener(accx, metay)` | El estudiante es capaz de obtener la meta *metay* al ejecutar la acción *accx*. |
| `Es_Capaz_De_Construir(objx)` | El estudiante es capaz de construir el objeto de conocimiento *objx*. |

Tabla 5.8: Predicados relacionados con las jerarquías de *Condition_On_State, Performance_State* y *Knowledge_Object*

| PREDICADOS | SIGNIFICADO |
|---|---|
| `Esta_En_Plan(objx)` | El objeto *objx* está implicado en el *plan*. |
| `Op_Sgte_Accion_Plan(opx)` | La siguiente acción en el plan tiene como operador *opx*. |
| `Obj_Sgte_Accion_Plan(objx)` | La siguiente acción en el plan se aplica al objeto *objx*. |
| `Clase_Obj_Sgte_Accion_Plan(clasex)` | La siguiente acción en el plan se aplica a un objeto de la clase de objetos *clasex*. |

Tabla 5.8: Predicados relacionados con las jerarquías de *Condition_On_State*, *Performance_State* y *Knowledge_Object* (cont.)

| PREDICADOS | SIGNIFICADO |
|---|---|
| `Pos_En_Sgte_Accion_Plan(posx)` | La siguiente acción en el plan tiene como precondición la posición final *posx* (posición absoluta, relativa a la posición de un objeto, o relativa a la posición de un subescenario. |
| `Area_En_Sgte_Accion_Plan(area)` | La siguiente acción tiene como precondición la posición de destino *area*. |
| `Obj_Pos_En_Sgte_Accion_Plan(objx)` | La siguiente acción de movimiento tiene como precondición el objeto *objx* destino al que desplazarse. |
| `Pos_Obj_Sgte_Accion_Plan(posx)` | La siguiente acción es de modificación de la relación entre el estudiante y un objeto (coger, por ejemplo) y una de sus precondiciones es la posición donde se encuentra el objeto de la acción. |
| `Pos_Relat_Obj_Sgte_Accion_Plan(posx)` | La siguiente acción es de modificación de la relación entre el estudiante y un objeto, y una de sus precondiciones es la posición (otro objeto) donde se encuentra el objeto al que se va a aplicar la acción. |
| `Plan_Actividad_Con_Coste(planx, costx)` | El coste del plan, *planx*, construido por el estudiante al realizar una actividad es *costex*. |
| `Sgte_Accion_En_Plan(accx)` | La siguiente acción en el plan es *accx*. |

Tabla 5.9: Predicados relacionados con las jerarquías de *Student_Trace_Related*, *Student_State*, y *Knowledge_Object*

| PREDICADOS | SIGNIFICADO |
|---|---|

Tabla 5.9: Predicados relacionados con la jerarquía de *Student_Trace_Related* (cont.)

| PREDICADOS | SIGNIFICADO |
|---|---|
| `Intenta_Aplicar(accx)` | El estudiante intentó aplicar la acción *accx*. |
| `Intenta_Aplicar_Sobre_Objeto(accx, objy)` | El estudiante intentó aplicar la acción *accx* sobre el objeto *objy*. |
| `Aplicar(accx)` | El estudiante aplicó la acción *accx*. |
| `Aplicar_A_Objeto(accx, objy)` | El estudiante aplicó la acción *accx* sobre el objeto *objy*. |
| `Aplicar_A_Pos(accx, pfinal)` | El estudiante aplicó la acción *accx* que le llevó a una posición *pfinal*. |
| `Da_Pista(pistax)` | El tutor da la pista o instrucción *pistax* al estudiante para realizar la actividad. |
| `Aplicar_Plan(planx)` | El estudiante aplicó el plan *planx*. |

Tabla 5.10: Predicados relacionados con la jerarquía de *Performance_State*

| PREDICADOS | SIGNIFICADO |
|---|---|
| `Calidad_Ejecucion_Actividad(estActivx, factorCalidadActuacionx)` | La calidad de la ejecución de la actividad representada por *estActivx* viene dada por *factorCalidadEjecucionx*. |
| `Calidad_Trayectoria(estActividadx, factorCalidadTrayecx)` | La calidad de la trayectoria en la ejecución de la actividad *estActividadx* es *factorCalidadTrayecx*. |

Tabla 5.10: Predicados relacionados con la jerarquía de *Performance_State* (cont.)

| PREDICADOS | SIGNIFICADO |
|---|---|
| `Calidad_Actividad(estActividadx, factorCalidadActivy)` | La calidad del conocimiento de la actividad *estActividadx* que posee el estudiante es *factorCalidadActivy*. |
| `Puede_Obtener_Meta_Con_Calidad(accx, metay, calidadz)` | El estudiante puede ejecutar la acción *accx* y obtener la meta *metay* con calidad *calidaz*. |
| `Incluye(estadox, metay)` | El *estadox* en la actuación del estudiante durante su aprendizaje incluye la meta *metay*. |
| `Consigue_Estado_Final(planx, estadoy)` | El estudiante consigue alcanzar el estado final *estadoy* asociado al plan *planx*. |

Tabla 5.11: Predicados relacionados con la jerarquía de *World State Ontology*

| PREDICADOS | SIGNIFICADO |
|---|---|
| `No_Cumple(accx, precondy)` | La acción *accx* no cumple su precondición *precondy*. |
| `Calidad_Optima_Meta(accx, metay, calidadz)` | La calidad óptima de la meta *metay* asociada a la acción *accx* es *calidadz*. |
| `Calidad_Pesima_Meta(accx, metay, calidadz)` | La calidad pésima de la meta *metay* asociada a la acción *accx* es *calidadz*. |
| `Esta_Situado_En(posx)` | El estudiante está situado en la posición *posx*. |
| `Esta_Situado_En(objx, posx)` | El objeto *objx* está en la posición *posx*. |

Tabla 5.11: Predicados relacionados con la jerarquía de *World State Ontology* (cont.)

| PREDICADOS | SIGNIFICADO |
|---|---|
| `Desviacion(trazTrayectx, trayect_opt, gradox)` | La trayectoria del estudiante, *trazTrayectx*, se desvía de la trayectoria óptima un cierto grado *gradox*. |
| `Tpo_Sin_Mirar(objx, tpox)` | El tiempo sin mirar el objeto *objx* es *tpox*. |
| `Tpo_Inactividad(tpox)` | El tiempo sin realizar una acción es *tpox*. |
| `Pos_Relativa_Objetos_De_Clase(clasex, posy)` | La posición relativa de los objetos de la clase *clasex* es *posy*. |
| `Se_Aleja_De(posx)` | El estudiante se aleja de la posición *posx*. |

Tabla 5.12: Predicados relacionados con la jerarquía de *Tutoring Ontology*

| PREDICADOS | SIGNIFICADO |
|---|---|
| `Coste_Sol_Optima(actividadx, costx)` | El coste del plan construido para la *actividadx* en la solución óptima es *coste*. |
| `Plan_Solucion(actividadx, costex)` | La valoración del coste del plan aplicado por el estudiante para la actividad *actividadx*, al realizar una cierta ejecución de actividad es *costex*. |
| `Plan_Actividad_Como_Principiante (actividadx)` | El estudiante construye un plan de nivel principiante para la actividad *actividadx* |
| `Plan_Actividad_Como_Intermedio(actividadx)` | El estudiante construye un plan de nivel intermedio para la actividad *actividadx* |
| `Plan_Actividad_Como_Avanzado(actividadx)` | El estudiante construye un plan de nivel avanzado para la actividad *actividadx* |

Tabla 5.12: Predicados relacionados con la jerarquía de *Tutoring Ontology* (cont.)

| PREDICADOS | SIGNIFICADO |
|---|---|
| `Plan_Actividad_Como_Experto(actividadx)` | El estudiante construye un plan de nivel experto para la actividad *actividadx* |
| `Trayectoria_Con_Calidad(actividadx, calidad)` | La calidad de la trayectoria tras la ejecución actual de la actividad *actividadx* es *calidad*. |

Tabla 5.13: Predicados relacionados con las jerarquías de *Performance_State* y de *Tutoring Ontology*

| PREDICADOS | SIGNIFICADO |
|---|---|
| `Es_De_Actividad(estActividadx, actividadx)` | El estado de ejecución *estActividadx* pertenece a la actividad *actividadx*. |

Tabla 5.14: Predicados relacionados con la jerarquía de *Knowledge_Object*

| PREDICADOS | SIGNIFICADO |
|---|---|
| `Usa_Objeto_Como_Herramienta(accx, objy)` | La acción *accx* usa el objeto *objy* como herramienta. |
| `Objeto_Usado_En_Accion(accx, objy)` | En la acción *accx* se usa el objeto *objy* (como herramienta o con otros fines). |
| `Modifica_Relacion_Entre_Objetos(accx, objx, objy)` | La acción *accx* modifica la relación entre dos objetos: *objx* y *objy*. |
| `Modifica_Relacion_Estudiante_Objeto(opx, objx)` | El operador *opx* modifica la relación entre el estudiante y el objeto *objx* |

Tabla 5.14: Predicados relacionados con la jerarquía de *Knowledge_Object* (cont.)

| PREDICADOS | SIGNIFICADO |
|---|---|
| `Modifica_Posicion(opx, pfinal)` | El operador *opx* aplicado modifica la posición del estudiante y el destino de la acción es la posición *pfinal* (*posx*/objy/area). |
| `Requiere_Objeto(accx, objy)` | La acción *accx* requiere un objeto *objy* para su aplicación. |
| `Esta_Objeto(pregunta, objx)` | La pregunta *pregunta* incluye el objeto *objx*. |
| `Esta_Accion(pregunta, accx)` | La pregunta *pregunta* incluye la acción *accx*. |
| `Tipo_Pregunta(preguntax, tipox)` | La pregunta *preguntax* es del tipo *tipox*. |
| `Tipo_Pregunta(preguntax, tipox, objx)` | La pregunta *preguntax* es del tipo *tipox* y con el objeto *objx*. |
| `Orden_Args_Es_Relevante(accx)` | El orden de los argumentos asociados a la acción *accx* es relevante para su correcta ejecución. |
| `Parte_De(objx, objy)` | El objeto *objx* es parte del objeto *objy*. |
| `Es_De_Tipo(objx, clasey)` | El objeto *objx* es del tipo de objetos *clasey*. |
| `Existe(objx)` | Existe el objeto *objx*. |
| `Objetos_De_Clase(clasex)` | Objetos de la clase *clasex*. |
| `Objetos_De_Clases(lclasesx)` | Objetos de clases pertenecientes a la lista *lclasesx*. |

Tabla 5.14: Predicados relacionados con la jerarquía de *Knowledge_Object* (cont.)

| PREDICADOS | SIGNIFICADO |
|---|---|
| `Subclase_De(objx, objy)` | La clase del objeto *objx* es subclase de *objy*. |
| `Usada_Para(accx, metay)` | La acción *accx* se utiliza para alcanzar la meta *metay*. |
| `Instancia_De(objx, clasey)` | El objeto *objx* es instancia de *clasey*. |
| `Instancia_De(objx, lclasey)` | El objeto *objx* es instancia de alguna de las clase de objetos de la lista *lclasey*. |
| `Se_Ejecuta(accx, modoEjecuciony)` | El modo de ejecución del operador asociado a *accx* es *modoEjecuciony*. |
| `Camino(porigen, pfinal)` | Camino entre la posición *porigen* y la posición *pfinal*. |
| `Es_Aplicable(accx, objy)` | El operador *accx* se puede aplicar al objy (el tipo del objeto es el adecuado para poder realizar la acción correspondiente sobre él). |
| `Son_Relacionables(objx,objy)` | Son relacionables los objetos *objx* y *objy*. |

Tabla 5.15: Predicados relacionados con la jerarquía de *Procedural_Knowledge*

| PREDICADOS | PREDICADOS |
|---|---|
| `Requiere_Precond(accx, precondy)` | El operador de la acción *accx* requiere para su aplicación el cumplimiento de la precondición *precondy*. |
| `Operador(accx, opy)` | El operador de la acción *accx* es *opy*. |

Tabla 5.15: Predicados relacionados con la jerarquía de *Procedural_Knowledge* (cont.)

| PREDICADOS | PREDICADOS |
|---|---|
| `Args(accx, lobjx)` | Lista de lo objetos a los que se aplica la acción *accx*. |
| `Accion_Final_Plan(planx, apl_accx)` | La última acción del plan planx es *apl_accx*, que especifica su posición relativa en el plan y su tiempo de inicio y fin. |
| `Accion_Puntual(apl_accx, accx)` | La acción asociada a un elemento de un plan de tipo aplicación de acción, *apl_accx*, es *accx*. |
| `Orden_Args(lobjx)` | Orden de los argumentos de la lista *lobjx*. |
| `Accion_Cumplimiento_Parcial(accx)` | La ejecución de la acción *accx* puede suponer un grado de cumplimiento de sus metas parcial. |
| `Meta(postcondx)` | La meta es *postcondx*. |
| `Proporciona_Postcond(accx, postcondy)` | El operador de la acción *accx* proporciona tras su aplicación la post-condición *postcondy*. |
| `Obtener_Meta(accx, metay)` | Se obtiene la meta *metay* tras ejecutar la acción *accx*. |
| `Puede_Obtener_Meta(accx, metay)` | Si se cumplen las precondiciones asociadas a la acción *accx*, se puede obtener la meta *metay* tras ejecutar el operador. |
| `Acc_En_Plan_Post(accx)` | La acción *accx* está en el plan posteriormente. |
| `Op_En_Plan_Post(opx)` | El operador *opx* está en el plan posteriormente. |
| `Op_Obj_En_Plan_Post(opx, objx)` | Si el operador *opx* está o no en el plan posteriormente y si se aplica a *objx*. |

Tabla 5.16: Predicados relacionados con la jerarquía de *Structural_Knowledge*

| PREDICADOS | PREDICADOS |
|---|---|
| `Subespacio(objx)` | Subespacio al que pertenece el objeto *objx* (este objeto puede ser una posición absoluta o relativa a un objeto o subespacio). |
| `Pos(subespaciox)` | Localización del subespacio *subespaciox*. |
| `Ubicacion_En_Mapa_Mental(porigen, pfinal)` | Ubicación en mapa mental entre la posición *porigen* y la posición *pfinal*. |
| `Subespacios_Y_Conexiones(porigen, pfinal)` | Subespacios y conexiones entre la posición *porigen* y la posición *pfinal*. |

### 5.3.3 Estructura de datos en el ATMS

Para realizar el proceso de inferencia no monótono, se deben establecer ciertas suposiciones sobre el estado de los objetivos en el modelado del estudiante puesto que, como ya hemos mencionado, el sistema no suele tener información completa al respecto para cada estudiante.

Los supuestos acerca del estado de un objetivo se pueden detectar a través de la propiedad *levelCurrentReliability* -si es 0, el valor de adquisición asociado a su estado es un valor hipotético, y si es >0, hasta el valor de la propiedad *levelReliability* del objetivo asociado, se trata de un estado de objetivo deducido tantas veces como indique el valor de esta propiedad. Un objetivo se considerará completamente alcanzado (o no) cuando la propiedad *levelCurrentReliability* tenga un valor mayor o igual que el valor de la propiedad *levelReliability* del objetivo correspondiente.

El MDP informará al ATMS de estos supuestos mediante los siguientes nodos asumidos:

$< Sup\_Est\_Objetivo(obj_i, estado_i), \{\{o_i\}\}, \{\{o_i\}\} >$
donde:

$o_i$ es el identificador de esta suposición y

$Sup\_Est\_Objetivo(obj_i, estado_i)$ expresa el estado $estado_i$ supuesto para el objetivo $obj_i$, y cuya propiedad *isAssumed* tiene valor *true*.

Durante el proceso de inferencia, a partir de cada regla $R_i$ de diagnóstico pedagógico disparada durante el encadenamiento hacia delante, se informará también al ATMS mediante la siguiente justificación de entrada:

$$\tilde{H}_i \wedge \varphi_i \Rightarrow \bigwedge_{i=1}^{n} Est\_Objetivo(obj_i, estado_i)$$

donde:

$\varphi_i = Plausible(r_j, tpo\_ejec_{r_j})$ es un nodo supuesto proporcionado al ATMS y
$\tilde{H}_i = H_1 \wedge \cdots \wedge H_m$; $H_i$ es un hecho de nuestro modelado de la forma:
$H_i = [\neg]p(X_1, X_2, \cdots, X_n)$, siendo $p$ el nombre de un predicado definido para el diagnóstico pedágogico y, $X_i$ es un término que representa una instancia en la ontología del ME, o en la ontología que representa el estado del mundo en un instante determinado.

### 5.3.4 Control de contradicciones

Una contradicción entre estados de un objetivo se pueden producir por dos motivos:

- *Contradicción con respecto a un estado de objetivo supuesto inicialmente en el modelado.* La contradicción se debe al establecimiento de cierto supuesto inicial respecto al estado de un objetivo que, durante el proceso de diagnóstico, llega a contradecirse con otro estado del objetivo deducido al ejecutar alguna de las reglas del módulo de diagnóstico pedagógico.

- *Contradicción entre estados deducidos de un objetivo.* A partir de las acciones realizadas por el estudiante, y como resultado del proceso de inferencia llevado a cabo durante el diagnóstico pedagógico, pueden derivarse inconsistencias entre estados deducidos de un objetivo (en un instante determinado que coexista un objetivo como adquirido y no adquirido a la vez).

  La regla en el MDP que permite detectar ambos tipos de contradicciones es la siguiente:

  $Est\_Objetivo(obj_i, estado_i) \wedge Est\_Objetivo(obj_i, estado_j) \wedge$
  $\neg Eq(estado_i, unknown) \wedge \neg Eq(estado_j, unknown) \wedge \neg Eq(estado_i, estado_j) \Rightarrow \perp$[5]

### 5.3.5 Módulo Gestor de Conflictos

La determinación de la causa del estado contradictorio de un objetivo y su resolución son responsabilidad del módulo Gestor de Conflictos que se describe a continuación.

#### 5.3.5.1 Detección de contradicciones

Durante el aprendizaje, el comportamiento del estudiante reflejado a través de las acciones que va ejecutando, tiende a ser inconsistente por diferentes razones. Adaptando la clasificación de las contradicciones dada por Mizoguchi (Ikeda et al. (1993)) al enfoque pedagógico propuesto, éstas se pueden clasificar de acuerdo a los siguientes criterios:

- Según la causa de la contradicción:

  1. **Contradicciones en el estado de los objetivos debidas a la no monotonía en el comportamiento o en el propio conocimiento del estudiante**.

[5] Las siguientes conjunciones entre estados de un objetivo no provocan contradicción: $unknown \wedge true = unknown$, y $unknown \wedge false = false$.

1.1 *Contradicciones en el estado de objetivos causadas por cambios en la mente del estudiante*. Durante el periodo de aprendizaje, el estudiante va adquiriendo nuevos conocimientos (por ejemplo, a través de una tutoría que proporcione una pista o una instrucción), y va rectificando ciertos comportamientos cuando se equivoca. Los cambios, frecuentes, en el comportamiento del estudiante provocan estados de objetivos inconsistentes en un cierto instante.
Así pues, cambios en el estado de un objetivo de *acquired=false* a *acquired=true* tras una tutoría indicaría muy probablemente una contradicción de este tipo. La heurística requeriría para ello del análisis de la traza de posibles acciones por parte del tutor reflejadas también en la ontología del ME. Si ha habido previamente una acción de tutoría y el estado del objetivo es no adquirido (propiedad *acquired* a valor *false*), y actualmente lo ha adquirido, es muy probable que haya sido porque el alumno ha aprendido con la tutoría. Por lo tanto, sea $objetivo_i$ el objetivo en el que se ha detectado una contradicción de este tipo tal que es cierto: $Estado\_Actual\_Objetivo(objetivo_i, true)$, es decir, el estado más actual para ese objetivo tiene la propiedad `acquired` a *true*, una heurística para detectar este tipo de cambios en la mente del estudiante puede ser la siguiente:

$$
\begin{aligned}
R_{C1}: \text{SI } & \text{Existe\_Contradiccion(objx)} \wedge \\
& \text{Estado\_Actual\_Objetivo(objx, estobjx)} \wedge \text{Es\_Alcanzado(estobjx)} \wedge \\
& \text{Se\_Obtiene\_De\_Pista(estobjx)} \rightarrow \\
& \text{Tipo\_Contradiccion(objx, cambio\_mente\_estudiante)}
\end{aligned}
\tag{5.67}
$$

También, es posible que el estudiante tras realizar mal un número determinado de veces un comportamiento que le ha llevado reiteradamente a no cumplir un cierto objetivo, él mismo deduzca (sin tutoría) que su comportamiento era erróneo y en consecuencia, lo modifique y cumpla ahora el objetivo anterior. En este caso, se puede considerar que es muy probable que no sea el puro azar lo que ha llevado a que el estudiante tras no cumplir un objetivo, al menos, un número fiable de veces, llega a cumplirlo. Por lo tanto, también se puede establecer la siguiente heurística para detectar este tipo de contradicciones:

$$
\begin{aligned}
R_{C2}: \text{SI } & \text{Existe\_Contradiccion(objx)} \wedge \\
& \text{Estado\_Actual\_Objetivo(objx, estobjx)} \wedge \text{Es\_Alcanzado(estobjx)} \wedge \\
& \geq (\text{Fiabilidad\_Est}( \\
& \quad \text{Estado\_Previo\_Objetivo(objx, esty)}, \text{fiabz}), \\
& \quad\quad \text{Fiabilidad\_Obj(objx, fiabk)}) \rightarrow \\
& \text{Tipo\_Contradiccion(objx, cambio\_mente\_estudiante)}
\end{aligned}
\tag{5.68}
$$

1.2 *Contradicciones en el estado de objetivos causadas por fallos*. En este tipo de contradicciones se pueden distinguir los siguientes subtipos:

- *Contradicciones provocadas por olvidos del estudiante*. Por ejemplo, el estudiante pudo alcanzar un cierto objetivo en una ejecución previa de una

cierta actividad y actualmente, bien por un despiste, bien porque ha olvidado cómo realizar ciertas acciones que conduzcan a la consecución de un cierto objetivo, no llega a alcanzarlo en el instante actual.
El módulo de resolución de conflictos detectará este tipo de contradicción mediante el análisis de la traza del estudiante. Para diferenciar despiste/descuido de olvido podría realizarse con una heurística independiente del dominio cuyo soporte sea el registro o "histórico" de objetivos existente en la ontología de Modelado del Estudiante propuesta. Este registro es una traza o seguimiento temporal del cumplimiento (o no) de los objetivos a lo largo del aprendizaje del alumno, de la misma forma que la se almacena en la ontología la traza de la trayectoria del alumno o la traza de las acciones que realiza el estudiante. En la traza de un objetivo se almacena la modificación o refuerzo que se produce en el estado de un objetivo como consecuencia del comportamiento del estudiante. Esto, quizás, permitiría distinguir entre estos dos subtipos de contradicciones; si un estudiante adquirió un cierto objetivo reiteradamente pero en un pasado lejano y ahora no lo adquiere, es muy probable que haya sido un olvido. La valoración de la característica de lejanía temporal en la adquisición de un objetivo es dependiente de la complejidad y duración de las actividades que se realicen. Un tipo de información que también puede reforzar la creencia y distinción de este tipo de contradicción sería el propio perfil del estudiante en cuanto a sus características de personalidad (por ejemplo, si el estudiante es olvidadizo), así como otros aspectos emocionales que puedan, con probabilidad, provocar olvidos. Todos estos aspectos están almacenados también en la ontología de Modelado del Estudiante.

$$
\begin{aligned}
R_{C3}: \text{SI } & \text{Existe\_Contradiccion}(objx) \wedge \\
& \text{Estado\_Actual\_Objetivo}(objx, estobjx) \wedge \neg \text{EsAlcanzado}(estobjx) \wedge \\
& (\geq (\text{Fiabilidad\_Est}( \\
& \quad \text{Estado\_Previo\_Objetivo}(objx, esty), fiabz), \\
& \quad \text{Fiabilidad}(objx, fiabk)) \vee \\
& (\text{Estado\_Inferido\_Por}(esty, \\
& \quad \text{cambio\_mente\_estudiante})) \wedge \\
& (\geq (\text{Tpo\_Fin\_Adq}(esty, tadq), tpo\_lim\_olvido) \vee \\
& \text{Rasgo\_Personalidad}(rasgo, olvidadizo) \vee \\
& \text{Estado\_Emocional}(est\_e, olvido)) \rightarrow \\
& \text{Tipo\_Contradiccion}(objx, olvido)
\end{aligned}
\tag{5.69}
$$

- *Contradicciones provocadas por descuidos del estudiante*. Del mismo modo, puede suceder que el estudiante alcanzara un cierto objetivo en la ejecución previa de una cierta actividad y, actualmente, no por olvido sino por despiste, no llegue a alcanzar ese objetivo en el instante actual. El módulo de resolución de conflictos distinguirá este tipo de contradicción del anterior mediante el análisis de la traza del estudiante. En concreto, como se ha

visto previamente, con una heurística independiente del dominio llevando un registro o "histórico" de objetivos. Si un estudiante adquirió un cierto objetivo en un pasado próximo reiteradamente y ahora no lo adquiere, es muy probable que sea un descuido. La proximidad temporal en la adquisición de un objetivo es dependiente de la complejidad y duración de las actividades que se realicen (puede ser el tiempo correspondiente a una misma sesión, por ejemplo), al igual que la proximidad en el tiempo descrita en el subtipo anterior. Del mismo modo, también, el perfil del estudiante ayuda en cierta medida a distinguir este tipo de contradicciones. Por ejemplo, características de personalidad como que el estudiante es despistado u otros aspectos emocionales que pueden provocar descuidos en su comportamiento.

$$
\begin{aligned}
R_{C4}: \text{SI } & \text{Existe\_Contradiccion(objx)} \wedge \\
& \text{Estado\_Actual\_Objetivo(objx, estobjx)} \wedge \neg\text{EsAlcanzado(estobjx)} \wedge \\
& (\geq (\text{Fiabilidad\_Est}( \\
& \quad \text{Estado\_Previo\_Objetivo(objx, esty)}, \text{fiabz}), \\
& \quad \text{Fiabilidad(objx, fiabk)}) \vee \\
& \text{Estado\_Inferido\_Por(esty,} \\
& \quad \text{cambio\_mente\_estudiante)} \wedge \\
& (< (\text{Tpo\_Fin\_Adq(esty, tadq)}, \text{tpo\_limite\_olvido}) \vee \\
& \text{Rasgo\_Personalidad(rasgo, despistado)} \vee \\
& \text{Estado\_Emocional(est\_e, descuido))} \rightarrow \\
& \text{Tipo\_Contradiccion(objx, descuido)}
\end{aligned}
\tag{5.70}
$$

Otra heurística para detectar la contradicción debido a descuidos es la siguiente: Si el estudiante no alcanza un cierto objetivo y, a continuación, la estrategia de tutoría decide darle otra oportunidad pero sin proporcionarle una pista, y el número de veces que no se ha alcanzado el objetivo coincide con el número de veces que sí se ha alcanzado (por lo tanto, no es probable que la contradicción haya sido debida a un cambio en la mente del estudiante). Además, no hay ciclos en el histórico de ese objetivo (es decir, alternativamente no se deduce que el estudiante ha *alcanzado,no alcanzado,alcanzado, ....* un objetivo a lo largo del aprendizaje) que pudieran hacer pensar que ha sido el azar el que ha conducido al estudiante a alcanzar el objetivo, se puede suponer que la contradicción de estados en ese objetivo es debida a un descuido del estudiante.

$R_{C5}$ : SI Existe_Contradiccion(objx) ∧
Estado_Actual_Objetivo(objx, estobjx) ∧ EsAlcanzado(estobjx) ∧
Estado_Previo_Objetivo(objx, estobjy) ∧
¬Se_Obtiene_De_Pista(estobjx) ∧
Eq(Fiabilidad_Est(estobjx), Fiabilidad_Est(estobjy)) ∧
¬Existen_Ciclos_Estados(Historico_Objetivo(objx)) →
Tipo_Contradiccion(objx, descuido)
(5.71)

1.3 *Contradicciones en el estado de objetivos causadas por el propio conocimiento inconsistente del estudiante* que causa comportamientos inconsistentes con el conocimiento actual y, como consecuencia, estados de objetivos contradictorios. El distinguir las contradicciones de este tipo es difícil. A este tipo pertenecen las contradicciones semánticas, que no se van a detectar por el método. Las heurísticas para detectar este tipo podrían también basarse en el chequeo del "histórico" del objetivo y de la traza. Por ejemplo, si un estudiante está alternativamente adquiriendo y no adquiriendo durante su historia de aprendizaje un determinado objetivo con un nivel de fiabilidad mayor o igual que la fiabilidad del objetivo, esto, probablemente, implique que tiene conocimiento (consolidado) inconsistente asociado a la consecución de esos objetivos.

$R_{C6}$ : SI Existe_Contradiccion(objx) ∧
Estado_Actual_Objetivo(objx, estobjx) ∧
Estado_Previo_Objetivo(objx, estobjy) ∧
≥ (Fiabilidad_Est(estobjx, fiabx), Fiabilidad(objx, fiaby)) ∧ (5.72)
≥ (Fiabilidad_Est(estobjy, fiabz), Fiabilidad(objx, fiaby)) ∧
Existen_Ciclos_Estados((Historico_Objetivo(objx)) →
Tipo_Contradiccion(objx, conocimiento_inconsistente)

1.4 *Contradicciones provocadas por desconocimiento del estudiante.* El estudiante actúa muchas veces por azar en estos casos. Si adquiere un objetivo con una fiabilidad menor que la fiabilidad asociada a los objetivos y no lo adquiere a continuación, lo vuelve a adquirir, y así sucesivamente, obteniendo estos estados de forma cíclica, probablemente ha actuado por azar al conocer parcialmente los objetos de conocimiento asociados a esos objetivos. El estudio del histórico de los objetivos sería un soporte fundamental que permitiría al Módulo de Tutoría detectar incluso alumnos que por diferentes problemas en el aprendizaje muestren una continúa dificultad en el aprendizaje de la actividad actual. Esta situación supondría para el tutor adoptar las estrategias de tutoría oportunas y, según el análisis del problema, obligar o recomendar al alumno que abandone la tarea actual y regrese a tareas más sencillas o realizar actividades más especí-

ficas de su estado, refuerzos en ciertos materiales de enseñanza, etc.

$$
\begin{aligned}
R_{C7}: \text{SI } & \text{Existe\_Contradiccion(objx)} \wedge \\
& \text{Estado\_Actual\_Objetivo(objx, estobjx)} \wedge \\
& \text{Estado\_Previo\_Objetivo(objx, estobjy)} \wedge \\
& ((< (\text{Fiabilidad\_Est(estobjx, fiabx)}, \text{Fiabilidad(objx, fiaby)}) \wedge \\
& \quad < (\text{Fiabilidad\_Est(estobjy, fiabz)}, \text{Fiabilidad(objx, fiaby)})) \vee \\
& \text{Realiza\_Pregunta(Objetos\_Requeridos(objx))} \rightarrow \\
& \text{Tipo\_Contradiccion(objx, desconocimiento\_objetos)}
\end{aligned}
\tag{5.73}
$$

2. **Contradicciones en el estado de los objetivos surgidas por los supuestos adoptados en el modelado**. El ME no representa completamente el estado actual de conocimiento del estudiante. Ya desde su estado inicial se pueden realizar, como ya se ha descrito previamente, hipótesis sobre el estado de ciertos objetivos (supuestamente adquiridos o supuestamente no adquiridos). En el transcurso del aprendizaje del estudiante, algunas de estas suposiciones pueden llegar a ser inconsistentes con el estado de los objetivos deducidos en el proceso de razonamiento del diagnóstico pedagógico. Por lo tanto, estas contradicciones están causadas por inconsistencias entre el comportamiento actual del estudiante que permite deducir el estado de ciertos objetivos y el conocimiento a nivel de objetivos representado en el ME hasta el momento.

   En el método, se supone que, un tipo de contradicción que no sea de lo tipos anteriormente descritos, es una contradicción por supuestos en el modelado que debe también resolverse como se indica a continuación en el método. Este supuesto es registrado por el módulo Gestor de Conflictos tras detectarse la contradicción (propiedad *inferBy=modelling_assumption* en la instancia del estado de objetivo (instancia de *Objective_State*) que dio lugar a la contradicción.

- Según el tratamiento de la contradicción:

  1. Las contradicciones del tipo 1.1, 1.2, 1.4 y 2 deben resolverse revisando el ME con el método de diagnóstico. La forma de resolverse depende del tipo de contradicción generado así como del histórico del objetivo, eliminando en todos los casos alguno de los estados del objetivo que han provocado la contradicción. Asimismo, el módulo de resolución de conflictos, generará el entorno consistente candidato e informa de ello al ATMS.
  2. Las contradicciones del tipo 1.3 no deberían resolverse. Se trata de conocimiento contradictorio propio del estudiante que debería estar presente en el ME y ser usado para realizar una tutoría adecuada y efectiva.

#### 5.3.5.2 Resolución de contradicciones

Las contradicciones de los tipos 1.1, 1.2, 1.4, y 2 deben ser resueltas por el método. En general, usaremos la heurística de mantener el estado del objetivo más reciente, es decir, dar más fiabilidad al comportamiento del estudiante más actual sobre el pasado. Esta forma de resolver las contradicciones tiene excepciones. Por ejemplo, se supone que ha habido previamente una

contradicción de tipo cambio en la mente del estudiante y, como consecuencia, el estudiante ha alcanzado un cierto objetivo. Posteriormente, por un descuido durante el desarrollo de la actividad de aprendizaje, el estudiante vuelve a no alcanzar el mismo objetivo (contradicción de tipo descuido). Este es un caso particular en el que es conveniente mantener no el estado de ese objetivo actual (con propiedad *acquired=false*) sino el anterior. Para facilitar en gran medida la tarea de resolución y no volver a analizar el histórico del objetivo que cumple las condiciones mencionadas previamente, se mantiene en el estado de cada objetivo existente en la ontología el tipo de contradicción que se supuso para él, si tal contradicción hubiera existido.

Las reglas de resolución de contradicciones se muestran a continuación:

- Si el tipo de contradicción entre estados de un objetivo supuestamente ha sido provocada por un descuido del estudiante que ha conducido actualmente a un estado alcanzado del objetivo, para resolver la contradicción se elimina el estado previo no alcanzado de la ontología de ME (por ejemplo, el estudiante aplica una acción que no es la que corresponde ejecutar de acuerdo al plan de la actividad, como consecuencia se deduce un objetivo no alcanzado y, posteriormente, sin realizar más intentos y sin proporcionarle el tutor pistas, realiza una acción que permite deducir el objetivo como alcanzado). Del mismo modo, se resuelven las contradicciones de tipo olvido o por cambio en la mente del estudiante.

$$
\begin{aligned}
R_{C8}: \text{SI } & (\text{Tipo\_Contradiccion(objx, descuido)} \wedge \\
& \text{Estado\_Actual\_Objetivo(objx, estobjx)} \wedge \\
& \text{Es\_Alcanzado(estobjx))} \vee \\
& \text{Tipo\_Contradiccion(objx, olvido)} \vee \\
& \text{Tipo\_Contradiccion(objx, cambio\_mente\_estudiante)} \rightarrow \\
& \text{Eliminar\_Estado(Estado\_Previo\_Objetivo(objx, estobjy))}
\end{aligned}
\tag{5.74}
$$

- Si el tipo de contradicción entre estados de un objetivo supuestamente fue provocada por un descuido del estudiante precedido por un cambio en la mente del estudiante (por ejemplo, un estudiante realiza una acción que no es la correcta de acuerdo al plan, como consecuencia se deduce el objetivo no alcanzado, el tutor le proporciona una pista que permite deducir el objetivo alcanzado -cambio en la mente del estudiante- y, posteriormente, el estudiante vuelve a realizar una acción que vuelve a deducir el objetivo no alcanzado -descuido-. Este tipo de contradicción por descuido se resuelve al revés que en el caso anterior, es decir, eliminando el estado actual del objetivo de la ontología de ME.

$$
\begin{aligned}
R_{C9}: \text{SI } & \text{Tipo\_Contradiccion(objx, descuido)} \wedge \\
& \text{Estado\_Previo\_Objetivo(objx, estobjy)} \wedge \\
& \text{Estado\_Inferido\_Por(estobjy,} \\
& \quad \text{cambio\_mente\_estudiante)} \rightarrow \\
& \text{Eliminar\_Estado(Estado\_Objetivo\_Actual(objx, esty))}
\end{aligned}
\tag{5.75}
$$

$$
\begin{aligned}
R_{C10}: \text{SI } & \text{Tipo\_Contradiccion(objx, desconocimiento\_objetos)} \wedge \\
& \text{Estado\_Objetivo\_Actual(objx, estobjx)} \wedge \\
& \text{Estado\_Objetivo\_Previo(objx, estobjy)} \wedge \\
& \leq (\text{Fiabilidad\_Acumulada(estobjy, fiaby)}, \\
& \quad \text{Fiabilidad\_Acumulada(estobjx, fiabx)}) \rightarrow \\
& \text{Eliminar\_Estado(estobjy)}
\end{aligned}
\tag{5.76}
$$

- Si el tipo de contradicción entre estados de un objetivo supuestamente fue provocada por desconocimiento de objetos, es muy probable que en el comportamiento del estudiante predomine el azar y existan ciclos en el histórico del objetivo, es decir, obtención a lo largo del aprendizaje del estudiante de estados alternativamente distintos (alcanzado-no alcanzado) de ese objetivo. En este caso, se considera el número de veces que se han obtenido cada uno de los estados (alcanzado y no alcanzado) del objetivo a lo largo del aprendizaje del estudiante (a través del histórico registrado en la ontología de ME). Para resolver esta contradicción, se mantendrá en la ontología el estado del objetivo que se haya deducido más veces y se eliminará el contrario.

$$
\begin{aligned}
R_{C11}: \text{SI } & \text{Tipo\_Contradiccion(objx, desconocimiento\_objetos)} \wedge \\
& \text{Estado\_Objetivo\_Actual(objx, estobjx)} \wedge \\
& \text{Estado\_Objetivo\_Previo(objx, estobjy)} \wedge \\
& > (\text{Fiabilidad\_Acumulada(estobjy, fiaby)}, \\
& \quad \text{Fiabilidad\_Acumulada(estobjx, fiabx)}) \rightarrow \\
& \text{Eliminar\_Estado(estobjx)}
\end{aligned}
\tag{5.77}
$$

- Si el tipo de contradicción entre estados de un objetivo supuestamente fue provocada por supuestos adoptados en el modelado, se resuelve la contradicción eliminando de la ontología aquél estado que se ha deducido menos veces, es decir, con un valor inferior en su propiedad *levelCurrentReliabily*.

$$
\begin{aligned}
R_{C12}: \text{SI } & \text{Tipo\_Contradiccion(objx, supuesto\_modelado)} \wedge \\
& \text{Estado\_Actual\_Objetivo(objx, estobjx)} \wedge \\
& \text{Estado\_Previo\_Objetivo(objy, estobjy)} \wedge \\
& \leq (\text{Fiabilidad\_Est(estobjy, fiaby)}, \\
& \quad \text{Fiabilidad\_Est(estobjx, fiabx)}) \rightarrow \\
& \text{Eliminar\_Estado(estobjy)}
\end{aligned}
\tag{5.78}
$$

$$
\begin{aligned}
R_{C13}: \text{SI } & \text{Tipo\_Contradiccion(objx, supuesto\_modelado)} \wedge \\
& \text{Estado\_Actual\_Objetivo(objx, estobjx)} \wedge \\
& \text{Estado\_Previo\_Objetivo(objy, estobjy)} \wedge \\
& > (\text{Fiabilidad\_Est(estobjy, fiaby)}, \\
& \quad \text{Fiabilidad\_Est(estobjx, fiabx)}) \rightarrow \\
& \text{Eliminar\_Estado(estobjx)}
\end{aligned}
\tag{5.79}
$$

*Tratamiento de las reglas con consecuente disyunción.* Las reglas del módulo de diagnóstico pedagógico que contienen en su consecuente una disyunción se representan con tantas reglas en Jena como términos haya en la disyunción del consecuente. El módulo Gestor de Conflictos almacena la información referente a estas reglas. Si se alcanza un estado de objetivo durante el disparo de reglas y, posteriormente, se llega a una contradicción con él, se reforzará la creencia en el(los) otro(s) término(s) de la disyunción. Esto último consistirá en incrementar la fiabilidad actual de los estados de objetivos deducidos en el resto de las reglas asociadas a la disyunción.

## 5.4 Metodología para la aplicación a un sistema concreto

En el presente trabajo se pretende no sólo proporcionar un nuevo método de modelado del estudiante, ya descrito previamente, sino también unas guías metodológicas que permitan a cualquier diseñador de SITs adaptar y aplicar fácilmente el método propuesto a nuevos dominios de aprendizaje, fundamentalmente de tipo procedimental. Para lograrlo, se han abstraído los pasos seguidos al aplicar el modelado del estudiante propuesto a varios ejemplos de dominios de enseñanza/aprendizaje (aprendizaje de un GUI y varios entornos virtuales de entrenamiento de tipo procedimental).

La adaptación del método de ME a nuevos entornos consiste principalmente en dos procesos:

- **Adaptación de la Ontología General (OG) de ME** (ontología descrita en la sección 5.2.5). Un nuevo entorno de enseñanza/aprendizaje supondrá añadir nueva información al Modelado del Estudiante (sobre nuevos tipos de acciones, objetos del entorno específico, tipos de objetivos, etc.). La aplicación de la Ingeniería Ontológica en el Modelado del Estudiante propuesto facilita esta tarea; reutilizar ciertos conocimientos existentes, posiblemente, en otras ontologías u otros recursos no ontológicos, la propia adaptabilidad y extensibilidad de las ontologías en general, y en concreto de la OG de ME, facilita su aplicación a los nuevos entornos de aprendizaje añadiendo los conocimientos de los nuevos dominios de aprendizaje en términos del formalismo de ontologías. Asimismo, el uso de la metodología NeOn de construcción de ontologías ayuda, mediante sus guías metodológicas, a la transformación de la ontología OG para su aplicación a los nuevos dominios de enseñanza. Finalmente, unas guías de indiquen los pasos a realizar para aplicar el método presentado son claves para facilitar este proceso.

- **Adaptación del método de diagnóstico**. Un nuevo entorno de enseñanza/aprendizaje supondrá además la ampliación/modificación de las reglas del módulo de diagnóstico pedagógico desarrollado. Dadas las características específicas del método y, en especial, del

módulo de diagnóstico pedagógico, su adaptabilidad a nuevos entornos requiere unas guías metodológicas que ayuden a simplificar la complejidad de esta tarea.

En la figura 5.12 se muestra un esquema global de la aproximación metodológica propuesta:

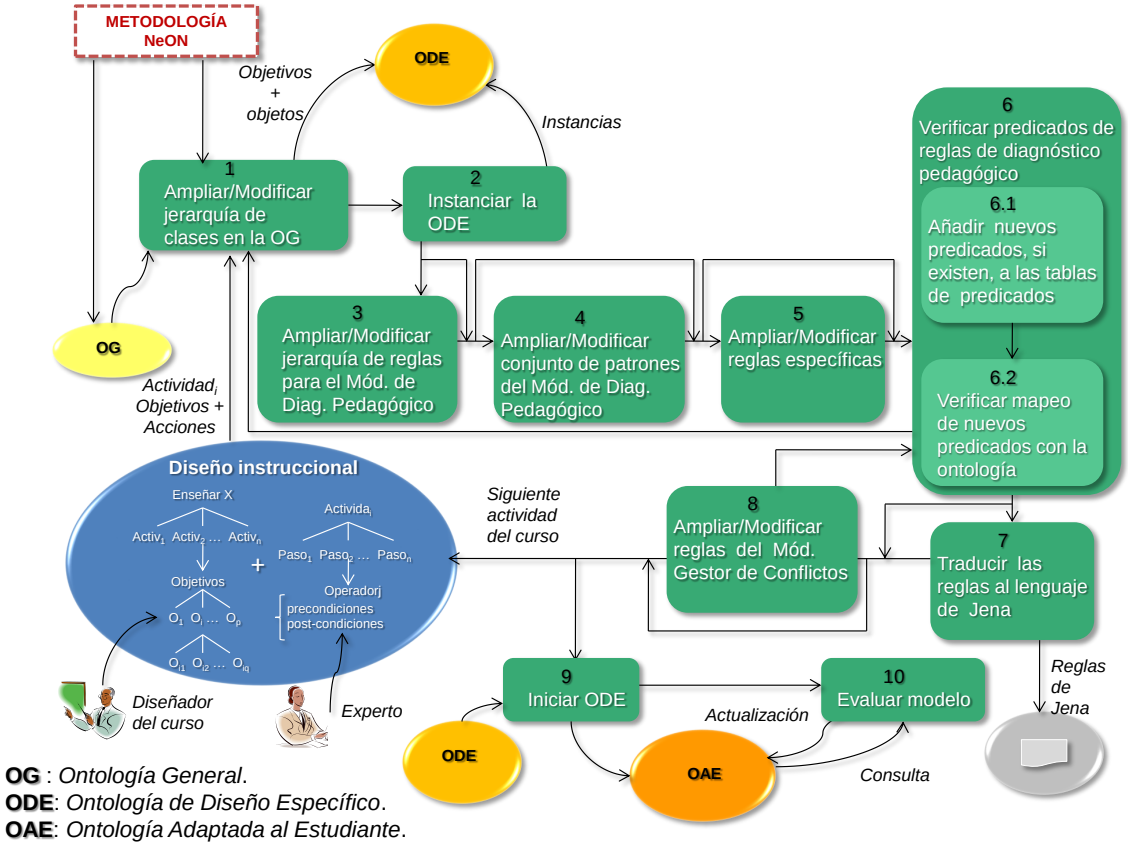


**Figura 5.12:** Esquema de la metodología de aplicación a un sistema concreto

Como se puede observar en el esquema 5.12, el diseño de cualquier SIT requiere un diseño instruccional para la materia que se va a enseñar, y ello conlleva definir un grupo de actividades para ella y los objetivos que el estudiante debería alcanzar en cada actividad. Por otro lado, el módulo Experto, usando un planificador, determinará los pasos o acciones que deberían realizarse para concluir cada actividad con éxito. Tanto los objetivos didácticos definidos como los pasos de planificador para cada actividad son fundamentales en la metodología de aplicación y en el método presentado.

Para cada actividad del curso a enseñar, la secuencia de pasos (o actividades) de la metodología es la que se muestra en la Figura 5.12. Las actividades (y sus tareas) de la metodología se identifican en el esquema mediante cajas redondeadas y numeradas:

- Actividades relativas a la ontología del ME:
  - ***Actividad 1**: Ampliar/Modificar jerarquía de clases en la OG del ME.* Partiendo de la Ontología General del ME (OG), esta actividad supone extender o especializar esta

ontología para su aplicación al nuevo dominio de aprendizaje. Aplicando la metodología NeOn de construcción de ontologías, será necesaria inicialmente una revisión y ampliación de las cuestiones de competencia que permitan extraer las nuevas clases necesarias a añadir a la OG presentada en la sección 5.2.5.

Por lo tanto, esta actividad supone, generalmente, añadir a la OG nuevas clases en alguna de las siguientes jerarquías: a) en la jerarquía de objetos de conocimiento, y sugerido por el experto, objetos estructurales relativos a la materia a enseñar (por ejemplo, objetos manipulables presentes en el espacio geométrico del nuevo entorno, como una probeta en un laboratorio, nuevas relaciones entre objetos, etc.), u objetos procedimentales tales como nuevos tipos de acciones propias del nuevo entorno que surgen a partir de los pasos del planificador, nuevos planes asociados a la materia a enseñar, etc., b) sugerido por el diseñador del curso, en la jerarquía de objetivos de aprendizaje (por ejemplo, partiendo de los objetivos diseñados para cada cada actividad, puede surgir la necesidad de una nueva clase objetivos de análisis, de aplicación, etc.). Asimismo, es posible que al incluir una cierta clase en la ontología, sea conveniente modificar parte de la estructura de la ontología (por ejemplo, por motivos de herencia) o, tal vez, añadir alguna propiedad a una cierta clase ya existente. En este punto de la metodología es importante resaltar de nuevo que, en ciertas ocasiones, gracias a las ventajas de la Ingeniería Ontológica, se podrá reutilizar directamente el conocimiento de otras ontologías ya creadas (por ejemplo, de ontologías existentes en ciertos repositorios de ontologías) o de recursos no ontológicos. Para ello, se aplicarán diferentes escenarios de construcción de la metodología NeOn utilizada ya previamente en la construcción de la OG (por ejemplo, los escenarios 2 y 8 de la metodología). El resultado de esta actividad es la Ontología General modificada denominada *Ontología de Diseño Específico* (ODE en el esquema).

- ***Actividad 2**: Instanciar la ODE.* Este paso requiere definir las instancias específicas propias de cada actividad a desarrollar, a partir de las clases ya definidas en la *Actividad 1*. Por ejemplo, en un entorno virtual de aprendizaje, supondrá la definición de instancias de nuevos objetos geométricos, subescenarios, etc., del escenario 3D, si es el caso del nuevo entorno de aprendizaje, instancias de acciones de una determinada clase en el nuevo entorno, etc.

- Actividades relativas al método de diagnóstico del ME:

  - ***Actividad 3**: Ampliar/Modificar la jerarquía de reglas para el Módulo de Diagnóstico Pedagógico.* Aunque se ha definido una clasificación de criterios de diagnóstico bastante extensa (y, basada en ella, un conjunto de categorías de reglas), sin embargo, nuevos objetivos o acciones añadidos al ME como consecuencia de la *Actividad 1* pueden eventualmente requerir una ampliación y/o modificación de las jerarquías de reglas. Por ejemplo, hay criterios que dependen fuertemente del tipo de acción (5.3.2). En este caso, es posible que si se han añadido nuevos tipos de acciones en la jerarquía de acciones de la ontología, esto conduzca a la ampliación/modificación de alguna de las categorías de reglas basadas en esos criterios.

  - ***Actividad 4**: Ampliar/modificar conjunto de patrones del Módulo de Diagnóstico Pedagógico.* Del mismo modo que en la actividad previa, es probable que sea necesario ampliar/modificar los patrones de reglas definidos genéricamente en una cierta

categoría. Esto puede suceder bien porque haya sido necesario realizar el paso previo de la metodología y sea imprescindible añadir nuevos patrones, o bien porque sin haber realizado la *Actividad 3*, el nuevo escenario de aprendizaje/enseñanza lo requiera (por ejemplo, es probable que una cierta actividad a desarrollar por el estudiante en un determinado entorno virtual de entrenamiento presente ciertos errores específicos de ese tipo de entornos y, como consecuencia, se defina en esta categoría de reglas ya existente un nuevo patrón).

- ***Actividad 5**: Ampliar/modificar reglas específicas*. Esta actividad, como las actividades 3 y 4 puede ser necesaria, aunque con menos frecuencia. Supone ampliar el conjunto de reglas específicas, no patrones de reglas, con alguna regla propia del nuevo entorno de aprendizaje, o modificar alguna de las reglas específicas ya existentes. Por ejemplo, porque en el diagnóstico de errores típicos intrínsecos a la actividad actual sea necesario añadir una regla específica para el nuevo entorno de aprendizaje.
- ***Actividad 6**: Verificar predicados de las reglas de diagnóstico pedagógico*. Esta actividad es necesaria sólo si se han realizado algunas de las actividades: 3, 4, ó 5. Si se han añadido/modificado alguna jerarquía de reglas, algún patrón de reglas, o nuevas reglas específicas, se requieren las siguientes tareas:
  - ***Tarea 6.1**: Añadir nuevos predicados, si existen, a la correspondiente tabla de predicados*. Como resultado de cualquiera de las actividades: 3, 4, ó 5, es posible que se necesite añadir algún nuevo predicado a una de las tablas definidas para tal fin (véase sección 5.3.2), de acuerdo a la parte de la jerarquía de la ontología referenciada en él. Además, es necesario realizar la actividad siguiente.
  - ***Tarea 6.2**: Verificar el mapeo con la ontología*, es decir, garantizar que existe en la ontología la clase y/o la propiedad correspondiente que coherentemente se equipare con los argumentos de los predicados añadidos. Este último punto puede suponer añadir nuevamente alguna propiedad olvidada o alguna clase o, la modificación de otras ya existentes (es decir, regresar a la *Actividad 1* del método).
- ***Actividad 7**: Traducir las reglas al lenguaje de Jena*. Si se han realizado alguna de las actividades: 3, 4, ó 5, y tras el resultado de la *Actividad 6*, las nuevas reglas deben ser añadidas al conjunto de reglas del Módulo de Diagnóstico Pedagógico en lenguaje de Jena.
- ***Actividad 8**: Ampliar/modificar reglas del Módulo Gestor de Conflictos*. Es probable que para alguna actividad del curso se tengan que añadir o modificar algunas reglas específicas en el Módulo Gestor de Conflictos. La modificación podría afectar al conjunto de reglas de este módulo que permiten detectar contradicciones según su causa. o al conjunto de reglas que incorpora para resolver las contradicciones. Los predicados de las reglas añadidas pueden también ser nuevos. En este caso, es necesario incluirlos también en las tablas de predicados establecidas en el método de diagnóstico.

- Actividad relativa a la inicialización de la ontología del ME con información del estudiante.

  Una vez que, para las actividades del curso, se han realizado todas las modificaciones en la ontología y en las reglas pedagógicas para su correcta aplicación al sistema concreto de enseñanza, es necesaria la siguiente actividad:

- ***Actividad 9***: *Iniciar la ODE*, obtenida tras la *Actividad 2*, *con información del estudiante*; con el estado inicial de los objetivos, alcanzados o no por él en las actividades del curso, y con su perfil de estudiante. El tutor o el experto definirá el estado inicial de cada objetivo para el estudiante con la propiedad *adquirido* con valor *true* (el sistema asume que el estudiante ya ha alcanzado ese objetivo), con valor *false* (el sistema asume que el estudiante no ha alcanzado todavía ese objetivo), o con valor *desconocido* (el sistema no sabe nada inicialmente acerca de la consecución del objetivo por parte del estudiante). La ontología así iniciada se denomina *Ontología Adaptada al Estudiante* (OAE).

- Actividad relativa a la evaluación.

  - ***Actividad 10***: Finalmente, una vez que ha adaptado la ontología del Modelado del Estudiante al nuevo entorno así como el método de diagnóstico, se debe realizar la actividad de *Evaluación de la ontología y del ME en general*. La evaluación de la ontología final del ME (OAE) crea un juicio técnico de la ontología, de sus entornos software asociados, y de la documentación. En el caso concreto de la construcción de la ontología del ME se han usado para su evaluación las cuestiones de competencia definidas (un fragmento se mostró en el apartado 5.2.2.3). Como se puede observar en el esquema 5.12, la evaluación del modelo implica realizar operaciones de consulta y actualización sobre la OAE. La actualización de la OAE supone, por ejemplo, el registro de información relativa al estado de los conocimientos, traza del comportamiento del estudiante, traza de sus objetivos, alcanzados o no por el estudiante, estado de la sesión de aprendizaje del estudiante, eliminación de estados de objetivos, etc. Todo ello, en términos de la ontología.

CAPÍTULO 6

# APLICACIÓN DEL MODELO

El Modelado del estudiante propuesto se ha adaptado para ser aplicado a tres prototipos de demostración. El primer prototipo pertenece a un entorno de aprendizaje de GUIs, y los otros dos a Entornos Virtuales de Aprendizaje, en concreto, Entornos Virtuales de Entrenamiento de tipo procedimental. A continuación, siguiendo la metodología propuesta en la sección 5.4, se describe cómo se ha extendido para cada prototipo la ontología de Modelado del Estudiante general presentada en este trabajo así como la adaptación del método de diagnóstico propuesto para dos de los prototipos de demostración.

## 6.1 Modelado del Estudiante para prototipo de demostración I: Aprendizaje de un GUI, editor de textos

Un ejemplo de dominios a los que se puede aplicar el modelado del estudiante descrito en este documento es el aprendizaje de actividades referentes a un GUI. Por ejemplo, en el ámbito del aprendizaje de un editor de textos, una posible actividad es *Apertura de un archivo* perteneciente al plan de estudios de cualquier aplicación que maneje archivos. A continuación, se muestran en la tabla 6.1 los pasos del planificador para dicha actividad así como los operadores aplicados en cada uno de ellos, de acuerdo al enfoque de diseño pedagógico en el que se sustenta la solución (figura 5.1). Ambos, pasos y operadores asociados, influirán notablemente en el contenido de la ontología del estudiante, como se verá más adelante. Por brevedad, se ha simplificado el ejemplo considerando sólo el aprendizaje a través de los menús (un análisis completo debería tener en cuenta también esta actividad realizada a través de los botones de la barra de herramientas o la combinación de teclas de función).

Tabla 6.1: Pasos del planificador para la actividad *Abrir Archivo*

| PASOS | OPERADORES APLICADOS |
|---|---|
| **1.** Coger ratón de su posición. | *coger(estudiante, objeto, p_objy)* |
| **2.** Arrastrar ratón desde posición origen hasta menú *Archivo*. | *arrastrar(estudiante, objeto, porigen, pobjy)* |
| **3.** Seleccionar menú *Archivo* pulsando el botón izquierdo del ratón sobre él. | *pulsarbotonizq(estudiante, objeto)* |
| **4.** Arrastrar ratón hasta opción. | *arrastrar(estudiante, objeto, porigen, pobjy)* |
| **5.** Seleccionar opción *Abrir* pulsando el botón izquierdo del ratón sobre ella. | *pulsarbotonizq(estudiante, objeto)* |
| **6.** Si el archivo está en el directorio actual, arrastrar ratón hasta nombre de archivo. Si no, arrastrar ratón hasta directorio de más alto nivel que contenga en su jerarquía el archivo a abrir (en cuadro de diálogo o lista desplegable de directorios). | *arrastrar(estudiante, objeto, porigen, pobjy)* |
| **7.** Seleccionar objeto (archivo o directorio) en el directorio actual donde se encuentra el ratón. | *pulsarbotonizq(estudiante, objeto)* |
| **8.** Si objeto seleccionado es el archivo a abrir, arrastrar ratón hasta botón *Abrir*. Si no es aún el archivo a abrir, ir a 5. | *arrastrar(estudiante, objeto, porigen, pobjy)* |
| **9.** Seleccionar botón *Abrir* pulsando botón izquierdo del ratón sobre él. | *pulsarbotonizq(estudiante, objeto)* |

Siguiendo siempre un enfoque pedagógico, en la actividad *Abrir archivo* se han definido sus objetivos (cognitivos) mostrados, a nivel abstracto, en la Tabla 6.2 . Se supone que el estudiante ya ha adquirido conocimientos acerca de conceptos propios del dominio de trabajo a nivel lógico, tales como unidad, carpeta, archivo, jerarquía de directorios, etc., a nivel físico, tales como ratón, teclado, monitor, etc., o a nivel de conceptos propios del entorno de trabajo, tales como barra de menús, menús, estructura de menús, cuadro de diálogo, cuadro de texto, botón y tipos, etc.

Tabla 6.2: Objetivos cognitivos de la actividad *Abrir archivo*

| CLASE DE OBJETIVOS | INSTANCIAS DE OBJETIVOS |
|---|---|
| ***Objetivos de conocimiento*** | ▪ Saber *reconocer* cada objeto (elemento) a usar de la aplicación para esta actividad: barra de menús, menú *Archivo,* la opción para abrir el archivo, los botones de *Abrir* y *Cancelar* la apertura del archivo, etc.<br>▪ Saber *dónde está* cada objeto (elemento) a usar de la aplicación para esta actividad.<br>▪ Saber *para qué sirve* cada objeto (elemento) a usar de la aplicación para esta actividad (por ejemplo, el botón *Cancelar* del diálogo sirve para cancelar la operación de apertura de un archivo).<br>▪ Saber *el efecto de aplicar un determinado operador* a cada objeto (elemento) a usar de la aplicación para esta actividad (por ejemplo, cuando se selecciona el botón *Cancelar* del diálogo *Abrir*, desaparece el diálogo para seleccionar el archivo a abrir).<br>▪ Saber *qué es* cada objeto (elemento) a usar de la aplicación para esta actividad.<br>▪ Saber que *existen diferentes tipos de acciones* posibles a realizar sobre los objetos en la actividad: *arrastrar* ratón hasta un objeto, *seleccionar objeto*, etc.<br>▪ Saber que *es posible aplicar un determinado operador a un cierto objeto* de la aplicación para la actividad.<br>▪ Saber *cómo realizar las distintas acciones* posibles en la actividad. Por ejemplo, cómo *arrastrar*el ratón hasta un objeto. |
| ***Objetivos de aplicación*** | ▪ Ser capaz de *realizar* las diferentes acciones para llevar a cabo la actividad.<br>▪ Ser capaz de *mantener la ubicación dentro del mapa mental* (objetos y conexión entre ellos) durante la navegación. |
| ***Objetivos de análisis*** | ▪ Ser capaz de *seleccionar* de entre todos los objetos de una determinada clase, cuál usar a continuación (por ejemplo, el menú *Archivo* de entre todos los menús existentes en la aplicación).<br>▪ Ser capaz de *clasificar* un determinado objeto dentro de una determinada categoría (por ejemplo, opción *Abrir* entre todas las opciones del menú *Archivo*). |

Tabla 6.2: Objetivos cognitivos de la actividad *Abrir archivo* (cont.)

| CLASE DE OBJETIVOS | INSTANCIAS DE OBJETIVOS |
|---|---|
| ***Objetivos de síntesis*** | ▪ Ser capaz de *construir un plan* para abrir un archivo mediante el uso de los menús.<br>▪ Ser capaz de *construir una ruta de navegación válida* para abrir un determinado archivo. |
| ***Objetivos de evaluación*** | ▪ Ser capaz de *elegir una trayectoria de navegación* que en el menor número de pasos le permita alcanzar el objetivo. |

### 6.1.1 Ontología del estudiante para prototipo de demostración I: Aprendizaje de un GUI, editor de textos

La actividad *Abrir archivo* requiere realizar una extensión y especialización de la ontología general desarrollada para este entorno de aprendizaje. Esto supone, como se vio en el apartado 3.3.6.1, aplicar el escenario 8 de construcción de ontologías de la metodología NeOn utilizada y, por lo tanto, realizar entre otras actividades una ampliación de las cuestiones de competencia. Un fragmento de estas cuestiones se presentan a continuación, del mismo modo que se especificaron en 5.2.2.3.

Tabla 6.3: Cuestiones de competencias específicas relacionadas con un *objeto de aprendizaje*

| IDENTIFICADOR | CUESTIÓN DE COMPETENCIA |
|---|---|
| **CC205.** | *¿Un cierto objeto es padre de otro, por ejemplo, un directorio es padre de otro directorio?* |
| **CC206.** | *¿Un cierto objeto es hijo de otro, por ejemplo, un archivo es hijo de un directorio?* |
| **CC207.** | *¿Esta seleccionado un cierto objeto GUI como, por ejemplo, un botón de pulsación?* |
| **CC208.** | *¿Se puede navegar a un cierto objeto GUI desde otro?* |
| **CC209.** | *¿A qué objeto GUI pertenece otro, por ejemplo, a qué menú pertenece una cierta opción de menú?* |

Tabla 6.3: Cuestiones de competencias específicas relacionadas con un *objeto de aprendizaje* (cont.)

| IDENTIFICADOR | CUESTIÓN DE COMPETENCIA |
|---|---|
| **CC210.** | *¿Hasta que posición absoluta hay que arrastrar un objeto como el ratón?* |
| **CC211.** | *¿Hasta que objeto GUI hay que arrastrar un objeto, por ejemplo, el ratón?* |
| **CC212.** | *¿Qué botón hay que pulsar para lograr un cierto objetivo como, por ejemplo, seleccionar un objeto GUI?* |
| **CC213.** | *¿Está activo un cierto objeto GUI, por ejemplo, una opción de un menú?* |

De las cuestiones de competencia añadidas se han extraído la definición de nuevas clases en la jerarquía de objetos de conocimiento de la ontología del ME. Una vez realizado el proceso de búsqueda de recursos ontológicos, no se ha hallado ninguno que satisfaciera las necesidades especificadas. El siguiente paso fue analizar posibles candidatos no ontológicos a ser usados siguiendo las guías metodológicas correspondientes de la metodología NeOn para construcción de redes de ontologías. Los procesos de reutilización y re-ingeniería de recursos no ontológicos pertenecen al escenario 2 de desarrollo llamado *Building Networks by Reusing and Re-engineering Non-Ontological Resources* de la metodología NeOn (Suárez-Figueroa et al. (2008a)).

Como ya se describió en (3.3.6.1), reutilizar recursos no ontológicos, es el proceso de elegir los recursos no ontológicos (por ejemplo, bases de datos, vocabularios controlados, esquemas de clasificación, etc.), para el desarrollo de ontologías. La re-ingeniería de recursos no ontológicos es definida como el proceso de recuperar y transformar un recurso no ontológico en una ontología (Suárez-Figueroa et al. (2008a)).

Las actividades llevadas a cabo para el proceso de reutilización de recursos no ontológicos se describen brevemente a continuación (se pueden ver detalladamente en Suárez-Figueroa et al. (2008a)):

- Actividad 1. *Buscar recursos no ontológicos.* El objetivo de esta actividad es buscar recursos no ontológicos a partir de sitios Web altamente fiables, lugares relacionados con el dominio y recursos dentro de organizaciones. La actividad se lleva a cabo teniendo en cuenta el documento de especificación de requerimientos de la ontología de Modelado del Estudiante. La salida de la actividad es un conjunto de candidatos de recursos no ontológicos que podrían presentar cualquiera de las tipologías descritas en (Suárez-Figueroa et al. (2008a)). En esta actividad, los resultados de la búsqueda fueron diversos *frameworks* (Swing, AWT, Qt) que incluyen bibliotecas para los objetos GUIs requeridos para el prototipo de demostración I al que se aplica el modelado del estudiante presentado.

- Actividad 2. *Valorar el conjunto de recursos no ontológicos*. Partiendo del conjunto de candidatos de recursos no ontológicos previos, se valoran éstos teniendo en cuenta como criterios medibles los siguientes: la cobertura, la precisión y el consenso como criterio subjetivo. La salida de esta actividad es una tabla de valoración que muestren los criterios de evaluación para cada recurso no ontológico. En el desarrollo de la ontología del estudiante de este trabajo predominó la valoración basada en el criterio del consenso con los expertos del dominio y tras los resultados del análisis exhaustivo realizados en la actividad 1.

- Actividad 3. *Seleccionar los recursos no ontológicos más apropiados*. Teniendo en cuenta la valoración realizada en la actividad 2, se debe realizar por los expertos del dominio, desarrolladores software y profesionales de ontologías la selección manual de los recursos no ontológicos más adecuados buscando consenso, alto valor de cobertura y precisión. La salida de esta actividad para el desarrollo de la ontología de Modelado del Estudiante es una lista de los recursos no ontológicos que cubren de alguna manera el dominio esperado. En nuestro caso, para la generación de objetos de conocimiento GUIs, la lista fue únicamente el framework *Swing*.

Para realizar la re-ingeniería del recurso no ontológico (framework *Swing*) previo, es decir, transformar este recurso en ontologías, también se han seguido en gran medida las guías de la metodología de construcción de redes de ontologías NeOn, y que se resumen a continuación:

- Actividad 1. *Ingeniería inversa de recursos no ontológicos*. El objetivo de esta actividad ha consistido en analizar el framework previamente seleccionado, para identificar sus componentes subyacentes y crear representaciones del recurso en diferentes niveles de abstracción (diseño, requerimientos y conceptual). De este modo, la estructura y el contenido de los recursos no ontológicos pueden ser recuperados.

- Actividad 2. *Transformación de los recursos no ontológicos*. El objetivo de esta actividad ha consistido en generar un modelo conceptual a partir de los recursos no ontológicos.

  Dado el tipo de recursos no ontológico, el framework *Swing*, se ha implementado un programa que, de acuerdo al tipo de representación del recurso no ontológico (archivos *html* generados con *javadoc* y que especifican, entre otros elementos, las clases de objetos GUIs de *Swing*), navega por estos archivos para extraer la taxonomía[1] de objetos gráficos en lenguaje *owl ad-hoc*.

- Actividad 3. *Ingeniera hacia adelante de ontología*. El objetivo de esta actividad es obtener una nueva implementación de la ontología sobre las bases del nuevo modelo conceptual resultante de la actividad previa. Para ello, se usan los niveles de abstracción de la ontología para describir esta actividad ya que están directamente relacionados con el proceso de desarrollo de la ontología.

[1] Una taxonomía es una forma de organizar una ontología como una estructura jerárquica de clases sólo relacionadas mediante relaciones de subsunción.

El modelo general de NeOn anterior propuesto para la re-ingeniería de recursos no ontológicos basado en el modelo de re-ingeniería software, se presenta en la figura 6.1:

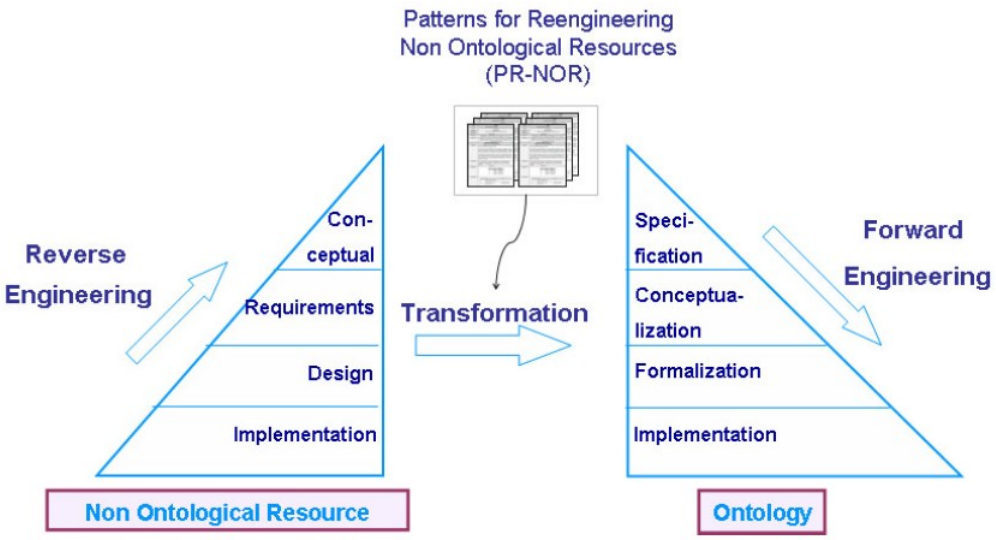


**Figura 6.1:** Modelo de Reingeniería de la metodología NeOn para recursos no ontológicos (Suárez-Figueroa et al. (2008a))

Como resultado del proceso anterior, se obtuvo una nueva subjerarquía en la ontología. La clase principal en esta subjerarquía se ha denominado **Computer_Interface_Object**. Entre sus subclases se encuentran: **File**, **Directory** y **Widget**. Esta última clase representa todos los objetos relacionados con los tipos de entornos de aprendizaje que requieren navegación, como es el caso de esta actividad. De este modo, la subjerarquía de **Widget** abarcaría conceptos tales como menú, opción de menú, caja de diálogo, cuadro de texto, etc. La subjerarquía mencionada se muestra en la siguiente figura y deriva de los módulos de clases de frameworks de desarrollo de *Swing*[2].

La descripción detallada de la jerarquía de **Computer_Interface_Object** añadida para la actividad *Abrir Archivo* se muestra en la Tabla A.8 del apéndice.

[2]La jerarquía de objetos **Widget** es muy extensa. Por ello, sólo se presentan en la Figura 6.2 las subclases correspondientes a los objetos principales que maneja el estudiante en la navegación a través de una interfaz gráfica

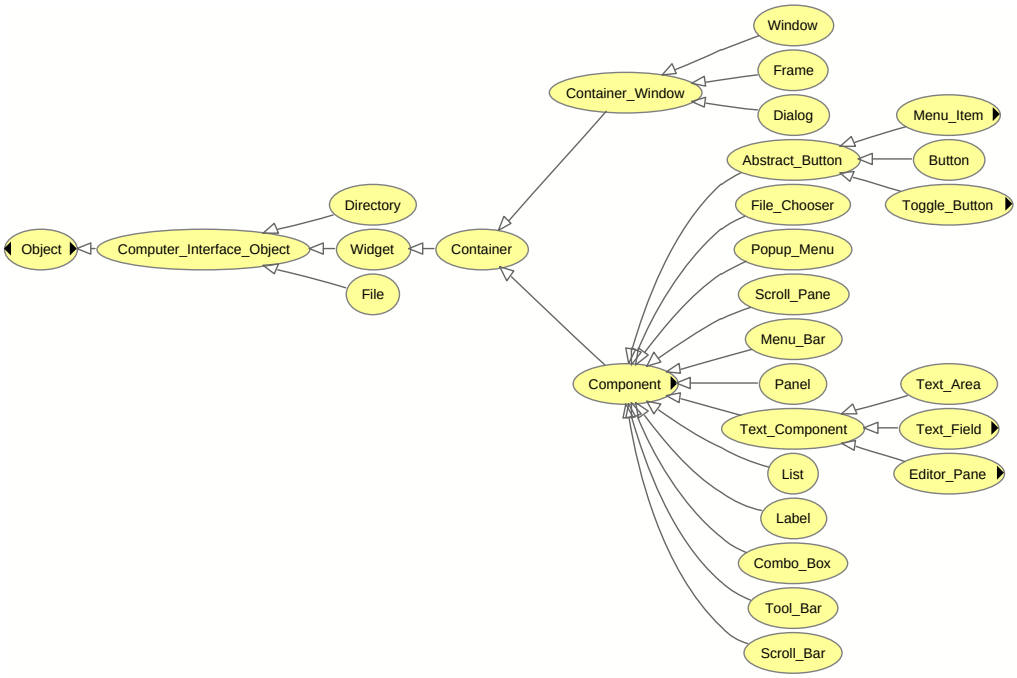


**Figura 6.2:** Jerarquía de **Computer_Interface_Object** para la actividad *Abrir Archivo* en la ontología del ME

En este tipo de entornos es habitual tener que realizar una determinada acción tal como .Arrastrar el ratón"de una posición inicial a una final. Esta posición final puede ser relativa a un objeto o una posición absoluta. Por esta razón, se ha añadido a la jerarquía de **Position** (conocimiento estructural), las subclases asociadas a estas dos formas de expresar la posición y además, a **To_Be_In_Position** (conocimiento pr), como condición en el estado de ejecución de una acción, dos subclases que indican el estar en una determinada posición absoluta o relativa un cierto objeto. Asimismo, como otro tipo de objeto de conocimiento a añadir, es preciso considerar ciertos tipos de relaciones existentes entre objetos involucrados en la actividad "Abrir Archivo" o similares. En la Figura 6.3 se muestra la subjerarquía de relaciones (conocimiento estructural) añadida a la ontología del ME para una actividad del tipo "Abrir Archivo" y en la tabla A.9 del apéndice, la descripción de las subclases asociadas.

Las relaciones **Is_Father_Of** y **Is_Son_Of** establecerían, respectivamente, la correspondencia entre un cierto directorio padre y cada uno de sus subdirectorios/archivos hijos. **Aggregation**, establecería la relación de añadidura de un objeto en otro como por ejemplo, la anexión de la instancia *Abrir* de **Menu_Item** en la instancia *Archivo* de la clase **Menu** (ver tabla de anexos A.8). La relación **Activated** o **Inhibited** establecerían, respectivamente, si un determinado objeto está habilitado o no en un momento de la navegación (por ejemplo, en el menú *Archivo*, la opción *Abrir* está activa -habilitada). La relación **Navigate_To** establecería si está permitido el acceso desde un objeto a otro de la interfaz (por ejemplo, el acceso a la **Dialog** *Abrir* -o similar, según la aplicación, para seleccionar el archivo a abrir a partir de la **Menu_Item** *Abrir*. Finalmente, otra relación importante en este tipo de actividad sería **Selection** que establecería, por ejemplo, si la instancia *Abrir* de la clase **Menu_Item** está seleccionada o no, si el botón *Abrir* (instancia de **Button**) está seleccionado o no, etc.

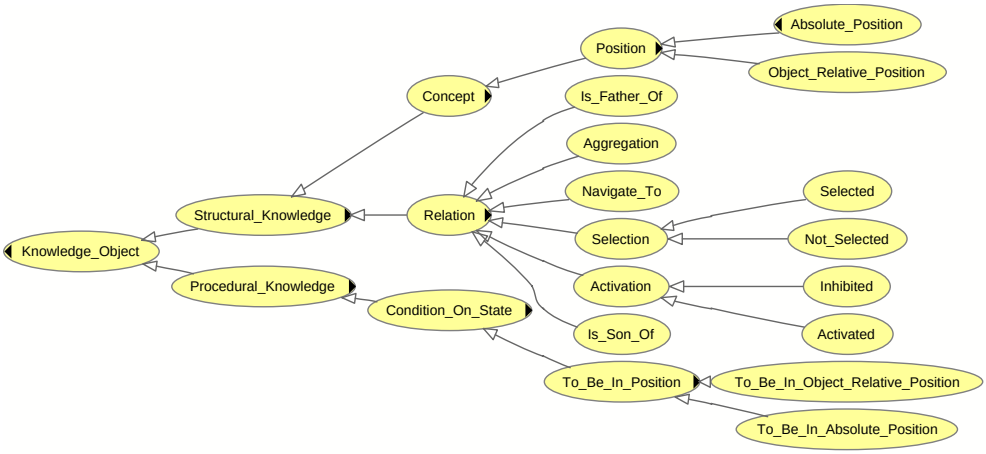


**Figura 6.3:** Otras jerarquías añadidas a **Knowledge_Object** en la ontología del ME

Como se describió en el apartado 5.2.5 (en la figura 5.9 y en la Tabla A.4 del apéndice), uno de los objetos de conocimiento procedimental es **Punctual_Action**. Este objeto representa cualquier tipo de operación realizada por el estudiante de acuerdo al entorno específico de aprendizaje. En el caso concreto de la actividad .Abrir Archivoz en el aprendizaje de un GUI en general, se realizan un conjunto de acciones de naturaleza bien diferente. Por ejemplo, surgen las acciones (ver Tabla 6.1) como *Arrastrar* y *Pulsar_Boton* (representadas en la ontología como **Drag_Object** y **Push**, respectivamente). La primera, implica el desplazamiento de un objeto y la segunda una acción que modifica el estado del objeto. Por lo tanto, la jerarquía asociada a **Punctual_Action** también ha sido ampliada para abordar este tipo de entornos de aprendizaje así como para entornos virtuales. Dada la fuerte interrelación entre las diferentes clases de acciones existentes en ambos tipos de entornos de aprendizaje, se muestra de forma completa la jerarquía de acciones para ambos en la descripción del siguiente prototipo correspondiente a un entorno virtual de entrenamiento.

### 6.1.2 Adaptación del método de diagnóstico para el prototipo de demostración I

El prototipo I de demostración ha servido de punto de partida para la creación de una extensa jerarquía de reglas de diagnóstico en el módulo de Diagnóstico Pedagógico a partir de las cuales se realizarán ampliaciones/moficaciones para otros prototipos.

- *Jerarquía de reglas definidas en el módulo de Diagnóstico Pedagógico*

  Se han incluido la mayoría de las categorías descritas en la sección 5.3.2 y que se enumeran a continuación:

  1. *Diagnóstico de acuerdo al tipo de acción que realiza el estudiante.* Se han distinguido las siguientes sub-categorías:
     - 1.1 *Diagnóstico de acuerdo a la aplicabilidad de una acción.*
     - 1.2 *Diagnóstico de acciones aplicadas de acuerdo a la efectividad de la ejecución.*
     - 1.3 *Diagnóstico de acciones aplicadas de acuerdo a su relevancia con respecto al plan.* Se han distinguido las siguientes sub-categorías:

1) *Diagnóstico de acuerdo a la coincidencia de su operador y/o argumentos con la siguiente acción en el plan.*
2) *Diagnóstico de acuerdo a su posición en el plan.*

1.4 *Diagnóstico de acciones erróneas dependientes del dominio.*

2. Diagnóstico de las ejecuciones de la actividad.

2.1 *Diagnóstico de la ejecución de la actividad.*

2.2 *Diagnóstico de las trayectorias de la actividad.*
La navegación a través de este tipo de entornos 2D se ha considerado también como una trayectoria seguida a través de objetos gráficos. En estos entornos el estudiante, mediante la selección de ciertos objetos gráficos, también se puede considerar que "navega.ª través de estos objetos hasta llegar a una posición final (que puede ser un objeto gráfico en este tipo de entornos). Se han considerado también en este criterio dos sub-categorías:

1) *Diagnóstico del grado de calidad de la trayectoria.*
2) *Diagnóstico de acuerdo a las características de la trayectoria.*

2.3 *Diagnóstico del grado de maestría.*

3. *Diagnóstico basado en el número y tipo de preguntas formuladas por el estudiante.*
A pesar de que en este tipo de entornos el tipo posible de preguntas que el estudiante puede realizar son, normalmente, escasas, esta categoría ha sido creada para el prototipo I.

4. *Diagnóstico basado en las pistas e instrucciones proporcionadas por el tutor.*
De forma similar a las dos categorías previas, este criterio ha sido creado para entornos de aprendizaje de GUIs, aunque su mayor aplicación se presenta en el aprendizaje en entornos EVIEs, como aparece en los prototipos II y III descritos más adelante.

- *Patrones de reglas en el módulo de Diagnóstico Pedagógico*
Se han incluido los patrones para los criterios de diagnóstico previos (ver sección 5.3.2), teniendo en cuenta el tipo de acciones que se realizan en este tipo de entorno para el aprendizaje de GUIs (acciones de movimiento de un objeto en entornos 2D, como *Arrastrar un objeto de una posición inicial a una posición final*, acciones que implican modificación del estado del objeto, como por ejemplo, *Pulsar un botón*), etc.

Los patrones añadidos para realizar el diagnóstico pedagógico con este prototipo de demostración también han requerido que sus nuevos predicados se incluyeran en las tablas de predicados del Módulo de Diagnóstico Pedagógico (sección 5.3.2.6). Asimismo, el mapeo de estos predicados con la ontología del ME también ha obligado a incluir algunas clases y propiedades en la ontología ya adaptada para entornos de aprendizaje de GUIs (descrita en la sección 6.1.1).

Finalmente, las reglas del Módulo Gestor de Conflictos descritas en la sección 5.3.5 no han sido modificadas para su aplicación a este prototipo.

## 6.2 Modelado del estudiante para prototipo de demostración II: Aprender a programar una lavadora

El Modelado del Estudiante presentado hasta este punto es aplicable al aprendizaje de actividades referentes a un GUI pero puede también ser adaptado y ampliado, como se detalla a continuación, para su validez en Entornos Virtuales de aprendizaje y, en concreto, para Entornos Virtuales de Entrenamiento. Esto supone nuevos tipos de información a añadir al ME, entre los que destacan:

- El conocimiento intrínseco a este tipo de entornos, relacionado con su dominio espacial (existencia de escenarios, subescenarios y conexiones entre ellos), objetos específicos, tanto manipulables como no manipulables, que pueden hallarse en él, así como sus posiciones en el espacio tridimensional correspondiente.
- La consideración de nuevos tipos de acciones que puede realizar el estudiante en el nuevo entorno de simulación (mirar, hablar, interactuar con los objetos de distinta naturaleza incluidos en el escenario geométrico del sistema, moverse de una posición inicial a una posición final con o sin objeto, etc.).
- El conocimiento adicional referente a la trayectoria que puede seguir el estudiante durante su aprendizaje.

Para ejemplificar la aplicación del enfoque de diseño pedagógico planteado y el modelado del estudiante en Entornos Virtuales de Entrenamiento, se considera a continuación una cierta actividad, *Programar lavadora con detergente de lavado*. Supuestamente, esta actividad, simple pero ilustrativa, pertenece al diseño formativo de un curso: *Cómo aprender a programar una lavadora*, teniendo en cuenta los tipos de ropa, tipos de detergentes, etc.

En el estado inicial de la actividad "*Programar lavadora con detergente de lavado*"se supone lo siguiente:

- El alumno conoce a priori los conceptos propios del dominio: programación de una lavadora, detergentes y tipos, botón de comienzo de lavado, botón de temperatura, cajón de detergentes y partes de que consta, etc.
- El estado inicial del escenario es el siguiente: la lavadora tiene una prenda en su interior, la puerta de la lavadora está cerrada, los diferentes tipos de detergentes (de lavado, lejía y suavizante) están en una mesa (mesa de lavado), el cajón de detergentes consta de tres cubetas (cubetas de detergente de lavado, lejía y suavizante respectivamente) y el programa de la lavadora y la temperatura están establecidos y
- La meta de la actividad es colocar el detergente apropiado (detergente de lavado) en el lugar apropiado (cubeta de detergente de lavado) de la lavadora y el botón de inicio de lavado pulsado.

Tabla 6.4: Pasos del planificador para la actividad *Programar lavadora con detergente de lavado*

| PASOS | OPERADORES APLICADOS |
|---|---|
| **1.** Moverse el estudiante desde su posición inicial hasta el lugar de la lavadora | *mover(estudiante, p_inicial, p_destino)* |
| **2.** Abrir cajón de detergentes. | *abrir(estudiante, objx)* |
| **3.** Moverse el estudiante desde la posición de la lavadora al lugar de los detergentes (mesa de lavado). | *mover(estudiante, p_ini, p_dest)* |
| **4.** Coger detergente de lavado de la mesa de lavado. | *coger(estudiante, objx, p_ini)* |
| **5.** Trasladar el detergente de lavado de la mesa de lavado hasta el lugar de la lavadora. | *trasladar(estudiante, objx, p_ini, p_des)* |
| **6.** Echar detergente de lavado en la cubeta del detergente de lavado que forma parte del cajón de detergentes de la lavadora.). | *echar(estudiante, objx, objy)* |
| **7.** Cerrar cajón de detergentes. | *cerrar(estudiante, objx)* |
| **8.** Pulsar el botón de comienzo de lavado. | *pulsar(estudiante, objx)* |

Es importante señal que la posición de un objeto o de una persona en un entorno virtual tridimensional (*p_ini* y *p_des* en la Tabla 6.4), se puede especificar ahora como una posición absoluta (coordenadas tridimensionales), como una posición relativa a un objeto o como una posición relativa a un área o escenario del espacio geométrico en el que se encuentre. Esta nueva fuente de información característica de este tipo de entornos debe ser añadida, como se verá más adelante, en la ontología del estudiante.

En primer lugar, y de forma similar al prototipo I de demostración, se han definido en la tabla 6.5 los objetivos cognitivos (a nivel abstracto) para la actividad *Programar lavadora con detergente de lavado*. Esta especificación de objetivos va a ser un punto clave para realizar las adaptaciones (ampliaciones y/o modificaciones) necesarias en la ontología del estudiante permitiendo así su aplicación para Entornos Virtuales de Entrenamiento.

Tabla 6.5: Objetivos cognitivos de la actividad *Programar lavadora con detergente de lavado*

| CLASE DE OBJETIVOS | INSTANCIAS DE OBJETIVOS |
|---|---|
| ***Objetivos de conocimiento*** | ▪ Saber *reconocer* cada objeto del escenario: detergentes, lavadora, cajón de detergentes, botón de comienzo, etc.<br>▪ Saber dónde *está* cada objeto del escenario.<br>▪ Saber el *efecto de aplicar un determinado operador* a cada objeto (elemento) a usar para esta actividad (por ejemplo, cuando se pulsa el botón de inicio de la lavadora comienza el lavado de la ropa).<br>▪ Saber *qué es* cada objeto del escenario.<br>▪ Saber *para qué sirve* cada objeto del escenario.<br>▪ Saber *que existen* diferentes tipos de acciones posibles a realizar sobre los objetos en la actividad: *coger* un objeto (detergente), *abrir* o *cerrar* un objeto (cajón de detergentes en la lavadora), *echar* un objeto en otro (el detergente de lavado en la cubeta de detergente de lavado), *moverse* de una posición inicial a una posición final, *trasladar un objeto* (detergente de lavado) de una posición inicial a una posición final o *pulsar un objeto* (botón de comienzo de la lavadora).<br>▪ Saber *que es posible aplicar un determinado operador a uno o varios de los objetos* del escenario en la actividad.<br>▪ Saber *cómo realizar* las distintas acciones posibles en la actividad.<br>▪ Saber *que es posible relacionar dos objetos* del escenario mediante cada acción de relación (en este ejemplo, echar el detergente de lavado en la cubeta de detergente de lavado).<br>▪ Saber *cuándo se dan las condiciones suficientes para realizar la acción* de pulsar el botón de comienzo. |
| ***Objetivos de aplicación*** | ▪ Ser capaz de *realizar* las diferentes acciones para llevar a cabo la actividad.<br>▪ Ser capaz de *mantener la ubicación dentro del mapa mental* (escenarios y conexión entre ellos) durante el desplazamiento. |

Tabla 6.5: Objetivos cognitivos de la actividad *Programar lavadora con detergente de lavado* (cont.)

| CLASE DE OBJETIVOS | INSTANCIAS DE OBJETIVOS |
|---|---|
| ***Objetivos de análisis*** | ▪ Ser capaz de *seleccionar un objeto* de entre todos los objetos de una determinada clase para usarlo a continuación (por ejemplo, el detergente de lavado entre los tres tipos considerados: detergente de lavado, lejía y suavizante).<br>▪ Ser capaz de *clasificar un determinado objeto* dentro de una determinada categoría (por ejemplo, detergente de lavado en la catergoría de detergentes). |
| ***Objetivos de síntesis*** | ▪ Ser capaz de *construir un plan* para programar la lavadora con detergente de lavado.<br>▪ Ser capaz de *construir una ruta o camino válido* para realizar la actividad.<br>▪ Ser capaz de *construir una ruta o camino válido desde una posición inicial a una posición final* en la actividad. |
| ***Objetivos de evaluación*** | ▪ Ser capaz de *elegir una solución óptima*, con el menor número de pasos, que permita alcanzar el objetivo de esta actividad.<br>▪ Ser capaz de *elegir una trayectoria óptima*a que permita alcanzar el objetivo de esta actividad. |

### 6.2.1 Ontología del estudiante para prototipo de demostración II

La actividad *Programar lavadora con detergente de lavado* requiere, al igual que la actividad del prototipo de demostración I, realizar una extensión y especialización de la ontología general desarrollada para este entorno de aprendizaje y que se hace patente en las dos tables previas. Esta fuente de información permitirá enriquecer el proceso de diagnóstico cognitivo del estudiante y, en consecuencia, el proceso de enseñanza/aprendizaje. Esto supone, como se vió en el capítulo 3.3.6.1, aplicar el escenario 8 de construcción de ontologías de la metodología NeOn utilizada y, por lo tanto, realizar entre otras actividades una ampliación de las cuestiones de competencia. Un fragmento de estas cuestiones se presentan a continuación, del mismo modo que se especificaron en 5.2.2.3.

Tabla 6.6: Cuestiones de competencias específicas relacionadas con un *objeto de aprendizaje*

| IDENTIFICADOR | CUESTIÓN DE COMPETENCIA |
|---|---|
| **CC214.** | *¿De qué escenarios se compone un determinado mapa virtual?* |

Tabla 6.6: Cuestiones de competencias específicas relacionadas con un *objeto de aprendizaje* (cont.)

| IDENTIFICADOR | CUESTIÓN DE COMPETENCIA |
|---|---|
| **CC215.** | *¿Cuál es la posición origen y posición destino de la ubicación mental lograda por una persona en una cierta trayectoria?* |
| **CC216.** | *¿Cuál es la posición absoluta de una persona o de un cierto objeto virtual en un espacio geométrico?* |
| **CC217.** | *¿Cuál es la posición de una persona o de un cierto objeto virtual respecto a un subescenario (área) del espacio geométrico?* |
| **CC218.** | *¿De qué subescenarios se compone un determinado escenario?* |
| **CC219.** | *¿A qué escenarios está conectado un determinado escenario?* |
| **CC220.** | *¿Qué objetos geométricos contiene un cierto subescenario?* |
| **CC221.** | *¿A qué subescenarios está conectado un cierto subescenario?* |
| **CC222.** | *¿A qué subescenario pertenece un cierto objeto del espacio geométrico?* |
| **CC223.** | *¿Un cierto objeto del espacio geométrico es simple o compuesto?* |
| **CC224.** | *¿De qué objetos simples se compone un cierto objeto geométrico compuesto?* |
| **CC225.** | *¿Un cierto objeto forma parte de otro objeto?* |
| **CC226.** | *¿Qué orden específica está asociada a una instrucción dada por un tutor?* |
| **CC227.** | *¿A qué objeto(s) se aplica una acción en la que el estudiante interacciona con esos objetos?* |

Tabla 6.6: Cuestiones de competencias específicas relacionadas con un *objeto de aprendizaje* (cont.)

| IDENTIFICADOR | CUESTIÓN DE COMPETENCIA |
|---|---|
| **CC228.** | *¿A qué objeto(s) se aplica una acción en la que el estudiante interacciona con esos objetos modificando su estado (por ejemplo, abrir una puerta)?* |
| **CC229.** | *¿A qué objeto(s) se aplica una acción en la que el estudiante interacciona con esos objetos modificando la relación entre esos objetos (por ejemplo, echar un líquido de un objeto a otro)?* |
| **CC230.** | *¿A qué objeto(s) se aplica una acción en la que el estudiante interacciona con esos objetos modificando la relación entre el estudiante y esos objetos (por ejemplo, coger un objeto)?* |
| **CC231.** | *¿El grado de cumplimiento de las metas de una cierta acción es total o parcial?* |
| **CC232.** | *¿Cuál es la posición de destino en una acción de movimiento de un estudiante?* |

Tabla 6.7: Cuestiones de competencias simples relacionadas con la *traza de un estudiante*

| IDENTIFICADOR | CUESTIÓN DE COMPETENCIA |
|---|---|
| **CC233.** | *¿Cuál es el estado de trayectoria asociado a una ciertas traza de trayectoria de un estudiante?* |
| **CC234.** | *¿De qué trazas de posición consta una cierta traza de tramo de un estudiante?* |
| **CC235.** | *¿Cuál es el estado de tramo asociado a una cierta traza de tramo?* |
| **CC236.** | *¿De qué trazas de tramos se compone una cierta traza de trayectoria de un estudiante?* |
| **CC237.** | *¿Cuál es la traza de trayectoria asociada a una determinada traza de movimiento de un estudiante?* |

Tabla 6.8: Cuestiones de competencias simples relacionadas con el estado del *conocimiento del estudiante*

| IDENTIFICADOR | CUESTIÓN DE COMPETENCIA |
|---|---|
| **CC238.** | *¿De qué estado de trayectoria concreta se ha obtenido un cierto estado de ejecución de una actividad?* |
| **CC239.** | *¿Cuál es la distancia media acumulada recorrida por el estudiante en la trayectoria recorrida hasta el momento en la ejecución de una cierta actividad?* |
| **CC240.** | *¿A qué traza de trayectoria corresponde un cierto estado de trayectoria?* |
| **CC241.** | *¿Cuál es la valoración cualitativa del movimiento realizado por el estudiante durante la ejecución de una actividad?* |
| **CC242.** | *¿Cuál es la valoración cuantitativa del movimiento realizado por el estudiante durante la ejecución de una actividad (factor de trayectoria)?* |
| **CC243.** | *¿Cuál es la valoración cuantitativa del movimiento realizado por el estudiante durante la ejecución de una actividad (factor de trayectoria)?* |
| **CC244.** | *¿Cuál es la traza del tramo correspondiente a un cierto estado de tramo?* |
| **CC245.** | *¿Cuál es la distancia del trayecto realizado por el estudiante en un tramo?* |
| **CC246.** | *¿Cuál es el valor comparativo de la longitud del tramo realizado por el estudiante con respecto a la longitud del tramo óptimo?* |

Tabla 6.9: Cuestiones de competencias compuestas

| IDENTIFICADOR | TIPO DE INFORMACIÓN | CUESTIÓN DE COMPETENCIA |
|---|---|---|
| **CC247.** | **Traza del estudiante**<br>**Estado del estudiante** | *Dada una cierta traza de movimiento asociada a un desplazamiento del estudiante en una actividad, ¿Cuál es el estado de la trayectoria asociado a su traza de trayectoria?* |
| **CC248.** | **Traza del estudiante**<br>**Estado del estudiante** | *Dada una actividad finalizada por un estudiante, ¿Cuál es la valoración cuantitativa de la trayectoria seguida por él durante la ejecución de la actividad?* |
| **CC249.** | **Traza del estudiante**<br>**Objeto de aprendizaje** | *Dada una cierta traza de movimiento asociada a un desplazamiento del estudiante en una actividad, ¿Cuáles son las precondiciones de la acción puntual asociada a esa traza de movimiento?* |
| **CC250.** | **Traza del estudiante**<br>**Objeto de aprendizaje** | *Dada una cierta traza de tramo registrada para un determinado estudiante, ¿Cuál es la posición asociada a sus correspondientes trazas de posición?* |
| **CC251.** | **Perfil del estudiante**<br>**Traza del estudiante** | *Dada el identificador de un estudiante, ¿Cuáles son las trazas de movimiento asociadas a sus desplazamientos durante la sesión actual?* |
| **CC252.** | **Perfil del estudiante**<br>**Traza del estudiante**<br>**Objeto de aprendizaje** | *Dada el identificador de un estudiante, ¿Cuáles son las acciones de movimiento asociadas a sus desplazamientos durante la sesión actual* |
| **CC253.** | **Perfil de estudiante**<br>**Traza del estudiante**<br>**Estado de estudiante**<br>**Objeto de aprendizaje** | *Dado un determina estudiante, ¿Cuáles han sido las valoraciones obtenidas de las trayectorias realizadas por él para ir desde un punto inicial a un punto final?* |

A partir de las cuestiones de competencia obtenidas, se han extraído las nuevas clases a añadir a la ontología de ME. Las dos ontologías, *Student_Profile* y *Learning_Objective*, no requieren ninguna adaptación y, por lo tanto, son aplicables al tipo de entornos que se está analizando. Sin embargo, el resto de las subjerarquías sí precisan adaptaciones, en mayor o menor medida, tal y como se detalla a continuación.

Respecto a la ontología de objetos de conocimiento, las actividades realizadas en EVs tales como *Programar lavadora con detergente de lavado*, requieren la definición de nuevas clases de conocimientos, tanto estructurales como procedimentales. En primer lugar, se definen a continuación los conocimientos estructurales que se incluyen en la ontología como subclases de la clase **Concept**. En la figura 6.4 se muestra esta jerarquía añadida, y en la tabla A.10 del apéndice, la descripción detallada de sus subclases.

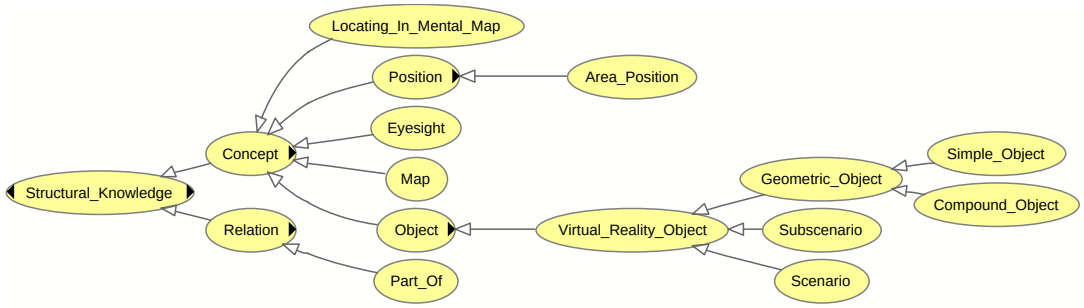


**Figura 6.4:** Jerarquía de conceptos virtuales añadida en la ontología del ME para Entornos Virtuales

Además de estos conceptos estructurales relativos a EVs, es necesario también añadir a la ontología **Knowledge_Object** del estudiante nuevos conceptos procedulares relativos a estos entornos. Aunque para un entorno de aprendizaje de GUIs, tal como el del prototipo I de demostración, se podrían haber incluido en este sentido conceptos procedimentales tales como **Way** o ruta de navegación a través de los elementos de la aplicación y **Trajectory** como la trayectoria de navigación, estos conceptos se incluyen en la ontología aquí por ser más significativos en este tipo de entornos virtuales. A continuación, se muestran en la Figura 6.5 las subclases añadidas al ME para **Procedural_Knowledge**. Su descripción detallada se presenta en la Tabla A.12 del apéndice. Por claridad, se muestra posteriormente, y de forma independiente, la extensa jerarquía añadida a **Punctual_Action**, a pesar de que es una subclase de **Procedural_Knowledge**.

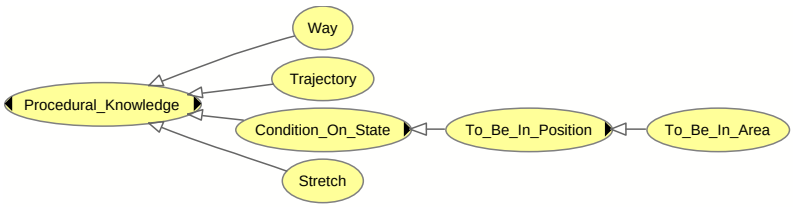


**Figura 6.5:** Jerarquía de **Procedural_Knowledge** añadida a la ontología del ME para EVs

Otra parte de la ontología **Knowledge_Object** del ME que requiere una extensa modificación es su jerarquía de **Punctual_Action**, conocimiento de tipo procedimental. Al igual que en los entornos de aprendizaje de GUIs, en los entornos virtuales un estudiante puede realizar distintos tipos de acciones, de naturaleza muy diversa. Por ejemplo, en el caso específico de la actividad *Programar lavadora con detergente de lavado*, tal y como se detalló en la tabla 6.4, el estudiante debe realizar acciones que implican la modificación de la posición del estudiante (por ejemplo, *mover*), en la interacción con un objeto, acciones que implican la modificación de la posición de ese objeto (por ejemplo, *trasladar* el detergente de una posición inicial a una posición final), acciones que implican la alteración del estado del objeto manipulado (por ejemplo, *abrir* el cajón de detergentes de la lavadora o *pulsar* el botón de comienzo de lavado), acciones que implican la modificación de la relación entre dos objetos (por ejemplo, *echar* el detergente en una de las cubetas del cajón de detergentes de la lavadora), etc. Se han considerado un conjunto bastante extenso de acciones típicas de este tipo de entornos más allá del propio escenario de la actividad analizada en el prototipo de demostración para EVs procedimentales. Dada la fuerte interrelación entre los diferentes tipos de acciones para los dos tipos entornos previos, se han representado aquí de forma completa pero fragmentada en diferentes figuras y tablas, las acciones puntuales requeridas para ambos tipos de entornos de aprendizaje, aprendizaje de GUIs y entornos virtuales de entrenamiento procedimental.

En primer lugar, se muestra en la figura 6.6 y en la tabla A.12 del apéndice, la jerarquía principal de **Punctual_Action**.

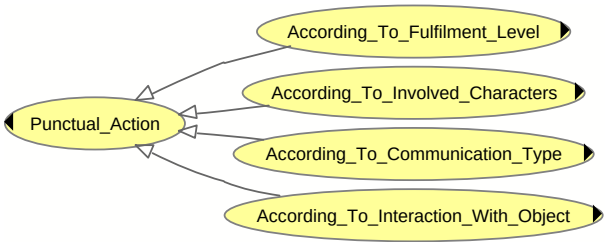


**Figura 6.6:** Jerarquía principal de acciones añadida a la ontología del ME para Entornos de Aprendizaje de GUIs y EVs

A continuación se van a ir describiendo cada una de las subclases de **Punctual_Action** presentadas en la figura y tabla anterior. Para cada una de ellas, del mismo modo, se mostrará la figura de su jerarquía y la tabla con la descripción de las subclases asociadas.

En primer lugar, se presenta la jerarquía de acciones según el tipo de comunicación, que se ha denominado en la ontología **According_To_Communication_Type** (Figura 6.7). En la tabla A.13 del apéndice, se muestra la descripción de sus subclases en la que se han obviado, al igual que en el resto de tablas, por simplicidad, la descripción de ciertas subclases que no intervienen en ninguno de los dos ejemplos de prototipos.

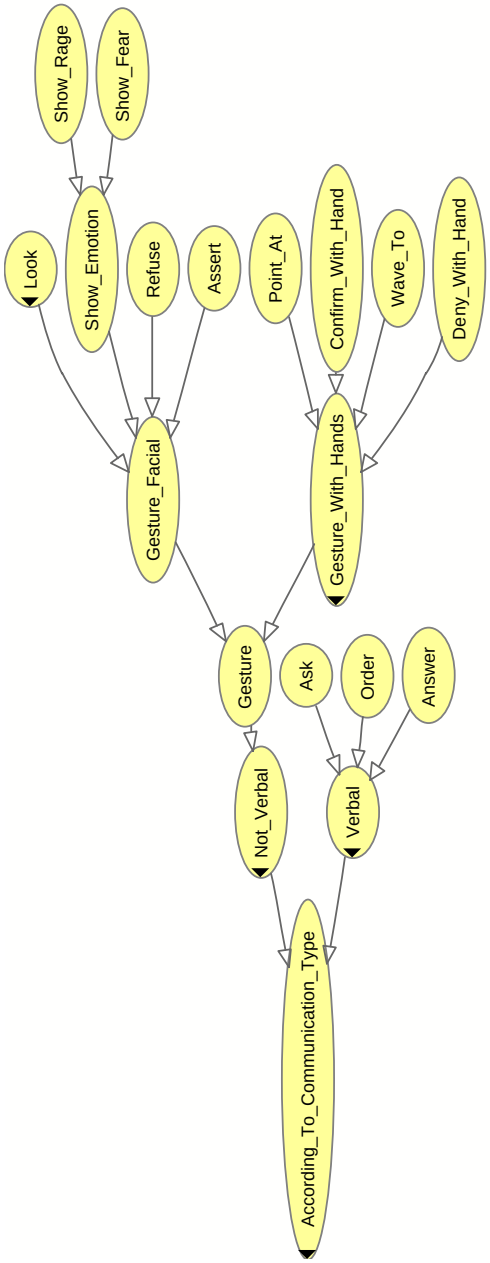


**Figura 6.7:** Jerarquía de acciones **According_To_Communication_Type** añadidas a la ontología del ME

La segunda jerarquía de acciones, que se ha denominado en la ontología **According_To_Interaction_With_Object**, corresponde a la clasificación de las acciones según la existencia o no de interacción del estudiante con objetos y el tipo de interacción, si existe. En la Figura 6.8 se muestra esta jerarquía y en la tabla A.14 del apéndice, su descripción detallada.

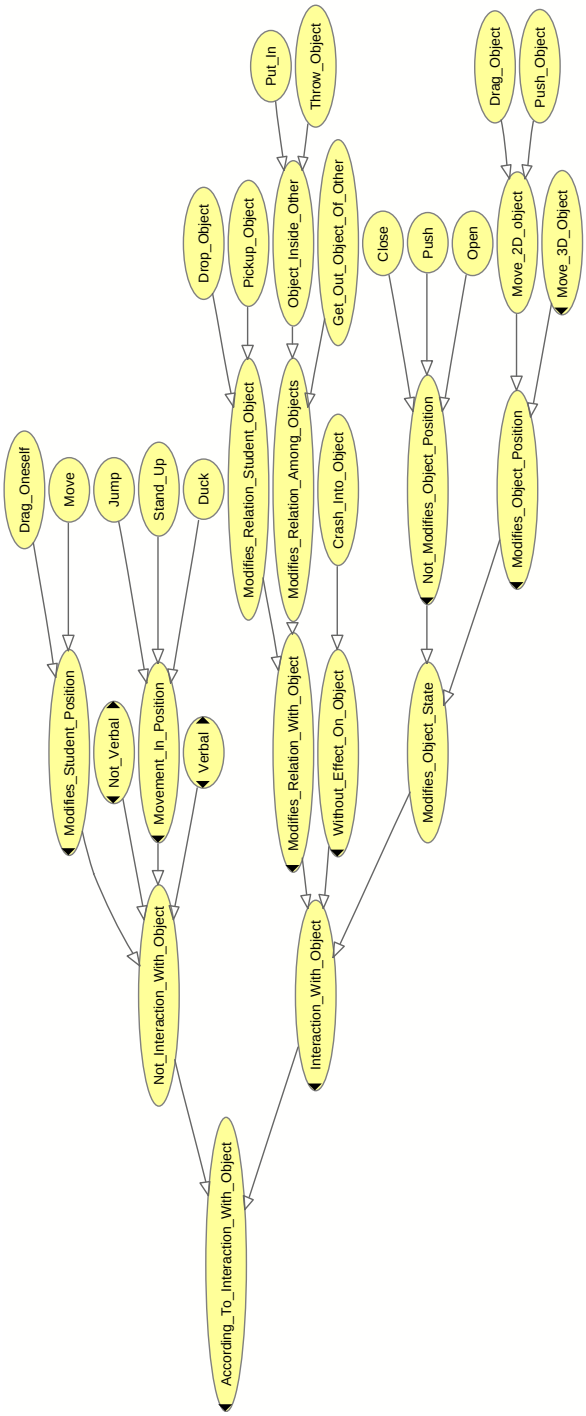


**Figura 6.8:** Jerarquía **According_To_Interaction_With_Object** en la ontología del ME

La tercera jerarquía de acciones, que se ha denominado en la ontología **According_To_Involved_Characters**, corresponde a la clasificación de las acciones según el número de personajes que participan en la ejecución de la acción. En la Figura 6.9 se muestra, igualmente, la jerarquía y en la tabla A.15 del apéndice su descripción detallada.

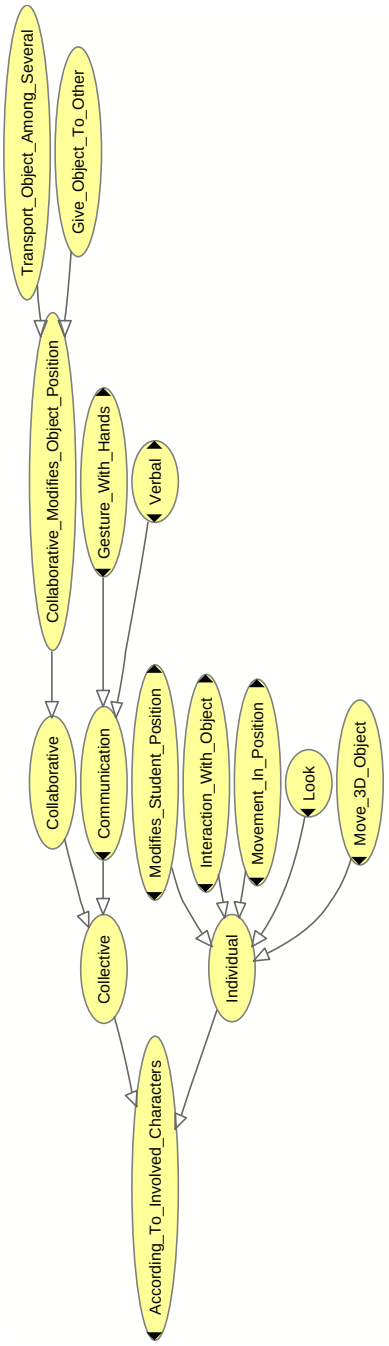


**Figura 6.9:** Jerarquía **According_To_Involved_Characters** en la ontología del ME

Respecto a la modificación de la jerarquía de acciones, finalmente se muestra la jerarquía de acciones denominada **According_To_Fulfilment_Level**.

En un EV, un estudiante puede ejecutar acciones correctamente y, sin embargo, alguna de las metas asociadas a esas acciones no se alcancen completamente. Por ejemplo, al moverse el

estudiante durante su aprendizaje por un espacio geométrico, el desplazamiento puede ser de mayor o menor calidad de acuerdo a la trayectoria seguida por el estudiante, de cuán buena sea respecto a la calidad de la trayectoria óptima. Otro ejemplo es en el aprendizaje de cirujía, al ejecutar acciones con instrumental cortante; el estudiante puede llegar a realizar la acción de cortar pero no realizarla con un cierto grado de inclinación o cogiendo el objeto cortante de forma adecuada y de ello dependa la calidad de la incisión. Por lo tanto, es importante en estos entornos representar estos tipos de acciones (acciones denominadas en este trabajo acciones con grado de cumplimiento parcial) para diagnosticar la calidad en la obtención de la meta de la acción por el estudiante y, en definitiva, otro aspecto importante de su estado de conocimiento. A diferencia de este tipo de acciones, la mayoría de las acciones implican que al ser ejecutadas por el estudiante se alcanzan sus metas o no, es decir, no hay grados de calidad en su diagnóstico (acciones que se denominan en este trabajo de grado de cumplimiento total de las metas). Por ejemplo, a este tipo pertenecen las acciones abrir, pulsar, etc. Al primer tipo se las denomina en el trabajo acciones con grado de cumplimiento parcial de las metas y a las segundas, acciones con grado de cumplimiento total de las metas.

La Figura 6.10 presenta las subclases de esta jerarquía de acciones y la descripción de sus subclases se muestra en la tabla A.16).

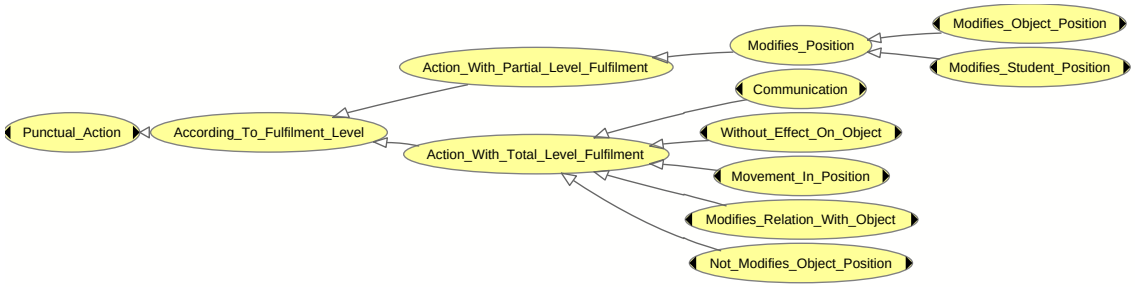


**Figura 6.10:** Jerarquía **According_To_Fulfilment_Level** en la ontología del ME

El resto de subclases que aparecen en la figura 6.10 no se han descrito puesto que son tipos de acciones ya mencionadas en las jerarquías previas.

El conocimiento adicional sobre la trayectoria del estudiante en los Entornos Virtuales de Entrenamiento conlleva la modificación de la jerarquía de la ontología **Student_State** que, como ya se describió, son datos obtenidos durante el desarrollo y tras finalizar un estudiante la actividad de enseñanza y que reflejan su estado en cada instante (estado pedagógico, estado de su comportamiento, estado emocional y de objetivos, alcanzados o no, etc.). en cada instante a varios . A continuación, se muestra junto con la jerarquía añadida al estado del estudiante (Figura 6.11), la descripción detallada de sus subclases en la tabla A.17.

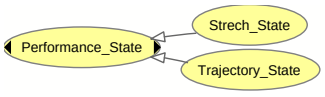


**Figura 6.11:** Jerarquía **Performance_State** añadida al ME

Del mismo modo, la jerarquía de la ontología **Student_Trace** debe ser ampliada para incluir información de la trayectoria seguida por el estudiante en su desplazamiento a través de los distintos escenarios. Este es un aspecto esencial de este tipo de entornos que, a través del diagnóstico, proporcionará al tutor mayor información para realizar una tutoría adecuada. La Figura 6.12 muestra la jerarquía añadida a la traza del estudiante y la tabla A.18 del apéndice detalla sus subclases.

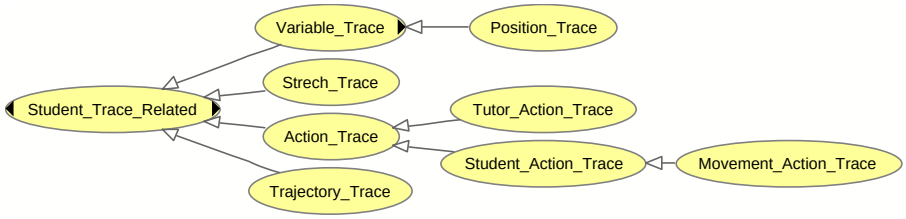


**Figura 6.12:** Jerarquía de **Student_Trace** añadida al ME

La jerarquía de la ontología **Student_Monitoring** debe también incluir ciertos parámetros relativos al seguimiento del estudiante en su desplazamiento a través de los distintos escenarios de los EVs. Esto permitirá también la adaptación de la ontología a este tipo de entornos de aprendizaje. En la Figura 6.13 se muestran las subclases añadidas y en la tabla A.19 del apéndice su descripción detallada.

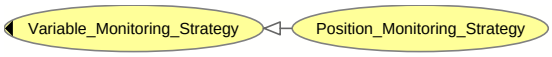


**Figura 6.13:** Jerarquía de **Student_Monitoring_Strategy** añadida al ME para EVs

### 6.2.2 Adaptación del método de diagnóstico para el prototipo de demostración II

El prototipo II de demostración ha requerido realizar modificaciones en el módulo de Diagnóstico Pedagógico. Las adaptaciones más importantes llevadas a cabo para tener en cuenta la interfaz inmersiva de los EVIEs en general y, en concreto, la de este prototipo son las siguientes:

- *Modificación de la jerarquía de reglas definidas en el módulo de Diagnóstico Pedagógico*

  1. En entornos EVIEs como el de este prototipo, el estudiante se representa como un avatar que está físicamente presente en el espacio virtual y puede moverse por él e interactuar con los objetos virtuales. El nuevo tipo de acciones de movimiento del estudiante ha dado lugar a la inclusión de un nuevo criterio de diagnóstico y, basado en él, en una nueva jerarquía de reglas: *diagnóstico de acciones aplicadas de acuerdo a la calidad de la ejecución*, es decir, diagnóstico de cómo las metas son alcanzadas. Este diagnóstico es interesante en acciones en las que el cumplimiento de las metas puede ser parcial, tales como la acción de movimiento, puesto que el estudiante puede moverse desde una posición inicial a la posición meta siguiendo un camino igual, más, o menos corto que la trayectoria óptima.

2. Otro criterio de diagnóstico muy relacionado con estos entornos es el *diagnóstico de las trayectorias de la actividad*. Esta jerarquía ya se incluyó en la adaptación del método para el prototipo I, ejemplo de entorno de aprendizaje de GUIs. La navegación a través de este tipo de entornos 2D se ha considerado también como una trayectoria a través de objetos gráficos. En estos entornos el estudiante también "navega.ª través de estos objetos hasta llegar a una posición final. A pesar de estas similitudes, las subcategorías de este criterio: el *diagnóstico del grado de calidad de la trayectoria*, y *diagnóstico de las características de la trayectorias* seguidas por el estudiante durante una actividad son características de los EVIEs.
3. Finalmente, otra jerarquía de reglas añadida ha sido el *diagnóstico basado en el comportamiento no verbal del estudiante*. Incluye reglas que permiten inferir los objetivos de aprendizaje supuestamente alcanzados o no basándose en el tiempo de inactividad del estudiante, dirección de la mirada, etc., en el EV.

- *Modificación/ampliación de los patrones de reglas en el módulo de Diagnóstico Pedagógico*

1. Se han incluido los patrones para los criterios de diagnóstico previos (ver sección 5.3.2).
2. Se han añadido nuevos patrones en el criterio de *diagnóstico de acciones aplicadas de acuerdo a la coincidencia de su operador y/o argumentos con la siguiente acción del plan*. Este criterio está estrechamente relacionado con los tipos de acciones y argumentos. En concreto, para EVIEs se han tenido que añadir patrones relativos a las nuevas acciones que surgen en este tipo de EVs: patrones para acciones que implican modificación de la relación entre el estudiante y el objeto con el que interactúa (por ejemplo, para la acción *Coger un objeto*), patrones para las acciones que implican modificación de la relación entre objetos (por ejemplo, *Echar un objeto en otro*), o patrones para acciones que se aplican a una posición como, por ejemplo, la acción de *Moverse una persona a una determinada posición*, o *Trasladar un objeto a una determinada posición* en el entorno 3D. Es importante señalar que, en estos últimos tipos de patrones, se ha tenido en cuenta la naturaleza de estos entornos (existencia de escenarios, subescenarios, y conexiones entre ellos, donde se encuentran los objetos virtuales).
3. El conjunto de patrones de reglas asociadas al criterio de *diagnóstico basado en el número y tipo de preguntas formuladas por el estudiante* se ha ampliado considerablemente. El tipo de preguntas planteadas por el estudiante relativas al EVIE puede ser muy variado (*¿para qué sirve este objeto?*, *¿dónde está el objeto X?*, *¿qué sucedería si yo hago esto?*, *¿porqué no puedo hacer esto?*, etc.), ha dado lugar a la incorporación de nuevos patrones de reglas en esta jerarquía de reglas de diagnóstico.
4. El conjunto de patrones de reglas asociadas al criterio de *diagnóstico basado en las pistas e instrucciones proporcionadas por el tutor* también se ha ampliado. Los nuevos tipos de pistas que el tutor puede proporcionar al estudiante en los EVIEs aportan diversa información sobre el estado de los conocimientos del estudiante tal como el estado del grado de conocimiento que el estudiante debería tener de los objetos existentes en el escenario 3D, de las posibles acciones, de la actividad, etc. Dada la importancia de esta información para diagnosticar el estado de los conocimientos del estudiante y, en definitiva, para la tutoría, se han incorporado un considerable número de patrones de reglas para este criterio.

No ha sido necesario modificar el conjunto de reglas específicas de diagnóstico para el entorno de aprendizaje concreto de este prototipo.

Los patrones añadidos para realizar el diagnóstico pedagógico con este prototipo de demostración, también han requerido que sus nuevos predicados se incluyeran en las tablas de predicados del Módulo de Diagnóstico Pedagógico (sección 5.3.2.6). Asimismo, el mapeo de estos predicados con la ontología del ME también ha obligado a incluir algunas clases y propiedades en la ontología ya adaptada para EVs (descrita en la sección 6.2.1).

Finalmente, las reglas del Módulo Gestor de Conflictos descritas en la sección 5.3.5 no se han modificado para su aplicación a este prototipo.

### 6.2.3 Pruebas con prototipo II

En este segundo escenario de prueba se ha considerado que los estudiantes están realizando el curso *Cómo programar una lavadora*.

En la sesión de prueba, comienzan la actividad 2: *Programar lavadora con detergente de lavado* perteneciente a la fase 0 del curso. La fase 0 del curso pretende que el estudiante conozca dónde están ubicados los elementos necesarios para realizar el proceso de lavado: cesto de ropa, lavadora, mandos de la lavadora, mesa de detergentes, etc. La característica fundamental de las actividades de esta fase es que no requieren ningún objetivo alcanzado previamente y pueden realizarse en cualquier orden.

La tabla 6.4, mostrada previamente, representa el plan de la solución. Esta tabla incluye las acciones que el estudiante debería llevar a cabo para concluir con éxito la actividad.

Antes de que los estudiantes hayan iniciado la sesión de aprendizaje (en MAEVIF), la ontología del estudiante almacena ya instancias sobre el perfil del estudiante, objetos de conocimiento implicados en el curso, objetivos de aprendizaje establecidos para cada actividad del curso, las dependencias entre los objetivos y los objetos de conocimiento, información sobre las sesiones (sesiones previas y sesión actual) incluyendo las trazas del estudiante (trazas de trayectorias, trazas de acciones ejecutadas, etc.), así como el estado acumulativo para cada estudiante (estados de objetivo, estado pedagógico, etc.).

Los objetivos asociados a la actividad han sido también definidos de acuerdo a la aproximación de diseño pedagógico en la que se basa el método (6.5).

El estado inicial para los objetivos en cada actividad de aprendizaje depende de varios factores tales como la estrategia de tutoría, la formación de los estudiantes, si los objetivos ya han sido alcanzados en actividades previas, etc. En los ejemplos que a continuación exponemos, se ha asumido el estado *desconocido* para los objetivos al comienzo de la actividad, si el objetivo no fue alcanzado en las actividades precedentes durante el aprendizaje del curso. Este estado asumido para cada objetivo, como ya se vio en el apartado 5.3.3, se informa al ATMS como nodos supuestos.

Los cinco ejemplos que se presentan en esta sección han sido seleccionados entre los casos de prueba que validan el método de diagnóstico pedagógico por la variedad de situaciones durante el aprendizaje de la actividad 2. Cada caso, se realiza por un estudiante distinto.

El primer ejemplo, representa una ejecución correcta de la actividad 2 por el primer estudiante. El resto de los ejemplos describen cómo los demás estudiantes comienzan la actividad 2 pero cometen diferentes tipos de error. Posteriormente, el Agente de Tutoría toma una cierta decisión, de acuerdo al diagnóstico del comportamiento del estudiante obtenido.

Para cada caso, se explica cómo se va actualizando la ontología del estudiante con cada acción ejecutada por el estudiante y cómo el ME obtiene ciertas conclusiones sobre el estado de

conocimiento del estudiante mediante las reglas del módulo de Diagnóstico Pedagógico y con el soporte de un ATMS y el módulo de Resolución de Conflictos.

#### 6.2.3.1 Caso de prueba 1: *El primer estudiante ejecuta la actividad completa sin ningún error.*

En este caso, los pasos o acciones del plan se han realizado correctamente por el estudiante. El escenario es el siguiente:

El estudiante está inicialmente en la puerta y debe moverse a donde está la lavadora, que es la primera acción correcta del plan (6.4).

Generalmente, como en este ejemplo, la ejecución de una acción por el alumno y su representación en la ontología (traza de ejecución de la acción) provoca que más de una regla del módulo de diagnóstico pedagógico se dispare. Sin embargo, por brevedad en la exposición de las pruebas, se enfoca aquí el análisis en una única acción (la primera del plan) y en una única regla disparada, la regla $R_1$ (5.3.2). El análisis es similar para cualquiera otra acción y regla.

Como ya se mencionó en la sección 3.3.5, para implementar el método se ha utilizado como soporte el *framework* Jena y su motor de inferencia con encadenamiento hacia adelante. A continuación se muestra un ejemplo (regla 6.1) de cómo se han traducido en Jena las reglas del módulo de diagnóstico pedagógico:

$$\begin{aligned} \mathrm{R}_1 : \ & \mathrm{SI} \ \ \mathrm{Intenta\_Aplicar(accx)} \rightarrow \\ & \quad \mathrm{Sabe\_Que\_Existe(accx)} \end{aligned} \tag{6.1}$$

La regla 6.2) es la representación en Jena y con primitivas builtin de la regla 6.1):

```
regla1.1:
          (?a d:associatedSpecificAction ?act), (?a d:associatedActionState ?st),
          (?stobj d:valuedObjective ?obj), (?obj rdf:type Knows_That),
          (?obj d:requiresKnowledgeObjects ?rel) ∧
          (?rel rdf:typeExist) ∧(?rel d:domain act) →
           Annadir_ME(?obj, true, ?stobj)
```
(6.2)

Donde `Annadir_ME` en el consecuente de la regla es un nuevo *builtin* implementado expresamente para el módulo de diagnóstico pedagógico. Su uso simplifica la expresión de las reglas en Jena. La acción que realiza este nuevo builtin dependerá del estado en la ontología del objetivo pasado como primer argumento al mismo: si está su propiedad *acquired* al valor del segundo argumento del builtin (*true* en este caso), se encarga de aumentar su nivel de fialibilidad (propiedad *levelCurrentReliability*) y si no, añade una nueva instancia del mismo con propiedad *levelCurrentReliability* a *0*. Además, si existiera en la ontología un estado para el objetivo con su propiedad *acquired* a valor *unknown*, eliminará este estado de la ontología. También, añade una nueva instancia de *Objective_Trace* con el estado alcanzado.

En la Figura 6.18, se muestra un fragmento del estado de la ontología involucrado en el disparo de la regla 6.1. En concreto, las instancias que se requieren para satisfacer el antecedente de la regla se han destacado en el dibujo con un fondo oscuro.

De acuerdo al consecuente de la regla 6.1, se supone que el objetivo *El estudiante sabe que existe la acción Moverse* es adquirido tras el disparo de la regla. Este objetivo es representado

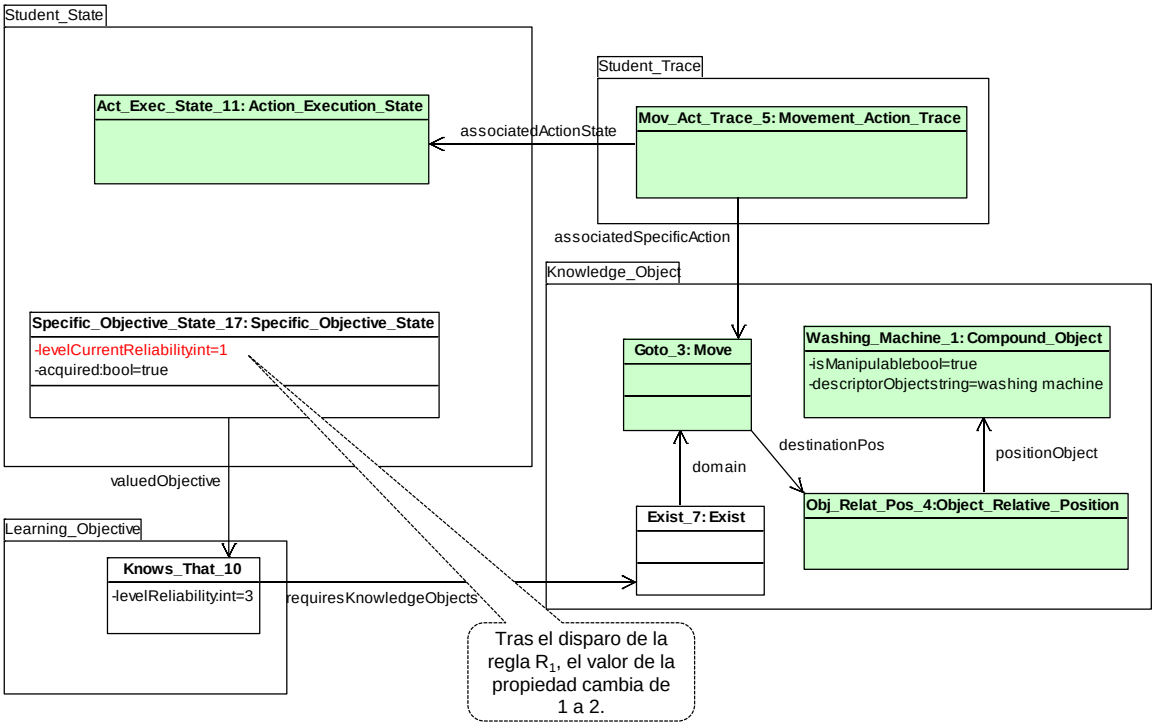


**Figura 6.14:** Estado de la ontología involucrado en el disparo de la regla $regla1.1$

en la ontología como una instancia de un tipo de objetivo cognitivo (*Knows_That*), y su estado se representa en la ontología por medio de una instancia de *Specific_Objective_State*, con la propiedad *acquired=true*.

El estudiante ya intentó aplicar esta acción en sesiones anteriores, realizando la actividad 0 del curso. Así pues, su estado ya se suposo adquirido en la ontología, añadiendo una instancia del estado del objetivo con propiedad *acquired=true*. Cuando se dispara la regla 6.1, la acción *Annadir_ME* en su consecuente provoca que el valor de la propiedad *levelCurrentReliability* de esta instancia, ya existente en la ontología, se incremente en 1. Los cambios en la ontología producidos por el disparo de esta regla se han reflejado en la figura 6.18.

El disparo de cualquier regla implica, además, se informa al ATMS mediante el siguiente nodo justificación, tal y como se vio en la sección 5.3.3:

$$\tilde{H}_i \wedge \varphi_i \Rightarrow \quad \text{Est\_Objetivo}(\text{obj}_i, \text{estado}_i) \tag{6.3}$$

donde:

$$\begin{aligned} \tilde{H}_i = &\quad H_1 \wedge ... \wedge H_m \; y \\ \varphi_i = &\quad \text{plausible}(r_i, \text{time\_exec}_{r_i}) \end{aligned} \tag{6.4}$$

De este modo, para la *regla10*, la entrada al ATMS $\tilde{H}_i$ son los hechos en formato de tripleta RDFS (sujeto, predicado, objeto) que satisfacen la regla. Por ejemplo:

$$H_1 = (\textit{Mov\_Act\_Trace\_8, d:associatedSpecificAction,Goto\_3}) \tag{6.5}$$

y

$$\varphi_i = \text{plausible}(R_1, \text{time\_exec}_{\text{regla1.1}}) \tag{6.6}$$

De manera similar, las restantes acciones del plan se ejecutan por el primer estudiante, y una traza del comportamiento del estudiante se registra en términos de la ontología del estudiante. Como resultado de este registro, algunas reglas se disparan, y el contenido de la ontología es actualizado de forma similar a como se ha explicado previamente.

En este primer escenario de prueba no existen contradicciones entre estados para el objetivo deducido por la regla 6.2. Por consiguiente, la regla del módulo de diagnóstico pedagógico que detecta la existencia de contradicciones entre estados de un mismo objetivo no llega a dispararse (sección 5.3.4).

#### 6.2.3.2 Caso de prueba 2: *El segundo estudiante ejecuta la actividad incorrectamente y se deducen objetivos no alcanzados pero sin existencia de contradicciones.*

El estudiante ha ejecutado la primera acción de movimiento del plan pero se ha movido hasta la posición de la mesa de los detergentes en vez de a la posición correcta, a la posición de la lavadora. En este caso, hay coincidencia de la acción ejecutada por el estudiante con la acción que corresponde ejecutar en el plan, pero no hay coincidencia entre la posición de destino alcanzada por el estudiante y la posición de destino de la siguiente acción del plan. A pesar de no existir esta coincidencia, sí que coinciden los subescenarios de ambas acciones. Como consecuencia, se disparará la regla 6.1 (caso de prueba 1) y otras del módulo de diagnóstico pedagógico como la siguiente (6.7):

$$\begin{aligned}
R_{23}: \text{SI } & \text{Aplicar_A_Pos}(accx, objy) \wedge \text{Sgte_Accion_En_Plan}(accx) \wedge \\
& \text{Pos_En_Sgte_Accion_Plan}(objy') \wedge \\
& \neg\text{Eq}(objy, objy')) \wedge \\
& \text{Eq}(\text{Subespacio}(objy), \text{Subespacio}(objy')) \rightarrow \\
& \text{Sabe}(\text{Subespacio}(objy')) \wedge \\
& \text{Es_Capaz_De_Construir}( \\
& \quad \text{Ubicacion_En_Mapa_Mental}( \\
& \quad\quad porigen, \text{Pos}(\text{Subespacio}(objy')))) \wedge \\
& \text{Es_Capaz_De_Construir}( \\
& \quad \text{Camino}(porigen, \text{Pos}(\text{Subespacio}(objy')))) \wedge \\
& \text{Sabe}(\text{Donde_Esta}(\text{Subespacio}(objy'))) \wedge \\
& \text{Sabe}(\text{Reconocer}(\text{Subespacio}(objy'))) \wedge \\
& \text{Sabe}(\text{Subespacio}(objy)) \wedge \\
& (\neg\text{Sabe}(\text{Reconocer}(objy')) \vee \\
& \neg\text{Sabe}(\text{Pos_En_Sgte_Accion_Plan}(objy')))
\end{aligned} \tag{6.7}$$

Como se puede observar, la regla 6.7 tiene en el consecuente la conjunción de varios objetivos alcanzados y una disyunción de objetivos no alcanzados. Esta disyunción se va a expresar como dos reglas en Jena que se dispararán cuando el antecedente de la regla 6.7 se cumpla como en el escenario de este caso de prueba. El estudiante puede no alcanzar ambos, alcanzar los dos o sólo uno de ellos. La corroboración de ello vendrá dada en el transcurso del aprendizaje, mediante el tratamiento de la no monotonía por el ATMS. Si la deducción de uno de esos objetivos no alcanzados da lugar a una contradicción posteriormente, el módulo de Resolución de Conflictos retraerá el supuesto asociado a él, esto es, el disparo de dicha regla, informando de ello al ATMS. Además, reforzará la creencia sobre la otra conclusión (segundo término de la disyunción).

El resto de los términos de la conjunción en el consecuente de la regla darán lugar también a una regla independiente de tal modo que, expresada en Jena, la regla 6.7 se desdobla en tantas reglas como consecuentes aparecen en la regla (de la regla23.1 a la regla23.8). Como ejemplo de alguna de estas reglas cuyo consecuente es un objetivo no alcanzado, se presenta a continuación la regla 6.8 que tienen como antecedente, el antecedente de la regla 6.7, y como consecuente, la última condición:

```
regla23.8:
            (?a d:associatedSpecificAction ?act), (?a d:associatedActionState ?st),
            (?st d:wasApplied true), (?act d:destinationPos ?posdest),
            (?st d:isOfActivityExecutionState ?stactiv),
            (?stactiv d:nextActionPlan ?actplan),
            (?actplan d:destinationPos ?posdestplan), notEqual(?posdest, ?posdestplan),
            (?posdest d:belongsToSubscenario ?subesc),
            (?posdestplan d:belongsToSubscenario ?subesc),
            (?stobj d:valuedObject ?obj), (?obj rdf:type Knows_That),
            (?obj d:requiresKnowledgeObjects ?reqprecond),
            (?reqprecond d:domain ?actplan), (?reqprecond d:range ?precond),
            (?precond d:condOnState ?cond), (?cond d:propertyName ?destinationPos),
            (?cond d:propertyValue ?posdestplan) →
            Annadir_ME(?obj, false, ?stobj)
```

(6.8)

De este modo, enfocando el análisis en el disparo de esta regla 6.8, el objetivo inferido es *El/la estudiante no sabe la posición en la siguiente acción del plan (moverse a la posición de la lavadora* o, expresado de otra forma: *El estudiante no sabe que la siguiente acción en el plan, moverse, requiere como precondición que la posición destino es la posición de la lavadora.* Este objetivo no fue logrado porque el alumno realizó como primera actividad del curso la actividad 2 sin ejecutar previamente ninguna otra actividad de la fase 0. Así pues, su estado en la ontología tenía la propiedad acquired a valor *unknown* antes del disparo de 6.8. Por lo tanto, tras dispararse esta regla, la acción *Annadir_ME*, de acuerdo a su funcionalidad ya descrita en el caso de prueba 1, consulta el estado del objetivo no alcanzado en la ontología del ME, y realiza la actualización oportuna añadiendo un estado de este objetivo con propiedad *acquired=false* y su propiedad *levelCurrentReliability* a 1 en la ontología del estudiante.

El disparo de esta regla implica que se informa al ATMS mediante el nodo justificación correspondiente, de acuerdo a 6.3 y 6.4, de forma similar a como se explicó en la regla analizada en el caso de prueba1.

Al no existir contradicción entre estados para ningún objetivo, las reglas del módulo de diagnóstico pedagógico que detectan existencia de contradicción entre estados de objetivos no se disparan (sección 5.3.4).

La estrategia de tutoría decide que el objetivo no alcanzado no debe requerir pista puesto que el objeto que incluye, la lavadora, es demasiado sencillo. Sin embargo, se permite al estudiante nuevos intentos de ejecución de la acción. El estudiante reiteradamente vuelve a cometer el mismo error hasta alcanzar el umbral de número de intentos permitidos por la estrategia de tutoría y el estudiante debe abandonar la sesión. Sin embargo, el comportamiento a partir de este punto variará acorde a otras estrategias de tutoría que podrían aplicarse.

En todas las ejecuciones reiteradas y erróneas de la acción, el método se comporta de forma similar a como se ha explicado previamente; la propiedad *levelCurrentReliability* del estado del objetivo *El/la estudiante no sabe la posición en la siguiente acción del plan, moverse a donde está la lavadora* en la ontología del ME se incrementa en 1 después de cada ejecución sin existencia en ningún momento de contradicciones.

#### 6.2.3.3 Caso de prueba 3: *El tercer estudiante ejecuta la actividad incorrectamente, se deducen objetivos no alcanzados y contradicciones en el estado de objetivos causadas por cambios en la mente del estudiante.*

El tercer estudiante ejecuta correctamente la actividad hasta que tiene que trasladar el detergente de lavado de la mesa de detergentes a la lavadora para echarlo en la cubeta correspondiente. En vez de realizar esta acción, *el estudiante traslada el detergente a la puerta*. Como consecuencia, la regla 6.7 se dispara. Como se detalló en el caso de prueba 2, en realidad se disparan las 8 reglas asociadas a esta regla expresadas en Jena, desde la regla23.1 hasta la regla23.8, entre otras. Enfocando el análisis en la regla 6.8 disparada, se deduce el siguiente objetivo no alcanzado: *El estudiante no sabe la posición en la siguiente acción, moverse a la posición de la lavadora*. Antes del disparo de la regla 6.8, el estado de este objetivo en la ontología tenía la propiedad *acquired* a valor *unknown*, debido a que no se dedujo en ninguna actividad anterior. Después del disparo de la regla, la ontología es actualizada de forma similar al caso de prueba 2. Por lo tanto, existe en este instante en la ontología una instancia del estado del objetivo con la propiedad *acquired=false*.

En este caso, la estrategia de tutoría decide dar al estudiante una segunda oportunidad, proporcionándole una pista sobre una precondición asociada a la siguiente acción de acuerdo al plan: la posición de destino (posición de la lavadora). Esta pista proporcionada por el tutor, provoca el disparo de la regla siguiente (regla 6.9) del módulo de diagnóstico pedagógico:

$$\begin{aligned} R_{62}: \text{SI}\ & \text{Da\_Pista}(\text{Sgte\_Accion\_En\_Plan}(\text{accx})) \wedge \\ & \text{Da\_Pista}(\text{Requiere\_Precond}(\text{accx}, \text{precondy}) \rightarrow \\ & \text{Sabe}(\text{Requiere\_Precond}(\text{accx}, \text{precondy})) \end{aligned} \tag{6.9}$$

La regla previa se puede expresar en Jena de la forma siguiente:

```
regla62.1 :
        (?a d:associatedSpecificAction ?act), (?act rdf:type Order),
        (?est d:wasApplied true), (?act d:associatedActionState ?st),
        (?st d:isOfActivityExecutionState ?stactiv),
        (?stactiv d:nextActionPlan ?actplan),
        (?actplan d:destinationPos ?destplan), (?act d:orderObjects ?reqprec),
        (?reqprec rdf:type ?Requires_Precond), (?reqprec d:domain ?actplan),
        (?reqprec d:range ?precond), (?precond d:condOnState ?cond),
        (?cond d:propertyName ?destinationPos), (?cond d:propertyValue ?destplan),
        (?stobj d:valuedObject ?obj), (?obj rdf:type Knows_That),
        (?obj d:requiresKnowledgeObjects ?reqprec) →
        Annadir_ME(?obj, true, ?estobj)
```

(6.10)

El disparo de esta regla implica que se informa al ATMS mediante el nodo justificación correspondiente de acuerdo a 6.3 y 6.4.

El estado del objetivo implicado en la regla 6.10 antes de su disparo tenía la propiedad *acquired* a valor *false*, tal y como se vio previamente. Al dispararse esta regla, la acción *Annadir_ME* no localiza un estado de este objetivo con la propiedad *adquirido=true* y, por lo tanto, tal y como se ha implementado, añade un nuevo estado para este objetivo con la propiedad *acquired=true* y su propiedad *levelCurrentReliability=1*. Como se puede observar, este caso implica la existencia de una contradicción; existen dos instancias del mismo objetivo, una con propiedad *acquired=false* y la recientemente añadida con propiedad *acquired=true*. Esta contradicción debe detectarse y resolverse para que el Agente del Estudiante razone de forma no monótona cuando, como es el caso, las evidencias más recientes rechacen las conclusiones previamente obtenidas.

De este modo, la regla de detección de contradicciones (sección 5.3.4) se dispara, al existir una contradicción entre estados del mismo objetivo: *El estudiante sabe que la siguiente acción tiene como precondición la posición de la lavadora*. Esta contradicción se informa al ATMS en forma de la correspondiente justificación:

ATMS almacena en el registro *no-good* el entorno inconsistente correspondiente y el módulo de resolución de conflictos es invocado para analizar el tipo de contradicción y resolverla adecuadamente. La regla $R_{C1}$ del módulo Gestor de Conflictos (sección 5.3.5) deduce que es una *contradicción en el estado de objetivos causadas por cambios en la mente del estudiante*.

$$\begin{aligned} R_{C1}: \text{SI } & \text{Existe\_Contradiccion(objx)} \wedge \\ & \text{Estado\_Actual\_Objetivo(objx, estobjx)} \wedge \text{EsAlcanzado(estobjx)} \wedge \\ & \text{Se\_Obtiene\_De\_Pista(estobjx)} \rightarrow \\ & \text{Tipo\_Contradiccion(objx, cambio\_mente\_estudiante)} \end{aligned} \tag{6.11}$$

El módulo Gestor de Conflictos también es responsable de resolver las contradicciones. En el ejemplo actual se satisface el antecedente de su regla $R_{C8}$ (sección 5.3.5.2) y la acción asociada a su consecuente resuelve la contradicción manteniendo el estado del objetivo más reciente (con

propiedad *acquired* a *true*), y eliminando el estado del objetivo con propiedad *acquired* a *false*.

$$\begin{aligned} R_{C8}: \text{SI } & (\text{Tipo\_Contradiccion}(objx, \text{descuido}) \wedge \\ & \text{Estado\_Previo\_Objetivo}(objx, estobjy) \wedge \\ & \neg\text{Estado\_Inferido\_Por}(estobjy, \\ & \quad \text{cambio\_mente\_estudiante})) \vee \\ & \text{Tipo\_Contradiccion}(objx, \text{olvido}) \vee \\ & \text{Tipo\_Contradiccion}(objx, \text{cambio\_mente\_estudiante}) \rightarrow \\ & \text{Eliminar\_Estado}(\text{Estado\_Previo\_Objetivo}(objx, estobjy)) \end{aligned} \tag{6.12}$$

El ATMS chequea la consistencia del nuevo entorno obtenido por el módulo de Resolución de Conflictos. Este módulose encarga también de retraer los supuestos que deducen el objetivo no alcanzado (el disparo concreto de la regla 6.8 que estableció en su momento el estado del objetivo como no alcanzado, así como las inferencias realizadas a partir de él.

#### 6.2.3.4 Caso de prueba 4: *El cuarto estudiante ejecuta incorrectamente la actividad y se deducen contradicciones entre estados de objetivos por descuido del estudiante*

El estudiante ejecuta correctamente la actividad 2 hasta que *Echa el detergente de lavado en la cubeta de suavizante* en vez de echar el detergente de lavado en la cubeta de detergente de acuerdo al plan. Como en los casos anteriores, se registra en la ontología del ME una traza del comportamiento del estudiante (de la ejecución de esta acción) y más de una regla del módulo de diagnóstico pedagógico se vuelven a disparar. Entre ellas, la regla siguiente, 6.13 (sección 5.3.2), cuya expresión en Jena también se muestra:

$$\begin{aligned} R_{18}: \text{SIAplicar\_A\_Objeto}(accionx, objx) \wedge & \text{Obj\_En\_Sgte\_Accion}(objy) \wedge \\ & \neg\text{Eq}(objx, objy) \rightarrow \\ & \text{Annadir\_ME}(\text{Know}(\neg\text{Sabe}(\text{Objeto\_En\_Sgte\_Accion}(objy)))) \end{aligned} \tag{6.13}$$

La regla previa se puede expresar en Jena de la forma siguiente:

```
regla18.1:
        (?a d:associatedSpecificAction ?act), (?act d:associatedActionState ?st),
        (?st d:wasApplied true), (?st d:isOfActivityExecutionState ?stactiv),
        (?stactiv d:nextActionPlan ?actplan),
        (?act d:isAppliedToObjects ?lobjact),
        (?actplan d:isAppliedToObjects ?lobjactplan),
        listNotEqualElems(?lobject,?lobjactplan,?lnotequal),
        (?reqprec rdf:type ?Requires_Precond), (?reqprec d:domain ?actplan),
        (?reqprec d:range ?precond), (?precond d:condOnState ?cond),
        (?cond d:propertyName isAppliedToObjects), (?cond d:propertyValue ?lnotequal),
        (?stobj d:valuedObject ?obj), (?obj rdf:type Knows_That),
        (?obj d:requiresKnowledgeObjects ?reqprec) →
        Annadir_ME(?obj, false, ?estobj)
```

(6.14)

Como resultado del disparo de la regla anterior, se deduce que *El/la estudiante no sabe que la siguiente acción en el plan requiere como precondición la cubeta de detergente*. El estudiante ha realizado previamente la actividad 0 de la fase a la que pertenece la actividad 2 pero el objetivo no fue deducido. Por lo tanto, el estado de este objetivo antes del disparo de la regla 6.14 en la ontología tiene la propiedad *acquired* a valor *unknown* y tras el disparo, la acción *Annadir_ME* actualiza la ontología añadiendo una instancia de estado del objetivo anterior con propiedad *acquired* a valor *false*. El disparo de esta regla es informada al ATMS como una justificación, de acuerdo a 6.3 y 6.4.

La estrategia de tutoría decide dar una pista al estudiante y la regla 6.10, vista en el caso de prueba 3, se dispara. De forma similar a como se explicó en el caso de prueba precedente, se deduce que *El/la estudiante sabe que la siguiente acción en el plan requiere como precondición la cubeta de detergente* y se actualiza la ontología con un nuevo estado del objetivo con la propiedad *adquirido* a valor *true*. Así pues, se produce una contradicción causada por un cambio en la mente del estudiante que se resuelve por el modulo Gestor de Conflictos, como ya se vio en el caso de prueba anterior, manteniendo en la ontología el estado actual con la propiedad *adquirido* a valor *true*.

En este momento, aún con la pista proporcionada por el agente de tutoría, el estudiante vuelve a ejecutar la acción y, de nuevo, no llega a alcanzar el objetivo -en este momento, el estudiante echa el detergente de lavado en la cubeta de lejía. Consecuentemente, la regla 6.14 vuelve a dispararse y, como al comienzo de la prueba, se añade a la ontología el estado del objetivo anterior con propiedad *acquired* a valor *false* y *levelCurrentReliability* a valor *1*. De nuevo, como en el caso 3, hay dos estados del mismo objetivo, uno con propiedad *acquired* a valor *false* y otro con propiedad *acquired* a valor *true* en la ontología del estudiante. La regla de detección de contradicciones del módulo de diagnóstico pedagógico detecta la contradicción y el ATMS es informado, de manera similar al caso 3, con la justificación de la contradicción correspondiente.

El tipo de contradicción es detectada ahora por el modulo Gestor de Conflictos a través de la regla $R_{C4}$ que se instancia en este punto de la siguiente forma:

$$\begin{aligned} \mathrm{R_{C4}}: \text{SI } & \text{Existe\_Contradiccion(Sabe\_Que\_13)} \wedge \\ & \text{Estado\_Actual\_Objetivo(Sabe\_Que\_13, esty)} \wedge \neg\text{EsAlcanzado(esty)} \wedge \\ & \text{Estado\_Previo\_Inferido(Sabe\_Que\_13, cambio\_mente)} \wedge \\ & \text{Menor(Tiempo\_Adquisicion(esty, 20), tiempo\_limite)} \rightarrow \\ & \text{Annadir\_ME(Tipo\_Contradiccion(Sabe\_Que\_13, descuido))} \end{aligned} \tag{6.15}$$

donde, *Sabe_Que_13* es la instancia del objetivo cognitivo *Sabe_Que* con los dos estados contradictorios.

Por lo tanto, se supone que el tipo de contradicción es un descuido del alumno y la regla $R_{C9}$ de resolución de conflictos se dispara resolviendo la contradicción mediante la eliminación del estado actual con propiedad *acquired=false*. Por lo tanto, en la ontología se mantiene el estado del objetivo previo con propiedad *acquired=true*.

$$\begin{aligned} \mathrm{R_{C9}}: \text{SI } & \text{Tipo\_Contradiccion(objx, descuido)} \wedge \\ & \text{Estado\_Previo\_Objetivo(objx, estobjy)} \wedge \\ & \text{Estado\_Inferido\_Por(estobjy,} \\ & \quad \text{cambio\_mente\_estudiante)} \rightarrow \\ & \text{Eliminar\_Estado(Estado\_Objetivo\_Actual(objx, esty))} \end{aligned} \tag{6.16}$$

Finalmente, el ATMS chequea la consistencia del nuevo entorno obtenido por el modulo Gestor de Conflictos. Este módulo se encarga además de retraer los supuestos que deducían el estado del objetivo no alcanzado. En este caso, el último disparo de la regla 6.14, representada en ATMS en forma de justificación, así como las inferencias realizadas a partir de él.

#### 6.2.3.5 Caso de prueba 5: *El quinto estudiante ejecuta incorrectamente la actividad y se deducen contradicciones entre estados de objetivos por olvido del estudiante*

El quinto estudiante ha realizado en varias sesiones anteriores todas las actividades de la fase 0. Una vez en la fase 1, donde es obligatorio que el alumno haya alcanzado todos objetivos de la fase 0, al realizar algunas de sus acciones se comprueba que el estudiante no cumple todos los objetivos supuestamente alcanzados en la realización de la actividad 2 de la fase 0. Así pues, el alumno debe volver a realizar esta actividad de la fase 0.

En la ejecución de la actividad 2 durante la sesión actual, el alumno realiza correctamente todas las acciones hasta que tiene que Echar el detergente de lavado en la cubeta adecuada. Como en el ejemplo anterior, el estudiante realiza incorrectamente la actividad y *Echa el detergente de lavado en la cubeta de suavizante*.

También, como en los casos anteriores, se registra en la ontología del ME una traza del comportamiento del estudiante (de la ejecución de esta acción) y más de una regla del módulo de diagnóstico pedagógico se vuelven a disparar. Entre ellas, la regla 6.14 se vuelve a disparar.

Como resultado del disparo de la regla anterior, se deduce que *El/la estudiante no sabe que la siguiente acción en el plan requiere como precondición la cubeta de detergente*. El estudiante realizó la actividad 2 en una sesión previa y, en concreto, este objetivo fue ya alcanzado mediante una pista proporcionada por el tutor. Por lo tanto, el estado de este objetivo antes del disparo de la regla 6.14 en la ontología tiene la propiedad *acquired=true*. Después del disparo, la acción *Annadir_ME* actualiza la ontología añadiendo una instancia de estado del objetivo con propiedad *acquired=false*.

El disparo de la regla 6.14 es informada al ATMS como una justificación, de forma similar a los casos precedentes, y la regla de detección de contradicciones en el módulo de diagnóstico pedagógico se dispara.

Del mismo modo, como siempre que se produce una contradicción, es informado de ello al ATMS que almacena en su registro *no-good* el entorno inconsistente correspondiente.

El tipo de contradicción es detectada ahora por el modulo Gestor de Conflictos a través de la regla $R_{C3}$. Esta regla se instancia en su disparo de la siguiente forma:

$$
\begin{aligned}
\mathrm{R_{C3}}: \text{SI } & \text{Existe\_Contradiccion(Sabe\_Que\_11)} \wedge \\
& \text{Estado\_Actual\_Objetivo(Sabe\_Que\_23, Est\_Obj\_32)} \wedge \\
& \neg\text{EsAlcanzado( Est\_Obj\_32)} \wedge \\
& \text{Estado\_Inferido\_Por(Est\_Obj\_22,} \\
& \quad \text{cambio\_mente\_estudiante))} \wedge \\
& (\geq (\text{Tpo\_Fin\_Adq(Est\_Obj\_22, tadq)}, \text{tpo\_lim\_olvido}) \rightarrow \\
& \text{Tipo\_Contradiccion(Sabe\_Que\_11, olvido)}
\end{aligned}
\tag{6.17}
$$

donde: *Sabe_Que_11*, *Est_Obj_32*, y *Est_Obj_22* son las instancias que satisfacen el antecedente de la regla 6.17 y, por lo tanto, las que provocan su disparo. *tadq* es el tiempo en que se adquirió el estado previo con propiedad *acquired=true* que es mayor o igual que *tpo_lim_olvido* (parámetro

configurable). *tpo_lim_olvido* es el umbral en el dominio de aprendizaje actual a partir del cuál se considera una contradicción provocada por un olvido y no por un descuido.

De este modo, se deduce que es una contradicción en el estado del objetivo analizado causada por olvido del estudiante. Puesto que se trata de este tipo de contradicción, se satisface el antecedente de la regla 6.34 del módulo Gestor de Conflictos (véase caso de prueba 3). Al dispararse esta regla, la acción en su consecuente resuelve la contradicción manteniendo el estado del objetivo que ha dado lugar al olvido del estudiante, y el estado previo se elimina (estado del objetivo con propiedad *acquired=true*).

Nuevamente, ATMS verifica la consistencia del nuevo entorno obtenido por el módulo Gestor de Conflictos. De nuevo, este módulo retrae las inferencias que deducían el estado previo del objetivo, es decir, el disparo de la regla 6.14 que dedujo el estado del objetivo con propiedad *acquired=true*, así como las inferencias realizadas a partir de este disparo.

## 6.3 Modelado del estudiante para prototipo de demostración III: Aprendizaje en un laboratorio de química

El tercer prototipo de demostración al que se ha aplicado la solución propuesta pertenece al proceso de aprendizaje en un laboratorio de química. En este dominio, los EVIEs son especialmente deseables cuando las prácticas implican un alto riesgo derivado de la manipulación de sustancias peligrosas (sustancias explosivas tales como ácidos, sustancias contaminantes tales como pesticidas o diversos tipos de hidrocarburos, etc.). En concreto, a continuación se presenta la aplicación del modelo propuesto a una de las prácticas peligrosas mencionadas previamente: la *Preparación de una disolución de ácido sulfúrico al 5 %*, correspondiente a un curso inicial en una carrera de Química.

La Figura 6.15 representa el plan solución con las acciones que el estudiante debería llevar a cabo para concluir con éxito la actividad.

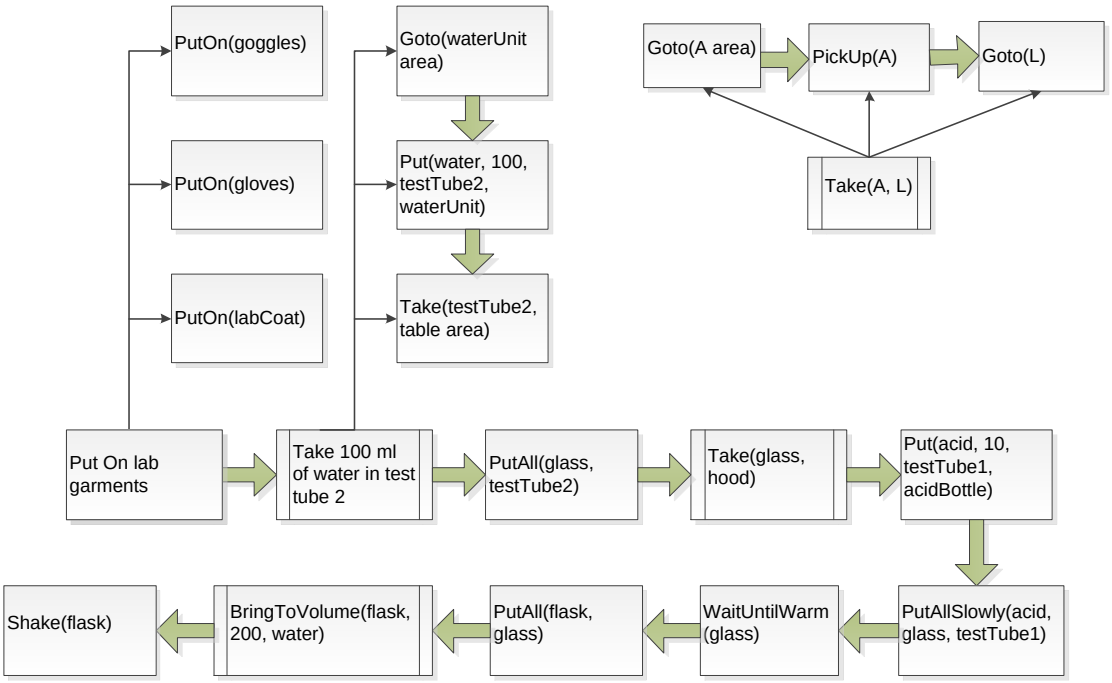


**Figura 6.15:** Plan de *Preparación de una disolución de ácido sulfúrico al 5 %*

En la Tabla 6.10 se describen más en detalle cada uno de los pasos de la solución de la Figura 6.15:

Tabla 6.10: Pasos del planificador para la actividad *Preparación de una disolución de ácido sulfúrico al 5%*

| PASOS | OPERADORES APLICADOS |
|---|---|
| **1.** Ponerse el equipo de laboratorio:<br>▪ Ponerse guantes.<br>▪ Ponerse gafas.<br>▪ Ponerse bata. | *PutOn(goggles)*<br>*PutOn(gloves)*<br>*PutOn(labCoat)* |
| **2.** Coger 100 ml de agua en la probeta 2.<br>**2.1.** Ir a coger agua.<br>**2.2.** Echar 100 ml. de agua en la probeta 2.<br>**2.3.** Llevar la probeta 2 a la mesa. | *Goto(waterUnit area)*<br>*Put(water, 100, testTube2, waterUnit)*<br>*Take(testTube2, table area)* |
| **3.** Echar el agua de la probeta 2 en el vaso de precipitado. | *PutAll(glass, testTube2)* |
| **4.** Llevar el vaso de precipitado a la campana extractora de gases.<br>**4.1.** Ir a por el vaso.<br>**4.2.** Coger el vaso de precipitado.<br>**4.3.** Ir a la campana extractora de gases. | *Goto(glass area)*<br>*PickUp(glass)*<br>*Goto(hoodUnit area)* |
| **5.** Echar 10 ml. de ácido sulfúrico en otra probeta (probeta 1). | *Put(acid, 10, testTube1, acidBottle)* |
| **6.** Echar lentamente el ácido de la probeta 1 en el agua del vaso de precipitado. | *PutAllSlowly(acid, glass, testTube1)* |
| **7.** Esperar a que el vaso de precipitado se enfríe. | *WaitUntilWarm(glass)* |
| **8.** Echar toda la disolución del vaso de precipitado al matraz. | *PutAll(flask, glass)* |

Tabla 6.10: Pasos del planificador para la actividad *Preparación de una disolución de ácido sulfúrico al 5 %* (cont.)

| PASOS | OPERADORES APLICADOS |
|---|---|
| **9.** Completar el volumen del matraz con agua.<br>**9.1.** Ir a coger agua.<br>**9.2.** Llenar el matraz (200 ml.) con agua.<br>**9.3.** Llevar el matraz a la mesa. | *Goto(waterUnit area)*<br>*Put(water, 200, flask, waterUnit)*<br>*Take(flask, table area)* |
| **10.** Agitar el matraz. | *Shake(flask)* |

Algunas de las acciones a realizar en la actividad son elementos del plan de tipo acciones compuestas en términos de la ontología de ME (**Compound_Action**), tal y como se vio en la sección A.1. Por ejemplo, el paso 2: *Take 100 ml of water in test tube 2*, y el paso 4: *Take(glass, hood)*, son acciones compuestas del plan de tipo bloque secuencia (**Sequence_Block**), es decir, ambas constan de acciones que deben realizarse en un orden determinado (ver Tabla 6.10). Sin embargo, el paso 1: *Put On lab garments*, es una acción compuesta de la clase **Unordered_Block** que consta, en este caso, de tres acciones que pueden realizarse en cualquier orden.

En primer lugar, y de forma similar a los prototipos I y II de demostración, se han definido en la tabla 6.11 los objetivos cognitivos (a nivel abstracto) para la actividad *Preparación de una disolución de ácido sulfúrico al 5 %*.

Tabla 6.11: Objetivos cognitivos de la actividad *Preparación de una disolución de ácido sulfúrico al 5 %*

| CLASE DE OBJETIVOS | INSTANCIAS DE OBJETIVOS |
|---|---|
| ***Objetivos de conocimiento*** | ▪ Saber *reconocer* cada objeto del escenario: armario del laboratorio, mesa, matraz, probeta, vaso de precipitado, campana extractora de gases, etc.<br>▪ Saber *qué es* cada objeto del escenario.<br>▪ Saber *que existen* diferentes tipos de acciones posibles a realizar sobre los objetos en la actividad: *PutOn,* ponerse una prenda (del equipo de laboratorio). *Goto,* ir a una determinada posición del escenario del laboratorio. *Put,* Echar algo (líquido) en un recipiente. *PutAll,* echar todo el contenido de un recipiente en otro (por ejemplo, de una probeta en un vaso de precipitado). *PutAllSlowly,* echar todo el contenido de un recipiente en otro lentamente. *Goto,* ir de una posición inicial a una posición final. *Take,* llevar un objeto (por ejemplo, una probeta) de una posición inicial a una posición final. *PickUp,* coger un objeto de una determinada posición). *WaitUntilWarm,* esperar hasta que un objeto se enfríe, por ejemplo, un vaso de precipitado. *Shake,* agitar un objeto, por ejemplo, un matraz. |

Tabla 6.11: Objetivos cognitivos de la actividad *Preparación de una disolución de ácido sulfúrico al 5%* (cont.)

| CLASE DE OBJETIVOS | INSTANCIAS DE OBJETIVOS |
|---|---|
| ***Objetivos de conocimiento*** | ▪ Saber *para qué sirve* cada objeto del escenario.<br>▪ Saber dónde *está* cada objeto del escenario.<br>▪ Saber el *efecto de aplicar un determinado operador* a cada objeto (elemento) a usar para esta actividad (por ejemplo, cuando echa el ácido sulfúrico en un vaso con agua se produce una reacción exotérmica).<br>▪ Saber *que es posible aplicar un determinado operador a uno o varios de los objetos* del escenario en la actividad.<br>▪ Saber *cómo realizar* las distintas acciones posibles en la actividad.<br>▪ Saber *que es posible relacionar dos objetos* del escenario mediante cada acción de relación (en este ejemplo, echar lentamente el ácido contenido en una probeta en un vaso de precipitado).<br>▪ Saber *que el orden de los argumentos al aplicar una acción es relevante*. Por ejemplo, se debe echar el ácido sulfúrico en un recipiente con agua pero nunca a la inversa porque el ácido puede salpicar y provocar quemaduras<br>.<br>▪ Saber *cuál es el orden correcto de los argumentos en la siguiente acción a realizar*<br>. |
| ***Objetivos de aplicación*** | ▪ Ser capaz de *realizar* las diferentes acciones para llevar a cabo la actividad.<br>▪ Ser capaz de *mantener la ubicación dentro del mapa mental* (escenarios y conexión entre ellos) durante el desplazamiento. |
| ***Objetivos de análisis*** | ▪ Ser capaz de *seleccionar un objeto* de entre varios objetos de diferentes clases existentes en una determinada posición para usarlo a continuación (por ejemplo, seleccionar un matraz de entre un conjunto de objetos: probetas, matraz, vasos de precipitado, etc.). |
| ***Objetivos de síntesis*** | ▪ Ser capaz de *construir un plan* para preparar una disolución de ácido sulfúrico en una cierta proporción (5%).<br>▪ Ser capaz de *construir una ruta o camino válido desde una posición inicial a una posición final* en la actividad. |

Tabla 6.11: Objetivos cognitivos de la actividad *Preparación de una disolución de ácido sulfúrico al 5 %* (cont.)

| CLASE DE OBJETIVOS | INSTANCIAS DE OBJETIVOS |
|---|---|
| ***Objetivos de evaluación*** | ▪ Ser capaz de *elegir una solución óptima*, con el menor número de pasos, que permita alcanzar el objetivo de esta actividad.<br>▪ Ser capaz de *elegir una trayectoria óptima*que permita alcanzar el objetivo de esta actividad. |

### 6.3.1 Adaptación de la Ontología del Estudiante para el prototipo de demostración III

La actividad *Preparar una disolución de ácido sulfúrico al 5 %* forma parte del aprendizaje en un laboratorio de química y se desarrolla en un Entorno Virtual de Entrenamiento. Gran parte del trabajo de adaptación de la ontología general de Modelado del Estudiante al aprendizaje de esta actividad particular ya se ha realizado para el prototipo II (*Aprender a programar una lavadora*) puesto que este prototipo también se lleva a cabo en un Entorno Virtual de Entrenamiento. Por lo tanto, ya se realizó una extensión de la ontología general de Modelado el Estudiante para su adaptación a este tipo de entornos (véase Apéndice A.3).

Así pues, el prototipo de demostración III requiere sólo ya especializar la ontología de Modelado del Estudiante para Entornos Virtuales de Entrenamiento para su aplicación al aprendizaje en un laboratorio de química, en concreto, para la práctica *Preparación de una disolución de ácido sulfúrico al 5 %*. De acuerdo a la metodología de aplicación del ME propuesta en este trabajo (sección 5.4), esto supone aplicar la metodología NeOn utilizada (escenario 8 de construcción de ontologías) y realizar, entre otras actividades, una ampliación de las cuestiones de competencia, similares a las mostradas en los dos prototipos anteriores.

A partir de las cuestiones de competencia obtenidas, se han extraído nuevas clases que se han añadido a la ontología de ME para Entornos Virtuales de Entrenamiento descrita en el Apéndice A.3.

Algunas de las clases principales añadidas a la jerarquía de objetos de conocimiento para el prototipo III se muestran en las figuras 6.16 y 6.17 respectivamente. Como se puede observar, ha sido necesario incluir objetos de conocimiento tanto de tipo estructural; objetos simples (*Flask*, *Test_Tube*, etc.) y compuestos (por ejemplo, *Hood*), así como objetos de conocimiento de tipo procedimental como, por ejemplo, nuevos tipos de acciones que implican sobre todo interacción con objetos y que no se tuvieron en cuenta previamente en la ontología general para Entornos Virtuales de Entrenamiento (*Put_On*, *PutAllSlowly*, *Shake*, etc.). Todos estos objetos son requeridos para desarrollar la práctica de la disolución de ácido sulfúrico.

Además, ha sido necesario añadir un nuevo tipo de objetivo de aprendizaje para la práctica a desarrollar. En el plan de esta actividad existen acciones como *Echar 10 ml. de ácido sulfúrico en probeta 1 (con agua)*, en las que el orden de los objetos a los que se aplica la acción es relevante (en este caso, aplicar la acción a los objetos en orden inverso, es decir, echar agua de la *probeta 1* en el recipiente del ácido sulfúrico, puede incluso ser peligroso al provocar proyecciones del ácido y, en consecuencia, posibles quemaduras al estudiante en la vida real). El objetivo añadido en la ontología **Learning_Objective** es de tipo *Objective_Knowledge* y se ha denominado *Knows_That_Arg_Order_Next_Action_Is*.

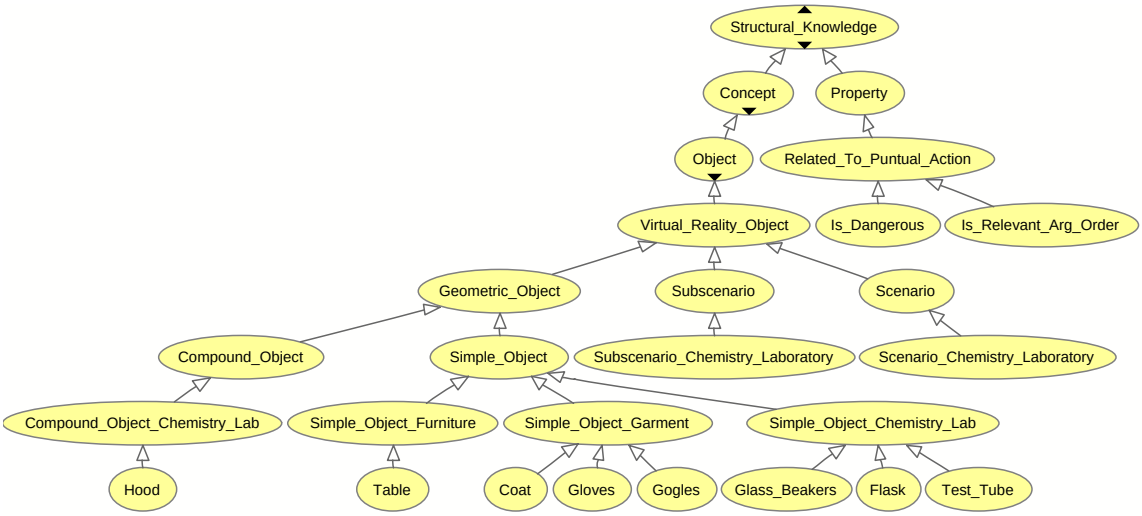


**Figura 6.16:** Clases añadidas a la jerarquía **Structural_Knowledge** del ME

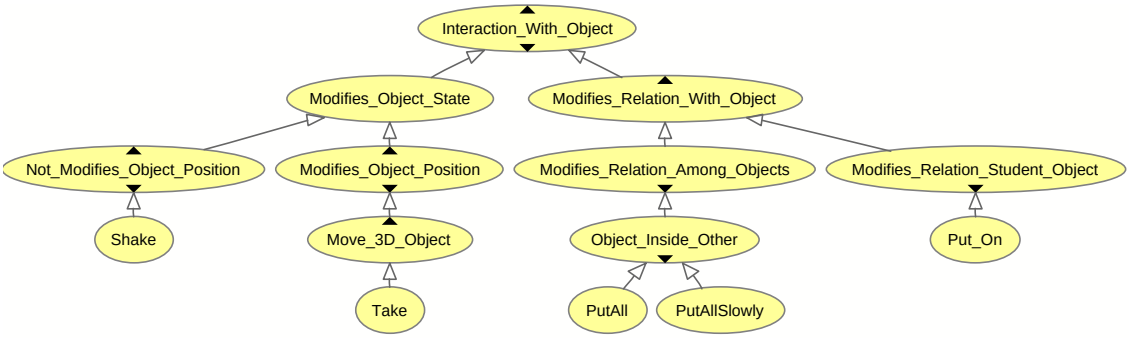


**Figura 6.17:** Clases añadidas a la jerarquía **Procedural_Knowledge** del ME

A continuación, como para cualquier actividad, es necesario instanciar la ontología de ME, ya adaptada. En este ejemplo, se deben añadir a la ontología los objetos específicos para el escenario del laboratorio de Química con los que se va a desarrollar la práctica, así como con las instancias de objetivos asociadas a esta actividad.

### 6.3.2 Adaptación del método de diagnóstico para el prototipo de demostración III

El prototipo III de demostración no requiere apenas modificaciones del método de diagnóstico pedagógico descrito en la sección 5.3. Para su adaptación al entorno de aprendizaje de la práctica *Preparar una disolución de ácido sulfúrico al 5 %* no ha sido necesario modificar o ampliar la jerarquía de reglas definida en el Módulo de Diagnóstico Pedagógico (sección 5.3.2). Sin embargo, sí ha sido necesario incluir un nuevo patrón de regla en la categoría de *Diagnóstico de acciones erróneas dependientes del dominio* originado por la existencia de acciones en las que el orden de los objetos a los que se aplica es relevante.

- Si el estudiante aplica una acción que crea una tupla de una relación entre varios objetos,

que implica exactamente el mismo operador y los mismos objetos que la siguiente acción en el plan, pero el orden de los objetos en la tupla es inadecuado, se puede asumir que él/ella no sabe que el orden de los objetos es relevante, y no sabe cuál es el orden correcto:

$$
\begin{aligned}
R_{30}: \text{SI } & \text{Aplicar}(accx) \wedge \\
& \text{Es\_De\_Tipo}(accx, \text{Modifies\_Relation\_Among\_Objects}) \wedge \\
& \text{Sgte\_Accion\_En\_Plan}(accx') \wedge \text{Operador}(accx, opx) \wedge \\
& \text{Operador}(accx', opx) \wedge \text{Eq}(\text{Args}(accx, lobjx), \text{Args}(accx', lobjx')) \wedge \\
& \neg\text{Eq}(\text{Orden\_Args}(lobjx), \text{Orden\_Args}(lobjx')) \rightarrow \\
& \neg\text{Sabe}(\text{Orden\_Args\_Es\_Relevante}(accx')) \wedge \\
& \neg\text{Sabe}(\text{Orden\_Args\_En\_Sgte\_Accion}(accx')) \wedge
\end{aligned}
\tag{6.18}
$$

La regla 6.25 añadida al Módulo de Diagnóstico Pedagógico ha requerido añadir sus nuevos predicados a las tablas de predicados del Módulo de Diagnóstico Pedagógico (sección 5.3.2.6). El mapeo de estos predicados con la ontología del ME también ha obligado a incluir un nuevo tipo de *Property*, *Is_Relevant_Arg_Order*), a la jerarquía de objetos estructurales (ver figura 6.16).

Las reglas del Módulo Gestor de Conflictos descritas en la sección 5.3.5 no han sido modificadas para su aplicación al prototipo de demostración III.

### 6.3.3 Pruebas con el prototipo III

Antes de que se el estudiante inicie la sesión de aprendizaje en MAEVIF, la ontología del estudiante ya almacena instancias relativas al diseño del curso: objetos de conocimiento implicados en el curso, objetivos de aprendizaje establecidos para cada actividad/práctica en el curso, las dependencias entre los objetivos y los objetos de conocimiento asociados. Además, almacena información sobre el estudiante: perfil del estudiante, información sobre sesiones previas incluyendo las trazas del estudiante trazas de trayectorias, trazas de acciones ejecutadas, etc., y el estado acumulativo para cada estudiante (estados de objetivos, estado pedagógico, etc.).

La práctica planteada: *Preparación de una disolución de ácido sulfúrico al 5 %*, se supone que es la actividad 1 del curso. El estado inicial para los objetivos en cada actividad de aprendizaje depende de varios factores tales como la estrategia de tutoría, la formación previa de los estudiantes, si los objetivos ya han sido alcanzados en actividades previas, etc. En los ejemplos que a continuación exponemos, se ha asumido el estado *unknown* para los objetivos al comienzo de la actividad, si los objetivos no han sido ya alcanzados en la actividad 0 previa. El estado asumido para cada objetivo, como se vio en el apartado 5.3.3, se informa al ATMS como nodos supuestos.

Los cinco ejemplos que se presentan en esta sección han sido seleccionados entre los casos de prueba que validan el método de diagnóstico pedagógico por la variedad de situaciones durante el aprendizaje de la actividad 2. Cada ejemplo de ejecución representa una situación o caso en el que está implicado un estudiante distinto. El primer ejemplo, representa una ejecución correcta de la actividad 1 por el primer estudiante y el resto, describen cómo los demás estudiantes comienzan la actividad 1 pero cometen diferentes tipos de error. Posteriormente, el Agente de Tutoría toma una cierta decisión de acuerdo al diagnóstico del comportamiento del estudiante obtenido.

Para cada caso, se explica cómo se va actualizando la ontología del ME con cada acción ejecutada por el estudiante, y cómo el ME obtiene ciertas conclusiones sobre el estado de conoci-

miento del estudiante por medio de las reglas del módulo de Diagnóstico Pedagógico y apoyándose en el ATMS y el módulo de Resolución de Conflictos.

#### 6.3.3.1 Caso de prueba 1: *El primer estudiante ejecuta la actividad sin ningún error.*

En este caso, el estudiante realiza correctamente los pasos o acciones del plan. Centrando el caso en una de las acciones, cuando el estudiante finaliza la acción compuesta *Take(glass, hood)* ejecutando la acción *Ir a la campana* (con el vaso de agua), que es la siguiente acción correcta de acuerdo al plan mostrado en la Figura 6.15), y como resultado de representar la ejecución de esta última acción en la ontología (traza de ejecución de la acción), la regla $R_{21}$ del módulo de diagnóstico pedagógico descrito en la sección 5.3.2), entre otras, se dispara.

$$
\begin{aligned}
R_{21}: \text{SI } & \text{Aplicar\_A\_Pos}(\text{accx}, \text{pfinal}) \wedge \text{Esta\_Situado\_En}(\text{porigen}) \wedge \\
& \text{Sgte\_Accion\_En\_Plan}(\text{accx}) \wedge \\
& \text{Pos\_En\_Sgte\_Accion\_Plan}(\text{pfinal}) \rightarrow \\
& \text{Sabe}(\text{Donde\_Esta}(\text{pfinal})) \wedge \\
& \text{Conoce}(\text{Subespacios\_Y\_Conexiones}(\text{porigen}, \text{pfinal})) \wedge \\
& \text{Es\_Capaz\_De\_Construir}( \\
& \quad \text{Ubicacion\_En\_Mapa\_Mental}(\text{porigen}, \text{pfinal})) \wedge \\
& \text{Es\_Capaz\_De\_Construir}(\text{Camino}(\text{porigen}, \text{pfinal}))
\end{aligned}
\tag{6.19}
$$

Por brevedad en la exposición de las pruebas, se enfoca aquí el análisis en una única acción *El estudiante va a la campana*, y en una única regla disparada, la regla $R_{21}$. El análisis es similar para cualquiera otra acción y regla.

Como ya se mencionó en la sección 5.3.2, para implementar el método se ha utilizado como soporte el *framework* Jena y su motor de inferencia con encadenamiento hacia adelante. A continuación se muestra la regla 5.20 y su representación en Jena y con primitivas *builtin* (regla 6.20):

Como se puede observar, la regla 6.19 tiene en el consecuente la conjunción de varios objetivos alcanzados. Cada uno de estos términos en la conjunción dará lugar a una regla independiente de tal modo que, expresada en Jena, la regla 6.19 se desdobla en tantas reglas como consecuentes aparecen en la regla (de la R21.1 a la R20.4). Como ejemplo de alguna de estas reglas cuyo consecuente es un objetivo alcanzado, a continuación se presenta la regla 6.20 que tienen como antecedente, el antecedente de la regla 6.19 y como consecuente, la última condición.

```
regla21.1:
            (?a d:associatedSpecificAction ?act), (?a d:associatedActionState ?st),
            (?st d:wasApplied true), (?act d:destinationPos ?posdest),
            (?st d:isOfActivityExecutionState ?stactiv),
            (?stactiv d:nextActionPlan ?actplan),
            (?actplan d:destinationPos ?posdest),
            (?stobj d:valuedObjective ?obj),
            (?obj rdf:type Knows_Where_Is),
            (?obj d:requiresKnowledgeObjects ?posdest) →
            Add_SM(?obj, true, ?stobj)
```

(6.20)

Donde `Add_SM` en el consecuente de la regla es un nuevo *builtin* implementado expresamente para el módulo de diagnóstico pedagógico. Su uso simplifica la expresión de las reglas en Jena. La acción que realiza este nuevo builtin dependerá del estado en la ontología del objetivo pasado como primer argumento al mismo: si está su propiedad *acquired* al valor del segundo argumento del builtin (*true* en este caso), se encarga de aumentar su nivel de fiabilidad (propiedad *levelCurrentReliability*) y si no, añade una nueva instancia del mismo con propiedad *levelCurrentReliability* a *0*. Además, si existiera en la ontología un estado para el objetivo con su propiedad *acquired* a valor *unknown*, eliminará este estado de la ontología. También añade una nueva instancia de *Objective_Trace* con el estado del objetivo alcanzado en este caso (estado con *acquired=true*).

En la Figura 6.18, se muestra un fragmento del estado de la ontología implicado en el disparo de la regla 6.20. En concreto, las instancias que se requieren para satisfacer el antecedente de la regla se han destacado en el dibujo con un fondo oscuro.

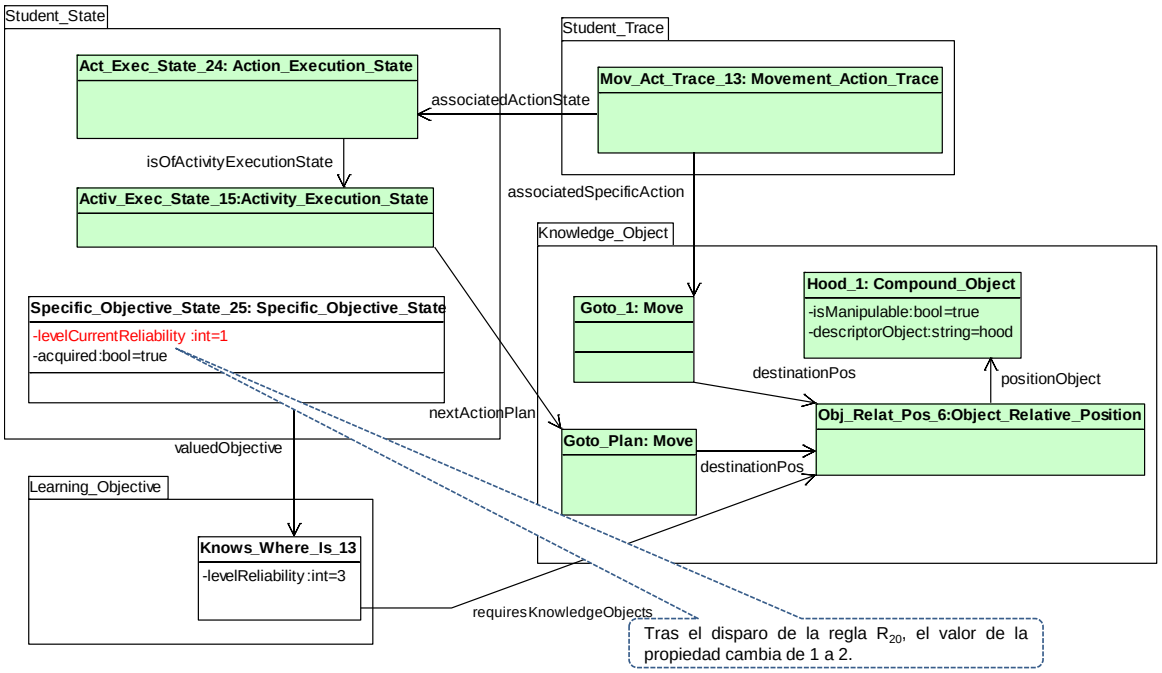


**Figura 6.18:** Estado de la ontología involucrado en el disparo de la regla $regla20.1$

De acuerdo al consecuente de la regla 6.20, se supone que el objetivo *El estudiante sabe donde está la posición final en la siguiente acción de movimiento* (posición de la campana) ha sido adquirido tras el disparo de la regla. Este objetivo es representado en la ontología como una instancia de un tipo de objetivo cognitivo (*Knows_Where_Is*), y su estado se representa en la ontología por medio de una instancia de *Specific_Objective_State*, con la propiedad *acquired=true*.

El estudiante ya intentó aplicar esta acción en sesiones anteriores, realizando la actividad 0 del curso en la que el estudiante comenzó a familiarizarse con el instrumental básico del laboratorio tal como la campana, probeta, matraz, etc. Así pues, su estado ya se asumía adquirido en la ontología, añadiendo una instancia del estado del objetivo con propiedad *acquired=true*.

Cuando se dispara la regla 6.20, la acción *Add_SM* en su consecuente provoca que el valor de la propiedad *levelCurrentReliability* de esta instancia, ya existente en la ontología, se incremente en 1. Los cambios en la ontología producidos por el disparo de esta regla se han reflejado en la figura 6.18.

El disparo de cualquier regla implica, además, que se informa al ATMS mediante los correspondientes nodos justificaciones, tal y como se vio en la sección 5.3.3:

$$\tilde{H}_i \wedge \varphi_i \Rightarrow \quad \text{Est\_Objetivo}(\text{obj}_i, \text{estado}_i) \tag{6.21}$$

donde:

$$\begin{aligned} \tilde{H}_i = &\quad H_1 \wedge ... \wedge H_m \ y \\ \varphi_i = &\quad \text{plausible}(r_i, \text{time\_exec}_{r_i}) \end{aligned} \tag{6.22}$$

De este modo, para la regla 6.20, la entrada al ATMS $\tilde{H}_i$ son los hechos en formato de tripleta RDFS (sujeto, predicado, objeto) que satisfacen la regla. Por ejemplo:

$$H_1 = (\textit{Mov\_Act\_Trace\_13, d:associatedSpecificAction,Goto\_1}) \tag{6.23}$$

y

$$\varphi_i = \text{plausible}(\text{regla21.1}, \text{time\_exec}_{\text{regla21.1}}) \tag{6.24}$$

De manera similar, las restantes acciones del plan se ejecutan por el primer estudiante; una traza del comportamiento del estudiante se registra en términos de la ontología del estudiante y, como resultado de este registro, algunas reglas se disparan y el contenido de la ontología es actualizado consecuentemente, de forma similar a como se ha explicado anteriormente.

En este primer escenario de prueba no existen contradicciones entre estados para el objetivo deducido por la regla 6.20. Por consiguiente, la regla del módulo de diagnóstico pedagógico que detecta la existencia de contradicciones entre estados de un mismo objetivo no llega a dispararse (sección 5.3.4).

#### 6.3.3.2 Caso de prueba 2: *El segundo estudiante ejecuta la actividad incorrectamente y se deducen objetivos no alcanzados pero sin existencia de contradicciones.*

El segundo estudiante ejecuta correctamente el plan de la actividad hasta la acción *Echar 10 ml. de ácido de la botella de ácido en la probeta 1*. Cuando el estudiante ejecuta la siguiente acción, en vez de echar el ácido de la probeta 1 en el vaso de agua, *echa el agua del vaso en la probeta 1*. Esta acción ejecutada no sólo es una acción errónea, sino también peligrosa puesto que el ácido se proyecta y puede producir quemaduras. Por lo tanto, la regla 6.25 del módulo de diagnóstico pedagógico, entre otras, se dispara:

$$\begin{aligned} R_{30}: \text{SI } & \text{Aplicar}(\text{accx}) \wedge \\ & \text{Es\_De\_Tipo}(\text{accx}, \text{Modifies\_Relation\_Among\_Objects}) \wedge \\ & \text{Sgte\_Accion\_En\_Plan}(\text{accx}') \wedge \text{Operador}(\text{accx}, \text{opx}) \wedge \\ & \text{Operador}(\text{accx}', \text{opx}) \wedge \text{Eq}(\text{Args}(\text{accx}, \text{lobjx}), \text{Args}(\text{accx}', \text{lobjx}')) \wedge \\ & \neg\text{Eq}(\text{Orden\_Args}(\text{lobjx}), \text{Orden\_Args}(\text{lobjx}')) \rightarrow \\ & \neg\text{Sabe}(\text{Orden\_Args\_Es\_Relevante}(\text{accx}')) \wedge \\ & \neg\text{Sabe}(\text{Orden\_Args\_En\_Sgte\_Accion}(\text{accx}')) \wedge \end{aligned} \tag{6.25}$$

Como se puede observar, la regla 6.25 tiene en el consecuente la conjunción de un conjunto de objetivos alcanzado. Por lo tanto, como ya se mencionó en el caso de prueba 1, cada uno de estos términos en la conjunción dará lugar a una regla independiente de tal modo que, expresada en Jena, la regla 6.25 se desdobla en tantas reglas como consecuentes aparecen en la regla (R30.1 y R30.2).

Enfocando el análisis en uno sólo de los objetivos, el primero, del consecuente de la regla 6.25, se infiere que *El/Ella no sabe que el orden de los objetos es relevante*. La regla expresada en Jena (regla 6.26) tiene como consecuente el objetivo anterior no alcanzado y, como antecedente, el antecedente de la regla 6.25:

```
regla30.1:
        (?a d:associatedSpecificAction ?act), (?a d:associatedActionState ?st),
        (?st d:wasApplied true), (?act rdf:type Modifies_Relation_Among_Objects),
        (?st d:isOfActivityExecutionState ?stactiv),
        (?stactiv d:nextActionPlan ?actplan), (?actplan d:operatorName ?op),
        (?act d:operatorName ?opact), equal(?op, ?opact),
        (?act rdfs:subClassOf ?class),
        (?actplan d:isAppliedToObjects ?lobjsactplan),
        (?act d:isAppliedToObjects ?lobjsact),
        listNotOrdEqual(?lobjsactplan, ?lobjsact), (?stobj d:valuedObjective ?obj),
        (?obj rdf:type Knows_That), (?obj d:requiresKnowledgeObjects ?objReq),
        (?objReq rdfs:subClassOf Is_Relevant_Arg_Order),
        (?objReq d:belongsTo ?class) →
        Annadir_ME(?obj, false, ?stobj)
```

(6.26)

Como se puede observar en la regla 6.26 aparece *listNotOrdEqual(?lista, listb)*, nuevo *builtin* desarrollado para verificar si dos listas de elementos, *lista* y *listb*, son iguales pero sus elementos no están en el mismo orden en ambas listas.

Dado que el objetivo *El/Ella no sabe que el orden de los objetos es relevante* no fue alcanzado en la actividad anterior, el estado del objetivo en la ontología era *acquired=unknown* antes del disparo de la regla 6.26. Sin embargo, al dispararse esta regla, la acción *Annadir_ME*, de acuerdo a su funcionalidad ya descrita en el caso de prueba 1, y consultando el estado del objetivo no alcanzado en la ontología del ME, realiza la actualización oportuna añadiendo un estado de este objetivo con propiedad *acquired* a valor *false* y su propiedad *levelCurrentReliability* a 1 en la ontología del estudiante.

El disparo de esta regla implica que se informa al ATMS mediante el nodo justificación expresado de acuerdo a 6.21 y 6.22, de forma similar a como se explicó previamente en la regla analizada en el caso de prueba1.

Al no existir contradicción entre estados para ningún objetivo, las reglas del módulo de diagnóstico pedagógico que detectan existencia de contradicción entre estados de objetivos no se disparan (sección 5.3.4).

Un EVIE, como MAEVIF, puede proporcionar al estudiante nuevos intentos de ejecución de la acción requerida sin suponer riesgos para el estudiante. Sin embargo, el objetivo analizado es de tal relevancia en la actividad que la estrategia de tutoría recomienda al estudiante finalizar la sesión actual y revisar los materiales de aprendizaje, considerando que el/ella no está mínimamente preparado para la práctica que está realizando. No obstante, otras estrategias de tutoría menos estrictas podrían también aplicarse en este punto.

#### 6.3.3.3 Caso de prueba 3: *El tercer estudiante ejecuta la actividad incorrectamente y se deducen objetivos no alcanzados pero sin existencia de contradicciones.*

El tercer estudiante ejecuta correctamente el plan de la actividad hasta la acción *Él/Ella echa lentamente el ácido de la probeta 1 en el vaso de agua*. Debido a las reacciones exotérmicas que se producen al ejecutar esta acción, el estudiante debería esperar a que se enfríe el vaso antes cogerlo. En vez de ello, el estudiante intenta inapropiadamente realizar la siguiente acción en el plan, *Echar el agua del vaso en el matraz*. Consecuentemente, la regla 6.27 del módulo de diagnóstico pedagógico, entre otras, se dispara:

$$\begin{aligned} R_5: \text{SI } & \text{Intenta\_Aplicar}(accx) \wedge \text{No\_Cumple}(accx, precondy) \wedge \\ & \text{Sgte\_Accion\_En\_Plan}(accy) \wedge \neg\text{Eq}(accx, accy) \rightarrow \\ & \neg\text{Sabe}(\text{Requiere\_Precond}(accx, precondy)) \wedge \\ & \neg\text{Sabe}(\text{Sgte\_Accion\_En\_Plan}(accy)) \end{aligned} \tag{6.27}$$

Como en los casos de prueba previos, cada uno de estos términos de la conjunción del consecuente de la regla anterior dará lugar a una regla independiente de tal modo que, expresada en Jena, la regla 6.27 se desdobla en tantas reglas como términos del consecuente (regla5.1 y regla 5.2, respectivamente). Enfocando el análisis en el primer objetivo, la regla correspondiente expresada en Jena es la siguiente (regla 6.28):

```
regla5.1:
        (?a d:associatedSpecificAction ?act), (?a d:associatedActionState ?st),
        (?st d:isOfActivityExecutionState ?stactiv),
        (?stactiv d:nextActionPlan ?actplan), notEqual(?actplan, ?act),
        (?stobj d:valuedObjective ?obj), (?obj rdf:type Knows_That),
        (?obj d:requiresKnowledgeObjects ?object),                          (6.28)
        (?object rdf:type Requires_Precond),
        (?stprecond d:isOfActionExecutionState st),
        (?stprecond d:isMet false), (?stprecond d:refersToCondition precondx),
        (?object d:domain ?act), (?object d:range ?precondx) →
        Annadir_ME(?obj, false, ?stobj)
```

Por lo tanto, del disparo de la regla 6.28 se infiere el objetivo *El estudiante no sabe que el operador (PutAll) requiere como precondición que la temperatura del objeto que se va a coger* (vaso de agua) *debe estar por debajo de un cierto umbral* (para que el estudiante no se queme). Nuevamente, antes de disparar la regla 5, el estado del objetivo anterior en la ontología era *acquired=unknown* por la misma razón que en el caso de prueba precedente. Así pues, tras el disparo de la regla, la ontología es actualizada con una instancia de estado de objetivo en la ontología con la propiedad *acquired=false* y su propiedad *levelCurrentReliability* a 1.

El disparo de esta regla implica que se informa al ATMS mediante el nodo justificación expresado de acuerdo a 6.21 y 6.22.

Al no existir contradicción entre estados para ningún objetivo, las reglas del módulo de diagnóstico pedagógico que detectan existencia de contradicción entre estados de objetivos no se disparan (sección 5.3.4).

El objetivo fallado en este ejemplo se considera menos relevante que el objetivo fallado en el ejemplo 2, y la estrategia de tutoría decide dar una segunda oportunidad al estudiante sin ninguna pista.

En el segundo intento, el estudiante intenta aplicar el operador *PutAll* de nuevo demasiado pronto. Por lo tanto, la regla 6.28, entre otras, se vuelve a disparar. El objetivo anterior ya tenía una instancia de estado de objetivo en la ontología con la propiedad *acquired=false*. Por consiguiente, la acción *Annadir_ME* en el consecuente de la regla 6.28 incrementa en 1 el valor de la propiedad *levelCurrentReliability* de la instancia de estado de objetivo ya existente. Una vez más, se informa a ATMS con el nodo justificación asociado al disparo de la regla.

El Agente de Tutoría decide entonces que el estudiante no ha preparado suficientemente bien esta práctica, y toma el control del resto de la actividad para demostrar y explicar al estudiante cómo realizar las acciones restantes.

#### 6.3.3.4 Caso de prueba 4: *El cuarto estudiante ejecuta la actividad incorrectamente, se deducen objetivos no alcanzados y contradicciones en el estado de objetivos causadas por descuido del estudiante.*

El cuarto estudiante es muy despistado. Él ejecuta correctamente el plan de la actividad hasta que ejecuta la acción *Echar toda la disolución del vaso de precipitado al matraz (PutAll(flask, glass))*. Después de esta acción, el estudiante agita el matraz antes de completar su volumen hasta 200 ml. con agua. Así pues, la regla 6.29, entre otras, se dispara, y se infieren los objetivos siguientes: *El estudiante sabe que el operador aplicado (Shake) está en el plan* y *El estudiante no sabe el siguiente operador en el plan (BringToVolume)*.

$$\begin{aligned} R_{28}: \text{SI } & \text{Aplicar}(accx) \wedge \text{Sgte\_Accion\_En\_Plan}(accy) \wedge \\ & \neg\text{Eq}(accx, accy) \wedge \text{Acc\_En\_Plan\_Post}(accx) \rightarrow \\ & \text{Sabe}(\text{Acc\_En\_Plan\_Post}(accx)) \wedge \\ & \neg\text{Sabe}(\text{Sgte\_Accion\_En\_Plan}(accy)) \end{aligned} \tag{6.29}$$

Como en los ejemplos anteriores, en realidad se disparan las dos reglas expresadas en Jena asociadas a la regla 6.29. Ambas reglas tienen como antecedente el antecedente de la regla anterior, y como consecuente uno de los objetivos inferidos. Enfocando el análisis en la segunda regla (regla 6.30) disparada, se deduce el objetivo no alcanzado *El estudiante no sabe el siguiente operador en el plan (BringToVolume)*.

```
regla28.2:
        (?a d:associatedSpecificAction ?act), (?a d:associatedActionState ?st),
        (?st d:wasApplied true), (?st d:isOfActivityExecutionState ?stactiv),
        (?stactiv d:nextActionPlan ?actplan), notEqual(?actplan, ?act),
        lisEntryElement(?act, ?actplan,?plan), (?stobj d:valuedObjective ?obj),
        (?obj rdf:type Knows_That_Next_Action_Is),
        (?obj d:requiresKnowledgeObjects ?actplan) →
        Annadir_ME(?obj, false, ?stobj)
```

(6.30)

En la regla 6.30 se ha introducido un nuevo builtin, *lisEntryElement(?ela, ?elb,?lista)*, desarrollado para verificar si *ela* está en la lista *lista* a partir del elemento de la lista *elb*.

Antes del disparo de la regla 6.30, el estado de su objetivo inferido tenía la propiedad *acquired* a valor *unknown* (Figura 6.19), debido a que no se dedujo en ninguna actividad anterior y

se supuso este valor inicialmente. Después del disparo de la regla (Figura 6.20), la ontología es actualizada; se añade una instancia del estado del objetivo con la propiedad *acquired* a *false*.

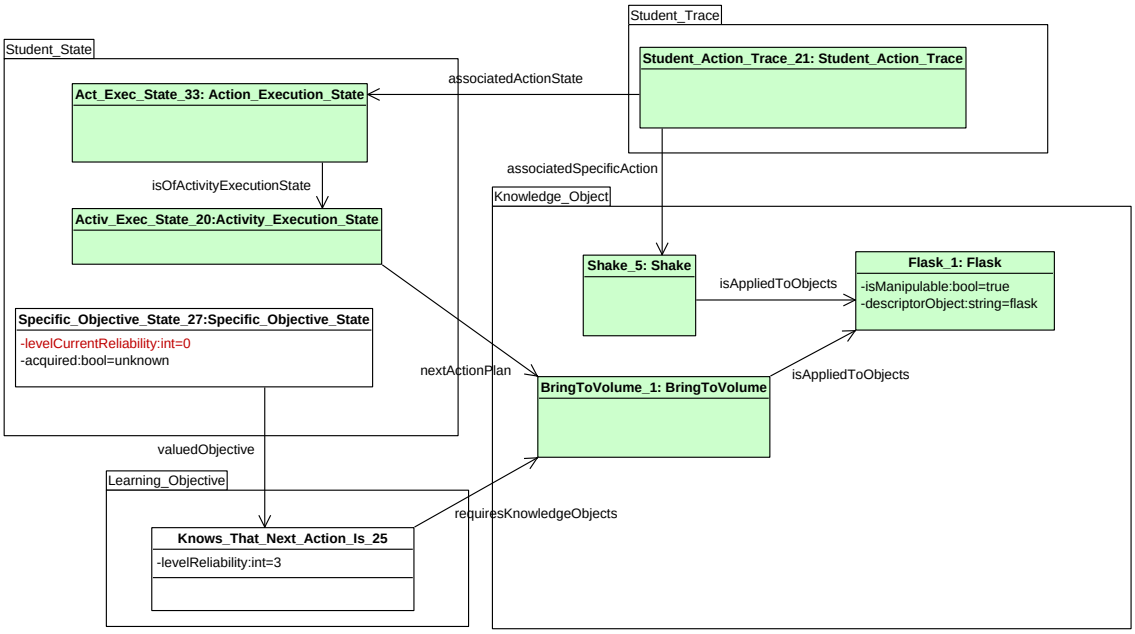


**Figura 6.19:** Estado de la ontología antes del disparo de la regla $regla28.2$

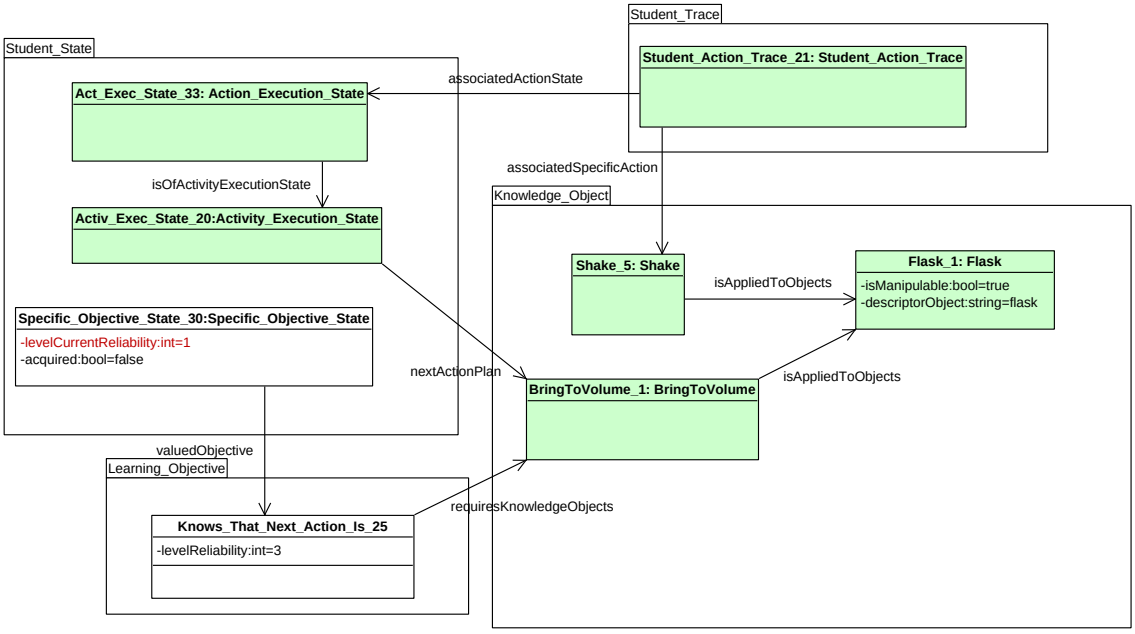


**Figura 6.20:** Estado de la ontología involucrado tras el disparo de la regla $regla28.2$

A continuación, la estrategia de tutoría decide dar una segunda oportunidad al estudiante sin proporcionarle una pista. Esta vez, el estudiante realiza la acción correcta de acuerdo al plan y *Completa el volumen del matraz con agua hasta 200 ml.* (*BringToVolume(flask, 200, water)*). Esto provoca el disparo de la regla 6.31.

$$\begin{aligned} R_{12}: \text{SI } & \text{Aplicar(accx)} \wedge \text{Sgte\_Accion\_Plan(accx)} \rightarrow \\ & \text{Sabe(Sgte\_Accion\_Plan(accx))} \end{aligned} \tag{6.31}$$

La regla previa se puede expresar en Jena de la forma siguiente:

```
regla12.1:
        (?a d:associatedSpecificAction ?act), (?a d:associatedActionState ?st),
        (?st d:wasApplied true), (?st d:isOfActivityExecutionState ?stactiv),
        (?stactiv d:nextActionPlan ?act),
        (?obj rdf:type Knows_That_Next_Action_Is),
        (?stobj d:valuedObjective ?obj), (?obj d:requiresKnowledgeObjects ?act) →
        Annadir_ME(?obj, true, ?stobj)
```

(6.32)

Como resultado del disparo de la regla anterior, el estado del objetivo *El estudiante sabe cuál es el siguiente operador en el plan (BringToVolume)* cambia de *false* a *true*. Además, se informa al ATMS mediante el nodo justificación correspondiente de acuerdo a 6.21 y 6.22.

El estado del objetivo implicado en la regla, *El estudiante sabe que el siguiente operador en el plan es BringToVolume*, antes del disparo de la regla 6.32 tenía la propiedad *acquired* a valor *false*, tal y como se vio previamente. Al dispararse esta regla, la acción *Annadir_ME* no localiza un estado de este objetivo con la propiedad *adquirido=true* y, por lo tanto, de acuerdo a su funcionalidad, añade un nuevo estado para este objetivo con la propiedad *adquirido=true* y su propiedad *levelCurrentReliability=1*.

Como se puede observar, este caso implica la existencia de una contradicción; existen dos instancias del mismo objetivo, una con propiedad *acquired=false* y la recientemente añadida con propiedad *acquired=true*. Esta contradicción debe detectarse y resolverse para que el Agente del Estudiante razone de forma no monótona cuando, como es el caso, las evidencias más recientes rechacen las conclusiones previamente obtenidas.

En la Figura 6.22, se muestra un fragmento del estado de la ontología tras del disparo de la regla 6.32.

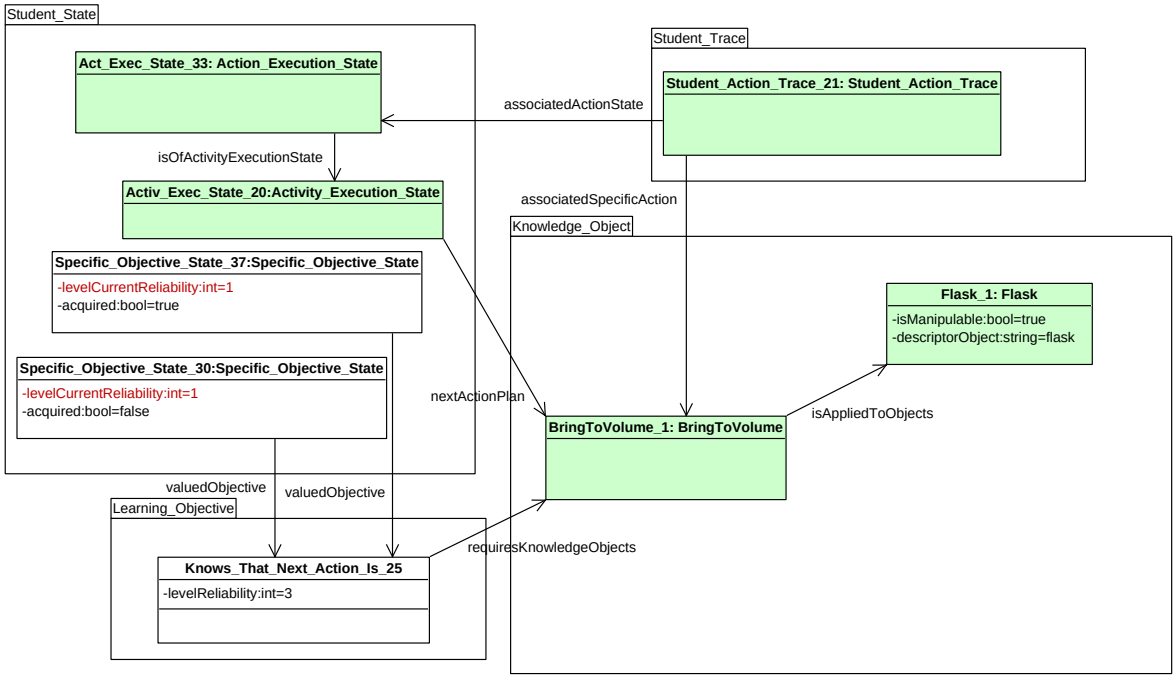


**Figura 6.21:** Estado de la ontología tras el disparo de la regla $regla12.1$

La regla de detección de contradicciones (sección 5.3.4) se dispara, al existir una contradicción entre estados del mismo objetivo: *El estudiante sabe el siguiente operador en el plan (BringToVolume)*. Esta contradicción se informa al ATMS en forma de justificación.

ATMS almacena en el registro *no-good* el entorno inconsistente correspondiente, y el módulo de resolución de conflictos es invocado para analizar el tipo de contradicción y resolverla adecuadamente. La regla 6.37 del módulo Gestor de conflictos (sección 5.3.5) deduce que es una *contradicción en el estado de objetivos causadas por cambios en la mente del estudiante.*

$$
\begin{aligned}
R_{C5}: \text{SI } & \text{Existe\_Contradiccion(objx)} \wedge \\
& \text{Estado\_Actual\_Objetivo(objx, estobjx)} \wedge \text{EsAlcanzado(estobjx)} \wedge \\
& \text{Estado\_Previo\_Objetivo(objx, estobjy)} \wedge \\
& \neg\text{Se\_Obtiene\_De\_Pista(estobjx)} \wedge \\
& \text{Eq(Fiabilidad}_{\text{E}}\text{st(estobjx), Fiabilidad}_{\text{E}}\text{st(estobjy))} \wedge \\
& \neg\text{Existen\_Ciclos\_Estados(Historico\_Objetivo(objx))} \rightarrow \\
& \text{Tipo\_Contradiccion(objx, descuido)}
\end{aligned}
\tag{6.33}
$$

El módulo Gestor de Conflictos también resuelve las contradicciones. En este caso, se satisface el antecedente de su regla $R_{C8}$ (sección 5.3.5.2), y la acción asociada a su consecuente resuelve la contradicción manteniendo el estado del objetivo más reciente (con propiedad *ac-*

*quired* a *true*) y eliminando el estado del objetivo con propiedad *acquired* a *false*.

$$
\begin{aligned}
R_{C8}: \text{SI } & (\text{Tipo\_Contradiccion}(objx, \text{descuido}) \wedge \\
& \text{Estado\_Previo\_Objetivo}(objx, estobjy) \wedge \\
& \neg\text{Estado\_Inferido\_Por}(estobjy, \\
& \quad \text{cambio\_mente\_estudiante})) \vee \\
& \text{Tipo\_Contradiccion}(objx, \text{olvido}) \vee \\
& \text{Tipo\_Contradiccion}(objx, \text{cambio\_mente\_estudiante}) \rightarrow \\
& \text{Eliminar\_Estado}(\text{Estado\_Previo\_Objetivo}(objx, estobjy))
\end{aligned}
\tag{6.34}
$$

En la Figura 6.22, se muestra un fragmento del estado de la ontología tras resolverse la contradicción al dispararse la regla 6.34.

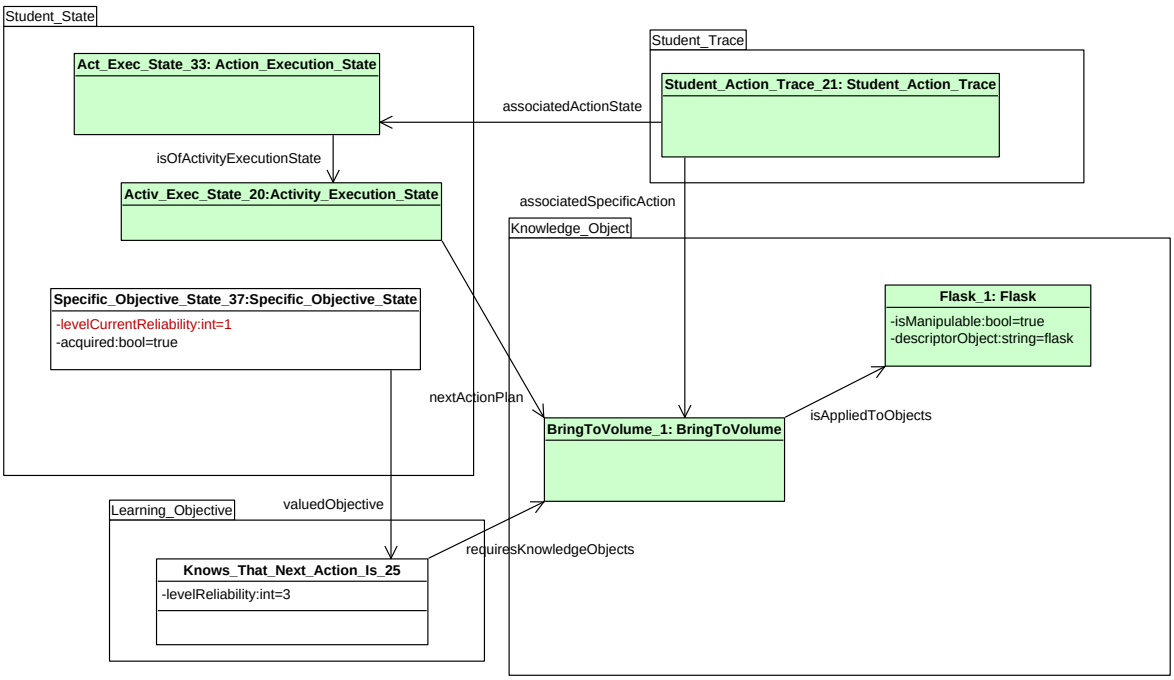


**Figura 6.22:** Estado de la ontología tras el disparo de la regla $R_{C7}$

El ATMS chequea la consistencia del nuevo entorno obtenido por el módulo de Resolución de Conflictos. Este módulo se encarga también de rechazar los supuestos a partir de los que se deducen el objetivo no alcanzado (el disparo concreto de la regla 6.30 que estableció en su momento el estado del objetivo como no alcanzado), así como las inferencias realizadas a partir de ellos.

#### 6.3.3.5 Caso de prueba 5: *El quinto estudiante ejecuta la actividad incorrectamente, se deducen objetivos no alcanzados y contradicciones en el estado de objetivos causadas por cambios en la mente del estudiante.*

El quinto estudiante ha adquirido nuevos conocimientos que han provocado cambios en su mente. Él ejecuta correctamente el plan de la actividad hasta que ejecuta la acción *Echar toda*

*la disolución del vaso de precipitado al matraz*. Después de esta acción, el estudiante agita el matraz antes de completar su volumen hasta 200 ml. con agua. Así pues, la regla 6.29 vista en el caso anterior, entre otras, se dispara, y se infieren los objetivos siguientes: *El estudiante sabe que el operador aplicado (Shake) está en el plan* y *El estudiante no sabe el siguiente operador en el plan (BringToVolume)*.

Como en la prueba anterior, en realidad se disparan las dos reglas expresadas en Jena asociadas a la regla 6.29, la R28.1 y la R28.2, entre otras. Ambas reglas tienen como antecedente el antecedente de la regla anterior, y como consecuente el objetivo inferido; R28.1 tiene como consecuente el primer objetivo inferido, y R28.2 tiene como consecuente el segundo objetivo no alcanzado.

Enfocando el análisis en la segunda regla (regla 6.30), se deduce el objetivo no alcanzado *El estudiante no sabe el siguiente operador en el plan (BringToVolume)*.

Antes del disparo de la regla 6.30, el estado de su objetivo inferido tenía la propiedad *acquired* a valor *unknown*, debido a que no se dedujo en ninguna actividad anterior y se supuso este valor inicialmente. Después del disparo de la regla, la ontología es actualizada de forma similar a los ejemplos previos. Por lo tanto, existe en este instante en la ontología una instancia del estado del objetivo con la propiedad *acquired* a *false*.

La estrategia de tutoría decide dar ahora al estudiante una segunda oportunidad pero proporcionándole una pista acerca de la siguiente acción del plan que debe ejecutar el estudiante. Como consecuencia, se dispara además la regla 6.35 del módulo de diagnóstico pedagógico (sección 5.3.2). Esta regla y su expresión en Jena se muestran a continuación:

$$\begin{aligned} R_{59}: \text{SI } & \text{Da\_Pista}(\text{Sgte\_Accion\_En\_Plan}(\text{accx})) \rightarrow \\ & \text{Sabe}(\text{Sgte\_Accion\_En\_Plan}(\text{accx})) \end{aligned} \tag{6.35}$$

La regla previa se puede expresar en Jena de la forma siguiente:

```
regla59.1:
            (?a d:associatedSpecificAction ?act), (?act rdf:type Order),
            (?st d:wasApplied true), (?st d:isOfActivityExecutionState ?stactiv),
            (?stactiv d:nextActionPlan ?actplan), (?acc d:orderObjects ?actplan),
            (?stobj d:valuedObjective ?obj),
            (?obj rdf:type Knows_That_Next_Action_Is),
            (?obj d:requiresKnowledgeObjects ?actplan) →
            Annadir_ME(?obj, true, ?stobj)
```
(6.36)

El disparo de esta regla implica que se informa al ATMS mediante el nodo justificación correspondiente de acuerdo a 6.21 y 6.22.

El estado del objetivo implicado en la regla, *El estudiante sabe que el siguiente operador en el plan es BringToVolume*, antes del disparo de la regla 6.36 tenía la propiedad *acquired* a valor *false*, tal y como se vio previamente. Al dispararse esta regla, la acción *Annadir_ME* no localiza un estado de este objetivo con la propiedad *adquirido=true* y, por lo tanto, de acuerdo a su funcionalidad, añade un nuevo estado para este objetivo con la propiedad *adquirido=true* y su propiedad *levelCurrentReliability=1*. Como se puede observar, este caso implica la existencia de una contradicción; existen dos instancias del mismo objetivo, una con propiedad *acquired=false* y la recientemente añadida con propiedad *acquired=true*. Esta contradicción debe detectarse y resolverse para que el Agente del Estudiante razone de forma no monótona cuando, como es el caso, las evidencias más recientes rechacen las conclusiones previamente obtenidas.

De este modo, la regla de detección de contradicciones (sección 5.3.4) se dispara, al existir una contradicción entre estados del mismo objetivo: *El estudiante sabe el siguiente operador en el plan (BringToVolume)*. Esta contradicción se informa al ATMS en forma de la correspondiente justificación:

ATMS almacena en el registro *no-good* el entorno inconsistente correspondiente, y el módulo de resolución de conflictos es invocado para analizar el tipo de contradicción y resolverla adecuadamente. La regla 6.37 del módulo Gestor de conflictos (sección 5.3.5) deduce que es una *contradicción en el estado de objetivos causadas por cambios en la mente del estudiante.*

$$\begin{aligned} R_{C1}: \text{SI } & \text{Existe\_Contradiccion(objx)} \wedge \\ & \text{Estado\_Actual\_Objetivo(objx, estobjx)} \wedge \text{EsAlcanzado(estobjx)} \wedge \\ & \text{Se\_Obtiene\_De\_Pista(estobjx)} \rightarrow \\ & \text{Tipo\_Contradiccion(objx, cambio\_mente\_estudiante)} \end{aligned} \tag{6.37}$$

El módulo Gestor de Conflictos también resuelve las contradicciones. En este caso, se satisface el antecedente de su regla $R_{C8}$ vista en el caso de prueba 4, y la acción asociada a su consecuente resuelve la contradicción manteniendo el estado del objetivo más reciente (con propiedad *acquired* a *true*) y eliminando el estado del objetivo con propiedad *acquired* a *false*.

El ATMS chequea la consistencia del nuevo entorno obtenido por el módulo de Resolución de Conflictos. Este módulo se encarga también de rechazar los supuestos a partir de los que se deducen el objetivo no alcanzado (el disparo concreto de la regla 6.30 que estableció en su momento el estado del objetivo como no alcanzado), así como las inferencias realizadas a partir de ellos.

#### 6.3.3.6 Caso de prueba 6: *El sexto estudiante ejecuta incorrectamente la actividad y se deducen contradicciones entre estados de objetivos por descuido del estudiante*

El sexto estudiante ejecuta correctamente el plan de la actividad hasta que debe *Llevar el vaso a la campana extractora de gases*. En vez de realizar esta acción compuesta correctamente, el estudiante *lleva el vaso a la zona donde se echa el agua* (*waterUnit area*) en vez de a la campana extractora de gases (paso 4.3 de la tabla 6.10). Como en los casos anteriores, se registra en la ontología del ME una traza de la ejecución de esta acción y más de una regla del módulo de diagnóstico pedagógico se vuelve a disparar. Entre ellas, la regla 6.38 (sección 5.3.2).

$$\begin{aligned} R_{26}: \text{SI } & \text{Aplicar(accx)} \wedge \text{Sgte\_Accion\_En\_Plan(accy)} \wedge \\ & \neg\text{Eq(accx, accy)} \wedge \text{Operador(accx, opx)} \wedge \\ & \text{Op\_Sgte\_Accion\_Plan(opy)} \wedge \neg\text{Eq(opx, opy)} \wedge \\ & \text{Op\_En\_Plan\_Post(opx)} \rightarrow \\ & \text{Sabe(Op\_En\_Plan\_Post(opx))} \wedge \\ & \neg\text{Sabe(Sgte\_Accion\_En\_Plan(accy))} \end{aligned} \tag{6.38}$$

La expresión en Jena de una de las dos reglas en las que se desdobla la regla 6.38 también se

muestra a continuación:

```
regla26.2:
            (?a d:associatedSpecificAction ?act), (?a d:associatedActionState ?st),
            (?st d:wasApplied true), (?st d:isOfActivityExecutionState ?stactiv),
            (?stactiv d:nextActionPlan ?actplan), notEqual(?actplan, ?act),
            (?act d:operatorName ?op), (?actplan d:operatorName ?opplan),
            notEqual(?op, ?opplan), lisEntryArgElement(?op, operatorName, ?actplan, ?plan),
            (?stobj d:valuedObjective ?obj),
            (?obj rdf:type Knows_That_Next_Action_Is),
            (?obj d:requiresKnowledgeObjects ?actplan) →
            Annadir_ME(?obj, false, ?stobj)
```

(6.39)

En la regla 6.39 se ha introducido un nuevo builtin, *lisEntryArgElement(?val, ?prop, ?ela, ?lista)*, desarrollado para verificar si el valor *val* para la propiedad *prop* está en algún elemento de la lista *lista* a partir del elemento *ela*.

Como resultado del disparo de la regla anterior, se deduce que *El/la estudiante no sabe la siguiente acción en el plan (Ir a la campana extractora con el vaso de precipitado)*. El objetivo no fue deducido en la actividad previa del curso así que, el estado de este objetivo antes del disparo de la regla 6.39 en la ontología tiene la propiedad *acquired* a valor *unknown* y tras el disparo, la acción *Annadir_ME* actualiza la ontología añadiendo una instancia de estado del objetivo anterior con propiedad *acquired* a valor *false*. El disparo de esta regla es informada al ATMS como una justificación, de acuerdo a 6.21 y 6.22.

El objetivo fallado no se considera extremadamente importante. La estrategia de tutoría decide dar una segunda oportunidad al estudiante proporcionándole una pista. Como consecuencia, la regla 6.36, vista en el caso de prueba 4 se dispara, entre otras reglas del módulo de diagnóstico pedagógico. De forma similar al caso de prueba precedente, se deduce que *El/la estudiante sabe la siguiente acción (en este caso, Ir a la campana extractora)* y se actualiza la ontología con un nuevo estado del objetivo con la propiedad *acquired* a valor *true*. Así pues, se produce una contradicción causada por un cambio en la mente del estudiante que se resuelve por el modulo Gestor de Conflictos, tal y como se vio en el caso de prueba anterior, manteniendo en la ontología el estado actual con la propiedad *acquired* a valor *true*.

En este momento, aún con la pista proporcionada por el Agente de Tutoría, el estudiante vuelve a ejecutar la acción y, de nuevo, no llega a alcanzar el objetivo -en este momento, el estudiante va a la puerta (con el vaso de precipitado). Consecuentemente, la regla 6.39 vuelve a dispararse (entre otras muchas) y, como al comienzo de la prueba, se añade a la ontología el estado del objetivo anterior con propiedad *acquired* a valor *false* y *levelCurrentReliability* a valor *1*. De nuevo, como en el caso 4, hay dos estados del mismo objetivo, uno con propiedad *acquired* a valor *false* y otro con propiedad *acquired* a valor *true* en la ontología del estudiante. La regla de detección de contradicciones del módulo de diagnóstico pedagógico detecta la contradicción y el ATMS es informado, de manera similar al caso 4, con la justificación de la contradicción correspondiente.

El tipo de contradicción es detectada ahora por el modulo Gestor de Conflictos a través de la

regla $R_{C4}$ que se instancia en este punto de la siguiente forma:

$$\begin{aligned} \mathrm{R_{C4}}: \text{SI } & \text{Existe\_Contradiccion(Knows\_That\_Next\_Action\_Is\_21)} \wedge \\ & \text{Estado\_Actual\_Objetivo(Knows\_That\_Next\_Action\_Is\_21,} \\ & \quad \text{Specific\_Objective\_State\_35)} \wedge \\ & \quad \neg\text{EsAlcanzado(Specific\_Objective\_State\_35)} \wedge \\ & \text{Estado\_Inferido\_Por(Knows\_That\_Next\_Action\_Is\_31,} \\ & \quad \text{cambio\_mente\_estudiante))} \wedge \\ & < (\text{Tpo\_Fin\_Adq(Knows\_That\_Next\_Action\_Is\_31, tadq), tpo\_limite\_olvido)} \rightarrow \\ & \text{Tipo\_Contradiccion(objx, descuido)} \end{aligned} \tag{6.40}$$

donde, *Knows_That_Next_Action_Is_21* es la instancia del objetivo cognitivo *Knows_That_Next_Action_Is* con los dos estados contradictorios y *Specific_Objective_State_35* es la instancia de estado del objetivo *Knows_That_Next_Action_Is_21* alcanzado actualmente (con propiedad *acquired* a valor *false*).

La expresión $< (Tiempo\_Adquisicion(Est\_Obj\_a, tpo), tpo\_lim\_descuido)$ comprueba, para un estudiante, si el tiempo desde que se adquirió el estado previo (*Knows_That_Next_Action_Is_31*) hasta la adquisición del estado actual (*Knows_That_Next_Action_Is_35*) está por debajo de un cierto umbral proporcionado por un parámetro configurable (*tpo_lim_olvido*).

Por lo tanto, en este caso, se deduce que el tipo de contradicción es un descuido del alumno y la regla 6.41 de resolución de conflictos se dispara. Esta regla resuelve la contradicción eliminando el estado actual con propiedad *acquired* a valor *false* y manteniendo el estado del objetivo previo con propiedad *acquired* a valor *true* en la ontología.

$$\begin{aligned} \mathrm{R_{C9}}: \text{SI } & \text{Tipo\_Contradiccion(objx, descuido)} \wedge \\ & \text{Estado\_Previo\_Objetivo(objx, estobjy)} \wedge \\ & \text{Estado\_Inferido\_Por(estobjy,} \\ & \quad \text{cambio\_mente\_estudiante)} \rightarrow \\ & \text{Eliminar\_Estado(Estado\_Objetivo\_Actual(objx, esty))} \end{aligned} \tag{6.41}$$

Finalmente, el ATMS chequea la consistencia del nuevo entorno obtenido por el modulo Gestor de Conflictos. Este módulo se encarga además de retraer los supuestos a partir de los que se deducían el estado del objetivo no alcanzado. En este caso, se trata de rechazar el último disparo de la regla 6.39, representada en ATMS en forma de justificación, así como las inferencias realizadas a partir de ella, de acuerdo a la estructura de datos en el ATMS.

CAPÍTULO 7

# CONCLUSIONES

Actualmente, los avances en el campo de los SITs se han acelerado, lo cual ha servido para que muchos investigadores se interesen y planteen nuevas aproximaciones que acentúen aún más este constante progreso. A pesar de ello, la construcción y mantenimiento de los SITs es compleja y aún presenta muchas carencias. En la resolución y mejora de ellos, la IA y la ingeniería del software juegan un papel crucial. Algunos autores como Mizoguchi & Bourdeau (Mizoguchi and Bourdeau (2000)) han achacado el estado actual de estos sistemas, primordialmente, a una falta de representación explícita de la conceptualización en la que se basa cada uno de ellos (ver sección 2.5.1). Una revisión minuciosa del estado del arte de los SITs nos ha llevado a corroborar la afirmación anterior debida, en gran medida -a criterio de la autora de este trabajo-, a la complejidad del desarrollo de estos SITs y, en especial, del ME y el diagnóstico pedagógico/cognitivo. Este hecho también ha motivado que el ME haya sido siempre una de las más importantes líneas de investigación en esta área y que hayan surgido numerosas aportaciones. En este documento se ha presentado una selección de algunos de los trabajos más interesantes y de mayor difusión publicados hasta la fecha.

La complejidad de los SITs se acrecienta aún más en los EVIEs, tal y como se ha analizado también en el estado de la cuestión, donde se añade a la complejidad de los SITs la de los Entornos Virtuales. Las nuevas capacidades de interacción que ofrecen estos entornos afectan a todo el proceso de aprendizaje del estudiante, lo que requiere que la estructura de los SITs, propia de los EVIEs, sea enriquecida con estos nuevos aspectos. Sin embargo, la enorme complejidad de este tipo de Entornos Virtuales no debería impedir mantener un modelo conceptual claro, estructurado y bien definido.

Del estudio y análisis de los SITs y los Entornos Virtuales y, en concreto, del ME y su diagnóstico pedagógico/cognitivo, también se ha concluido que la mayoría de ellos no consideran una taxonomía de conocimientos acerca del estudiante bien estructurada, clara y amplia. Ésta es una de las razones fundamentales para que sus MEs no sean válidos más que en ciertos dominios, o su adaptación sea muy difícil, incluso imposible, para su uso en diferentes SITs. Estas conclusiones extraídas del estudio de los MEs, tanto en SITs como en EVIEs, nos ha llevado a a desarrollar el trabajo presentado. Es importante señalar que debido a que el campo de los EVIEs es bastante reciente (finales de los años 90), no han surgido tampoco casi ninguna aproximación

con un modelo conceptual que incluya una taxonomía de conocimientos amplia o se aplique a varios dominios, circunstancia que también ha acrecentado nuestro interés por acometer esta investigación.

Como ya se estableció en el capítulo 4, el objetivo fundamental del trabajo es: *Desarrollar un nuevo modelo del estudiante basado en un enfoque pedagógico, que se caracterice por ser fácilmente adaptable, extensible, y reutilizable para su aplicación a diferentes tipos de SITs, y con un método de diagnóstico pedagógico-cognitivo con capacidades de razonamiento no monótono.* Del mismo modo, en este capítulo (secciones y ) se establecieron unos requisitos fundamentales que el modelo del estudiante propuesto debía cumplir para alcanzar el objetivo previo.

Teniendo en cuenta los requisitos definidos en este trabajo, se presentan a continuación las aportaciones más destacadas de la presente tesis al campo del modelado del estudiante:

- La primera contribución del trabajo es el desarrollo de una ontología del ME con una extensa representación de los diferentes tipos de conocimientos relativos al estudiante. El formalismo de representación de las ontologías ha permitido la representación de los conocimientos del estudiante en diferentes niveles de abstracción, y ha facilitado la reutilización y la extensión del modelo en el contexto de diferentes entornos de aprendizaje. Esta contribución está relacionada con el requisito R8: *Uso de ontologías que ayude a formalizar el proceso de construcción de un ME fácilmente extensible, adaptable, y reutilizable.*

  La ontología del ME propuesta se ha desarrollado como una red de ontologías utilizando el escenario 1 de la metodología NeOn (sección 3.3.6.1), y consta de las siguientes ontologías modulares:

  - Ontología *Student_Profile*: Representa la información personal del estudiante.

    Esta contribución está relacionada con el requisito R3: *Representación en el ME de una taxonomía del perfil del estudiante.*
  - Ontología *Knowledge_Object*: Representa los conocimientos de la materia a enseñar/aprender desde un punto de vista estructural y desde un punto de vista procedimental. Para su adaptación a EVs, se ha especializado la ontología incorporando objetos de conocimiento relativos a un EV (relativos a la estructura del EV, objetos virtuales manipulables y no manipulables del entorno, así como sus posiciones, objetos de tipo procedimental como nuevas acciones del estudiante que implican movimiento, el plan de aprendizaje, etc.). En esta extensa ontología destaca la aportación de una jerarquía de acciones clasificadas de acuerdo a cuatro criterios no disjuntos: acciones de acuerdo a la interacción con objetos, acciones de acuerdo a los personajes involucrados, acciones de acuerdo al tipo de comunicación (verbal o no verbal), y acciones de acuerdo al nivel de cumplimiento de una meta (ver sección 6.2.1)

    Esta contribución está relacionada con los requisitos R1: *Representación en el ME de una taxonomía amplia de objetos de conocimiento*, y con el requisito R9: *Representación en el ME de objetos de conocimiento relativos a un EV.*

    Es importante señalar respecto a esta ontología que, para su adaptación a uno de los entornos de aprendizaje (entornos de aprendizaje de GUIs), se ha extendido esta ontología utilizando el escenario 2 de la metodología NeOn, añadiendo una jerarquía de objetos gráficos ya existente, como recurso no ontológico (en concreto, objetos del framework *Swing*).
  - Ontología *Learning_Objective*: Representa los objetivos de aprendizaje, aspecto clave y novedoso en la ontología de Modelado del Estudiante. Da soporte al mecanismo de

diagnóstico pedagógico para inferir el estado de los objetivos (alcanzados o no) por el estudiante. La ontología se ha creado a partir de tres taxonomías ya existentes de objetivos.

Esta contribución está relacionada con el requisito R2: *Representación en el ME de una taxonomía de objetivos de aprendizaje*.

- Ontología *Student_State*: Describe el estado acumulado de los conocimientos del estudiante respecto a la ejecución de sesiones, actividades, acciones, cumplimiento de las precondiciones asociadas a las acciones, estado pedagógico del estudiante (estado de realización del plan, actividades, etc.), estado actual de objetivos de aprendizaje, estado emocional, estado de capacidades generales y competencias del estudiante (nivel de atención, memoria, etc.). Asimismo, se representa en esta ontología el estado acumulativo derivado de la traza de la trayectoria seguida por el estudiante. Estos datos acumulativos permitirán evaluar la trayectoria del alumno y complementar la valoración de la ejecución de la actividad realizada por el estudiante.

  Esta contribución está relacionada con los requisitos R4: *Representación en el ME de una taxonomía del estado de los conocimientos del estudiante*, y R11: *Representación en el ME del estado de las trayectorias seguidas por el estudiante*.

- Ontología *Student_Trace*: Representa un registro temporal de la trayectoria educativa del estudiante (traza de sesiones, actividades, variables, objetivos de aprendizaje, etc.). Además, mantiene un registro temporal de las acciones realizadas por el tutor, tales como pistas o instrucciones proporcionadas al estudiante. Adicionalmente, en un EV donde el estudiante, representado como un avatar, puede realizar otro tipos de acciones que incluyen movimiento, acciones de interacción con objetos u otros avatares, la traza de las preguntas que el estudiante puede realizar en esta clase de entornos también es representado en esta ontología. A partir de este registro, se obtienen datos acumulativos en la ontología *Student_State* utilizados para evaluar el aprendizaje del estudiante tras finalizar la ejecución de una actividad.

  Esta contribución está relacionada con los requisitos R5: *Representación en el ME de una taxonomía de la traza del estudiante*, y R10: *Representación en el ME de la traza de la trayectoria del estudiante*.

- La segunda contribución fundamental del trabajo es el desarrollo de un método de diagnóstico pedagógico-cognitivo. En primer lugar, el método es capaz de inferir el estado de los objetivos -alcanzados o no- por el estudiante durante su aprendizaje. En base a la relación ontológica establecida entre los estados de los objetivos de aprendizaje y los elementos de conocimiento que es necesario adquirir para alcanzar esos objetivos, el método propuesto permite, a posteriori, inferir el estado cognitivo del alumno.

  Para el desarrollo del método de diagnóstico se ha utilizado el motor de inferencia con encadenamiento hacia adelante del *framework* Jena.

  El desarrollo de este nuevo método ha dado lugar a las siguientes aportaciones específicas:

  - El estado de los conocimientos del estudiante obtenido por el método de diagnóstico puede realizarse en diferentes niveles de granularidad o detalle: a nivel pedagógico (a nivel de objetivos alcanzados o no por el estudiante), o a nivel cognitivo (objetos de conocimiento adquiridos o no por el estudiante).
  - Definición de una taxonomía amplia de criterios de diagnóstico (sección 5.3.2).

  - Definición de una jerarquía de patrones genéricos de reglas organizados en base a la taxonomía de criterios de diagnóstico previa. Esta jerarquía así creada ha facilitado la adaptación del método para su aplicación a diferentes SITs.

  Esta contribución está relacionada con el requisito R7: *Proporcionar un nuevo método de diagnóstico pedagógico-cognitivo fácilmente adaptable.*

- La tercera contribución es la incorporación al método de diagnóstico pedagógico propuesto de la capacidad de no monotonía. Se ha elegido para ello la técnica de IA de razonamiento no monótono ATMS (de Kleer (1986)), Sistema de Mantenimiento de la Verdad basado en suposiciones implementado por Ken Forbus y Johan de Kleer (Forbus and de Kleer (1993)). Asimismo, para realizar esta tarea se ha desarrollado un nuevo módulo de resolución de conflictos basado en reglas.

  Esta contribución está relacionada con el requisito R8: *Proporcionar un nuevo método de diagnóstico con capacidades de razonamiento no monótono.*

- La cuarta contribución es la descripción de una metodología de aplicación del ME propuesto para un sistema específico. Esta metodología ha facilitado la adaptación del modelo presentado a un sistema concreto. La metodología incluye como fases principales: la especialización de la ontología desarrollada y la adaptación del módulo de diagnóstico pedagógico al dominio de aprendizaje concreto.

La especialización del ME presentado y su aplicación se ha realizado mediante la metodología propuesta. En concreto, las pruebas se han realizado con tres prototipos: aprendizaje de un GUI -editor de textos-, aprendizaje de la programación de una lavadora y aprendizaje en un laboratorio de química, estos dos últimos en un entorno 3D. En este contexto, se ha demostrado lo siguiente:

- El uso de ontologías como formalismo de representación y el uso de las dos aportaciones previas en el método de diagnóstico, taxonomía de criterios de diagnóstico y conjunto de patrones asociados, han dotado al ME de una fácil adaptabilidad para su aplicación a diferentes SITs.

- El método de diagnóstico pedagógico-cognitivo desarrollado en el trabajo proporciona una amplia información sobre el estado de los conocimientos del estudiante al tutor, en diferentes niveles de detalle o granularidad, no sólo sobre el estado de los objetos de conocimiento adquiridos o no por el estudiante, sino también sobre el estado de los objetivos de aprendizaje, alcanzados o no por el estudiante.

  Asimismo, el diagnóstico obtenido con el soporte de la extensa taxonomía de criterios de diagnóstico aportada en el trabajo se ha demostrado que proporciona al tutor una fuente de información muy variada sobre el estado de los objetivos del estudiante. Esta información podrá mejorar las ayudas facilitadas por el tutor así como la selección de la siguiente estrategia de tutoría a seguir con el estudiante, proporcionando además una tutoría más personalizada. Durante el seguimiento o supervisión del tutor, éste puede obligar al estudiante a repetir actividades, de acuerdo a una cierta estrategia de tutoría, si detecta que el alumno no ha alcanzado ciertos objetivos o, si este error se reitera en varios sesiones, recomendarle que abandone la sesión. Pero, además, el estudiante que interactúa con el

SIT en el que está inmerso el ME propuesto tiene otras alternativas en su aprendizaje; puede realizar diferentes tipos de preguntas si comete errores, repetir una acción un número determinado de veces o recibir, según las veces que ha errado en una acción, diferentes niveles de pistas del tutor (más generales o más específicas, y de diferentes tipos), etc. Todo ello, facilita el aprendizaje con enfoque constructivista del estudiante.

CAPÍTULO 8

# LÍNEAS DE TRABAJO FUTURO

El trabajo expuesto en la presente tesis abre en el futuro importantes líneas de actuación dirigidas a las dos vertientes principales de la tesis: la ontología y el método de diagnóstico. Estas dos vertientes están estrechamente relacionadas, por lo que la realización de la mayoría de los trabajos que se proponen para una de ellas implica alguna modificación en la otra vertiente.

La línea fundamental en la que se centrará el trabajo en un futuro próximo es la realización de una validación empírica con experimentos dentro del contexto de MAEVIF y con estudiantes reales del modelado del estudiante propuesto. Esta validación nos ayudará esencialmente a lo siguiente:

- Refinar las reglas de diagnóstico.
- Detectar posibles lagunas de conocimiento en la ontología del estudiante.
- Realizar las pruebas del mecanismo no monótono asociado al proceso de diagnóstico y, por lo tanto, validar la taxonomía de conflictos incluida en él. Estos conflictos son las inconsistencias que pueden surgir en las creencias del modelo del estudiante a lo largo de una o más sesiones. Además, estas pruebas permitirán validar los mecanismos de resolución utilizados para restablecer la consistencia en el modelo mediante el uso de un ATMS.

Con todo lo anterior se pretende demostrar con más aplicaciones del ME la adaptabilidad del ME propuesto.

- Las líneas de trabajo futuro en el ámbito de la ontología del ME son:

1. Ampliación de la ontología del ME con conocimiento que proporcione un diagnóstico más preciso, más amplio y variado, incluyendo:

   1.1 Una jerarquía de diferentes taxonomías en los dominios: cognitivo, psicomotor y afectivo. Esta jerarquía permitirá al diseñador de un curso poder elegir las taxonomías que considere más adecuadas para establecer los objetivos a estos tres niveles.

1.2 Conocimiento sobre el proceso metacognitivo del alumno. La metacognición se define como la capacidad de un individuo para reflexionar, comprender y controlar su aprendizaje (Schraw and Dennison (1994)). Brown (Brown (1987)), identifica dos componentes de la metacognición: conocimiento sobre la cognición (consciencia del propio conocimiento) y regulación sobre la cognición o control sobre el proceso de aprendizaje (habilidad para hacer uso de estrategias, para planificar, regular, y evaluar el proceso de aprendizaje). Así pues, la metacognición implica conocimiento sobre lo que el estudiante sabe acerca de su propio conocimiento y lo que sabe sobre su proceso cognitivo (conocimiento sobre las estrategias a usar y sobre su aplicación).

La representación de este tipo de conocimiento, su diagnóstico y evaluación permitirían al tutor realizar una tutoría más adaptativa y facilitarían la selección de las tareas que a continuación va a plantear al estudiante con el fin de mejorar aspectos de su aprendizaje en los que tenga especial dificultad, para aprender nuevas estrategias o destrezas (enseñanza de habilidades metacognitivas) que le faciliten una formación más rápida o con mayor rendimiento, etc.

1.3 Conocimiento más detallado acerca de la trayectoria del alumno. Esta información adicional en el Entorno Virtual permitiría inferir más información sobre el estado del conocimiento del alumno.

1.4 Nuevos tipos de pistas e instrucciones proporcionados por el Módulo de Tutoría así como otros tipos de preguntas que puede realizar el estudiante en este tipo de entornos.

2. Adaptar la ontología del ME para lograr un modelado del estudiante adecuado a los requerimientos de entornos de aprendizaje virtuales como los siguientes:

2.1 Entornos en los que se requiera una mayor incorporación de aspectos íntimamente relacionados con el campo de la psicología, por ejemplo, a nivel afectivo y de habilidades emocionales. Esto permitiría aplicarlo a entornos virtuales de entrenamiento en habilidades emocionales relacionados con profesiones de alto riesgo, o con profesiones que requieran capacidades de liderazgo, etc.

2.2 Entornos en los que la ontología del ME deba proporcionar mayor información sobre aspectos psicomotores. Por ejemplo, en el ámbito de la danza o de cualquier profesión relacionada con la expresión corporal.

Los trabajos previos requerirían una ampliación de la ontología actual del ME por medio de la reutilización de otras ontologías que probablemente ya estén disponibles.

3. Incorporar la arquitectura propuesta a una arquitectura más general para su aplicación no sólo a entornos de aprendizaje monousuario sino también a entornos de enseñanza multiusuario, por ejemplo, entornos colaborativos sin las restricciones mencionadas en las hipótesis de trabajo. La jerarquía o esqueleto de la ontología presentada se ha pensado para que su conocimiento sea útil también para sistemas multiusuario. Por ejemplo, el propio registro de conocimientos tales como las trazas de objetivos, llevando un histórico del estado de los objetivos alcanzados o no por el estudiante modelado, proporcionaría una información clave para las estrategias de tutoría generales en entornos de aprendizaje en los que existan varios usuarios y se deban alcanzar, por ejemplo, objetivos por grupos de estudiantes.

- Las líneas de trabajo futuro en el ámbito del método de diagnostico propuesto son:

1. Mejora del método de resolución de conflictos con nuevas heurísticas dependientes e independientes del dominio.
2. Ampliación del conjunto de patrones de reglas propuesto. De esta forma, se conseguirá que el diagnóstico del estado del conocimiento del alumno sea más rico y más exacto. Por ejemplo, ampliación del conjunto de patrones asociado al diagnóstico de las trayectorias del estudiante. Este diagnóstico más exhaustivo puede proporcionar al Módulo de Tutoría mayor información sobre el estado cognitivo del estudiante.

- Otra línea de trabajo relacionada:

1. Desarrollo de una herramienta para la obtención (semi)automática de objetivos de aprendizaje basada en el tipo de objetos de conocimiento a adquirir y el tipo de acciones que sea posible realizar en determinados entornos. Por ejemplo, para entornos donde el objetivo es aprender la forma de interaccionar con una GUI.

# APÉNDICES

APÉNDICE A

# DESCRIPCIÓN DE LA ONTOLOGÍA DE MODELADO DEL ESTUDIANTE

En este anexo se realiza una descripción de la ontología desarrollada para el modelado del estudiante en un SIT. Con este objetivo, se presentan un conjunto de tablas que, de forma detallada, presentan poco a poco las jerarquías de la ontología; sus clases y propiedades, así como la interrelación entre ellas.

## A.1 Descripción de la ontología general de Modelado del Estudiante

La aplicación del método presentado en esta tesis incluye la ontología general de Modelado del Estudiante, que como se vio en la sección 5.2.4, ha sido concebida como una red de ontologías compuesta de 5 ontologías que se detallan a continuación mediante las tablas mencionadas previamente.

Tabla A.1: Descripción de la ontología **Student_Profile**

| CLASE | SUPERCLASE | PROPIEDADES DEFINIDAS | PROPIEDADES HEREDADAS |
|---|---|---|---|
| **Student_Profile**<br>Información personal del estudiante | **owl:Thing** | **idStudent**: identificador del estudiante<br>**hasExperienceIn**: área de experiencia del estudiante<br>**hasInteractionPreferences**: preferencias de interacción del estudiante<br>**hasLearningStyle**: preferencia de tipo de aprendizaje del estudiante<br>**hasPersonalData**: datos personales del estudiante<br>**hasPhysicalFeatures**: características físicas del estudiante<br>**hasPreviousExperienceWithComputers**: experiencia previa con ordenadores del estudiante | |
| **Physical_Features**<br>Aspectos físicos que pueden afectar al aprendizaje del alumno | **owl:Thing** | **corporalDimensions**: altura y contorno del alumno<br>**disabilities**: deficiencias del estudiante que pueden afectar al aprendizaje | |
| **Personal_Data**<br>Datos identificativos del alumno en la tutoría | **owl:Thing** | **name**<br>**firstSurname** y **secondSurname**<br>**sex**<br>**civilState**<br>**age**<br>**address**<br>**city**<br>**country**<br>**zipcode**<br>**email**<br>**dateOfBirth**<br>**phoneNumber** | |

Tabla A.1: Descripción de la ontología **Student_Profile** (cont.)

| CLASE | SUPERCLASE | PROPIEDADES DEFINIDAS | PROPIEDADES HEREDADAS |
|---|---|---|---|
| **Learning_Style**<br>Preferencias del estudiante relativas a tipo de aprendizaje (orientado a prácticas, orientado a principios, orientado a ejemplos, de lo general a lo específico o a la inversa, etc.) | **owl:Thing** | **styleType**: estilo de aprendizaje preferido por el estudiante | |
| **Experience_Area**<br>Experiencia previa en el área en el que se desarrolla el proceso del aprendizaje | **owl:Thing** | **areaName**: nombre del área de aprendizaje<br>**experiencelevel**: nivel de experiencia previa en el área de aprendizaje | |
| **Previous_Experience_Computers**<br>Nivel de experiencia previa del estudiante con los ordenadores y en el área en el que se desarrolla el proceso del aprendizaje | **owl:Thing** | **experienceWithComputers**: nivel de experiencia previa con ordenadores<br>**nameOfArea**: nombre del área concreto en el que posee experiencia<br>**experienceInArea**: nivel de experiencia previa en el área<br>**prospects**: perspectivas que posee el estudiante relativas al área de aprendizaje<br>**occupation**: actividad profesional actual del alumno | |
| **Interaction_Preferences**<br>Preferencias que posee el estudiante relativas a los medios o dispositivos de entrada y salida con los que interacciona | **owl:Thing** | **inputPreferences**: preferencia del estudiante respecto a los dispositivos de entrada (teclado, ratón, joystick, etc.)<br>**outputPreferences**: preferencia del estudiante respecto a los dispositivos de salida (monitor, pantallas de diferentes tipos, etc.) | |

Tabla A.1: Descripción de la ontología **Student_Profile** (cont.)

| CLASE | SUPERCLASE | PROPIEDADES DEFINIDAS | PROPIEDADES HEREDADAS |
|---|---|---|---|
| **Psychological_Feature**<br>Rasgos de personalidad y actitudes del estudiante frente a los contenidos de aprendizaje | **owl:Thing** | | |
| **Personality_Features**<br>Atributos de la personalidad del estudiante | **Psychological_Feature** | **personality**: características más destacadas de la personalidad del estudiante que pueden influir en su aprendizaje | |
| **Attitude**<br>Disposición del estudiante frente a lo que va a aprender | **Psychological_Feature** | **disposition**: forma de enfrentarse el estudiante a lo que a aprender (rechazo, manía o interés especial por un contenido de aprendizaje | |
| **Interest**<br>Nivel concreto de interés a priori y dinámico que presenta el estudiante frente a los contenidos de aprendizaje | **Attitude** | **aPrioriInterest**: nivel de interés a priori que presenta el estudiante frente a los contenidos de aprendizaje<br>**instantaneousInterest**: nivel de interés dinámico que va presentando el estudiante frente a los contenidos de aprendizaje | **disposition** |
| **Transitory_State**<br>Estado psicológico temporal del estudiante durante el desarrollo de una actividad | **Psychological_Feature** | **temporaryState**: estado psicológico del estudiante en un instante determinado en el desarrollo de una actividad de aprendizaje<br>**degreeOfState**: medida del estado psicológico transitorio del estudiante | |

Tabla A.2: Descripción de la ontología **Learning_Objective**

| CLASE | SUPERCLASE | PROPIEDADES DEFINIDAS | PROPIEDADES HEREDADAS |
|---|---|---|---|
| **Learning_Objective**<br>Objetivo de aprendizaje definido en un proceso educativo a nivel cognitivo, psicomotor o afectivo | **owl:Thing** | **identifierObjective**<br>**objectiveDescriptor**<br>**describesFollowUpOf**: especifica los alumnos que deberían alcanzar este objetivo<br>**isAchievedFromActivies**: especifica las actividades en las que el objetivo está definido para ser alcanzado por el alumno<br>**requiredByActivities**: especifica las actividades que requieren el objetivo como precondición para ser realizadas por el estudiante<br>**requiresKnowledgeObjects**: describe los objetos de conocimiento que debe adquirir el estudiante para alcanzar el objetivo de aprendizaje | |
| **Didactic_Objective**<br>Objetivo definido en las unidades de mayor nivel de un curso como las fases | **Learning_Objective** | **consistOfObjectives**: especifica los posibles objetivos de aprendizaje (didácticos y/o específicos) en los que se descompone un objetivo didáctico<br>**isAchievedFromPhases**: especifica las fases en las que el objetivo está definido para ser alcanzado por el alumno<br>**requiredByPhases**: especifica las fases que requieren el objetivo como precondición para ser realizadas por el estudiante | **identifierObjective**<br>**objectiveDescriptor**<br>**describesFollowUpOf**<br>**isAchievedFromActivies**<br>**requiredByActivities**<br>**requiresKnowledgeObjects** |
| **Specific_Objective**<br>Objetivo definido en los niveles bajos de las unidades de aprendizaje. Por ejemplo, en las actividades establecidas. | **Learning_Objective** | **levelReliability**: especifica el número de veces que el estudiante debe alcanzar el objetivo para considerarlo completamente adquirido | Ídem **Didactic_Objective** |

Tabla A.2: Descripción de la ontología **Learning_Objective** (cont.)

| CLASE | SUPERCLASE | PROPIEDADES DEFINIDAS | PROPIEDADES HEREDADAS |
|---|---|---|---|
| **Cognitive_Level_Objective**<br>Objetivo referido a estructuras de conocimiento[a]<br>[a]La jerarquía de objetivos a nivel cognitivo está inspirada en la taxonomía de *Bloom* (véase apartado 3.2.5) | **Specific_Objective** | | **identifierObjective**<br>**objectiveDescriptor**<br>**describesFollowUpOf**<br>**isAchievedFromActivies**<br>**requiredByActivities**<br>**requiresKnowledgeObjects**<br>**levelReliability** |
| **Objective_Knowledge** | **Cognitive_Level_ Objective** | | Ídem **Cognitive_Level_ Objective** |
| **Knows_Recognize**<br>El estudiante sabe reconocer un determinado objeto | **Objective_ Knowledge** | | Ídem **Cognitive_Level_Objective** |
| **Knows_That**<br>El estudiante sabe una cierta relación entre objetos de conocimiento. Por ejemplo: que una acción es aplicable a un determinado objeto, que dos objetos son relacionables, que un objeto existe, etc. | **Objective_ Knowledge** | | Ídem **Cognitive_Level_Objective** |
| **Knows_Where_Is**<br>El estudiante sabe donde está un cierto objeto | **Objective_ Knowledge** | | Ídem **Cognitive_Level_Objective** |
| **Knows_That_Next_Action_Is**<br>El estudiante sabe la siguiente acción en el plan | **Knows_That** | | Ídem **Cognitive_Level_Objective** |

Tabla A.2: Descripción de la ontología **Learning_Objective** (cont.)

| CLASE | SUPERCLASE | PROPIEDADES DEFINIDAS | PROPIEDADES HEREDADAS |
|---|---|---|---|
| **Knows_That_Action_In_Plan_Post** El estudiante sabe que la acción asociada están en el plan posteriormente | **Knows_That** | | Ídem **Cognitive_Level_Objective** |
| **Knows_What_Is** El estudiante conoce el objeto asociado | **Objective_ Knowledge** | | Ídem **Cognitive_Level_Objective** |
| **Objective_Comprehension** | **Cognitive_Level_ Objective** | | Ídem **Cognitive_Level_Objective** |
| **Objective_Application** | **Cognitive_Level_ Objective** | | Ídem **Cognitive_Level_Objective** |
| **Is_Able_To_Apply** | **Objective_Application** | **applicationQuality**: especifica el grado de calidad con el que el estudiante aplica un cierto operador | Ídem **Cognitive_Level_Objective** |
| **Objective_Analysis** | **Cognitive_Level_ Objective** | | Ídem **Cognitive_Level_Objective** |
| **Knows_Choose** El estudiante sabe elegir un determinado objeto | **Objective_Analysis** | | Ídem **Cognitive_Level_Objective** |
| **Objective_Synthesis** | **Cognitive_Level_ Objective** | | Ídem **Cognitive_Level_Objective** |
| **Is_Able_To_Build** El estudiante es capaz de construir un objeto que consta de partes | **Objective_Synthesis** | | Ídem **Cognitive_Level_Objective** |

Tabla A.2: Descripción de la ontología **Learning_Objective** (cont.)

| CLASE | SUPERCLASE | PROPIEDADES DEFINIDAS | PROPIEDADES HEREDADAS |
|---|---|---|---|
| **Is_Able_To_Obtain**<br>El estudiante es capaz de obtener un objeto | **Objective_Synthesis** | | Ídem **Cognitive_Level_Objective** |
| **Objective_Evaluation** | **Cognitive_Level_ Objective** | | Ídem **Cognitive_Level_Objective** |
| **Objective_Affective_Level**<br>Objetivo referido a actitudes y aspectos emocionales[a]<br>[a]La jerarquía de objetivos a nivel afectivo está inspirada en la taxonomía de *Krathwohl* (véase apartado 3.2.6) | **Specific_Objective** | | Ídem **Cognitive_Level_Objective** |
| **Objective_Attention** | **Objective_Affective_ Level** | | Ídem **Cognitive_Level_Objective** |
| **Objective_Interest** | **Objective_Affective_ Level** | | Ídem **Cognitive_Level_Objective** |
| **Objective_Attitude** | **Objective_Affective_ Level** | | Ídem **Cognitive_Level_Objective** |
| **Emotional_Ability**<br>Capacidades o aspectos emocionales que pueden requerirse como metas en el aprendizaje | **Objective_Affective_ Level** | | Ídem **Cognitive_Level_Objective** |

Tabla A.2: Descripción de la ontología **Learning_Objective** (cont.)

| CLASE | SUPERCLASE | PROPIEDADES DEFINIDAS | PROPIEDADES HEREDADAS |
|---|---|---|---|
| **Curiosity**<br>Aspecto emocional referente al deseo de saber sobre una cierta materia | **Emotional_Ability** | | Ídem **Cognitive_Level_Objective** |
| **Trust**<br>Aspecto emocional referente a la seguridad que tiene una persona sobre sí mismo | **Emotional_Ability** | | Ídem **Cognitive_Level_Objective** |
| **Stress_Control**<br>Aspecto emocional referente a la capacidad de dominio de una persona en situaciones en las que se exigen de ella un rendimiento mayor del normal | **Emotional_Ability** | | Ídem **Cognitive_Level_Objective** |
| **Fear_Control**<br>Aspecto emocional referente a la capacidad de dominio de una persona en situaciones de angustia por un riesgo o daño imaginario o real | **Emotional_Ability** | | Ídem **Cognitive_Level_Objective** |
| **Empathy**<br>Aspecto emocional referente a la capacidad de participación afectiva y, generalmente, emotiva de una persona en una realidad ajena | **Emotional_Ability** | | Ídem **Cognitive_Level_Objective** |

Tabla A.2: Descripción de la ontología **Learning_Objective** (cont.)

| CLASE | SUPERCLASE | PROPIEDADES DEFINIDAS | PROPIEDADES HEREDADAS |
|---|---|---|---|
| **Self_Regulation**<br>Aspecto emocional referente a la capacidad de preparación mental de uno mismo para una determinada cosa | **Emotional_Ability** | | Ídem **Cognitive_Level_Objective** |
| **Self_Known**<br>Aspecto emocional referente a la capacidad de conocerse a uno mismo | **Emotional_Ability** | | Ídem **Cognitive_Level_Objective** |
| **Self_Control**<br>Aspecto emocional referente a la capacidad de dominarse o controlarse uno mismo | **Emotional_Ability** | | Ídem **Cognitive_Level_Objective** |
| **Psychomotor_Level_Objective**<br>Objetivo referido a capacidades físicas, motoras o de coordinación[a]<br>[a]La jerarquía de objetivos a nivel psicomotor está inspirada en la taxonomía de *Harrow* (véase apartado 3.2.7.3) | **Specific_Objective** | | Ídem **Cognitive_Level_Objective** |
| **Objective_Reflex** | **Psychomotor_Level_Objective** | | Ídem **Cognitive_Level_Objective** |
| **Objective_Perceptual_Ability** | **Psychomotor_Level_Objective** | | Ídem **Cognitive_Level_Objective** |
| **Objective_Physical_Ability** | **Psychomotor_Level_Objective** | | Ídem **Cognitive_Level_Objective** |

Tabla A.2: Descripción de la ontología **Learning_Objective** (cont.)

| CLASE | SUPERCLASE | PROPIEDADES DEFINIDAS | PROPIEDADES HEREDADAS |
|---|---|---|---|
| **Objetive_Basis_ Fundamental_Movement** | **Psychomotor_Level_ Objective** | | Ídem **Cognitive_Level_Objective** |
| **Objective_Skilled_ Movement** | **Psychomotor_Level_ Objective** | **automationDegree**: especifica el grado de automatización requerido | Ídem **Cognitive_Level_Objective** |
| **Objective_Communication_ Non_Discursive** | **Psychomotor_Level_ Objective** | | Ídem **Cognitive_Level_Objective** |

Tabla A.3: Descripción de la jerarquía **Structural_Knowledge**

| CLASE | SUPERCLASE | PROPIEDADES DEFINIDAS | PROPIEDADES HEREDADAS |
|---|---|---|---|
| **Knowledge_Object**<br>Elemento de conocimiento que se puede aprender en una determinada actividad educativa | **owl:Thing** | **idObject**<br>**descriptorObject** | |
| **Structural_Knowledge**<br>Objeto de conocimiento que describe un conocimiento estructural | **Knowledge_ Object** | | **idObject**<br>**descriptorObject** |
| **Concept**<br>Representa una idea que puede generalizarse a todos los objetos, fenómenos, etc., de la misma categoría | **Structural_ Knowledge** | | Ídem **Structural_Knowledge** |
| **Eversight**<br>Conocimiento adquirido a través de los sentidos | **Concept** | | Ídem **Concept** |
| **Object** | **Concept** | **objectPosition**: especifica la posición del objeto asociado | Ídem **Concept** |
| **Position**<br>Describe la situación de una persona u objeto en el espacio geométrico | **Concept** | | Ídem **Concept** |
| **Process**<br>Conjunto de fases de un fenómeno natural o de una operación artificial | **Structural_ Knowledge** | | Ídem **Concept** |

Tabla A.3: Descripción de la jerarquía **Structural_Knowledge** (cont.)

| CLASE | SUPERCLASE | PROPIEDADES DEFINIDAS | PROPIEDADES HEREDADAS |
|---|---|---|---|
| **Property** Conocimiento que establece el atributo o propiedad de una persona o cosa | **Structural_ Knowledge** | **isFunctional isInverseFunctional isSymmetrical belongsTo**: especifica el concepto al que pertenece **valueType**: especifica los tipos de valores que admite **valueByDefault**: especifica el valor por defecto que toma la propiedad en caso de no asignarle otro valor **value**: valor asociado a la propiedad | Ídem **Concept** |
| **Proposition** Conocimiento que expresa una enunciación de una verdad demostrada o que se pretende demostrar | **Structural_ Knowledge** | | Ídem **Concept** |
| **Definition** Proposición que expone clara y exactamente las características genéricas que diferencian una cosa | **Proposition** | | Ídem **Concept** |
| **Formal_Definition** Definición que se expresa con precisión | **Definition** | | Ídem **Concept** |
| **Informal_Definition** Definición que se expresa sin completa precisión | **Definition** | | Ídem **Concept** |
| **Proposition_Exact_Sciences** Proposición expresada en matemáticas y en lógica | **Proposition** | | Ídem **Concept** |

Tabla A.3: Descripción de la jerarquía **Structural_Knowledge** (cont.)

| CLASE | SUPERCLASE | PROPIEDADES DEFINIDAS | PROPIEDADES HEREDADAS |
|---|---|---|---|
| **Axiom**<br>Proposición admitida sin demostración | **Proposition_Exact_ Sciences** | | Ídem **Concept** |
| **Conjecture**<br>Proposición formada por observaciones o acaecimientos | **Proposition_Exact_ Sciences** | | Ídem **Concept** |
| **Corollary**<br>Proposición deducida de lo demostrado antes (no necesita una prueba particular) | **Proposition_Exact_ Sciences** | | Ídem **Concept** |
| **Formal_Fact**<br>Proposición que especifica con exactitud una acción o cosa que sucede | **Proposition_Exact_ Sciences** | | Ídem **Concept** |
| **Lemma**<br>Proposición que se debe demostrar antes de establecer un teorema | **Proposition_Exact_ Sciences** | | Ídem **Concept** |
| **Theorem** | **Proposition_Exact_ Sciences** | | Ídem **Concept** |
| **Proposition_Natural_Sciences**<br>Proposición expresada en las ciencias que estudian la naturaleza | **Proposition** | | Ídem **Concept** |

Tabla A.3: Descripción de la jerarquía **Structural_Knowledge** (cont.)

| CLASE | SUPERCLASE | PROPIEDADES DEFINIDAS | PROPIEDADES HEREDADAS |
|---|---|---|---|
| **Empirical_Fact** Proposición que expresa una acción o cosa que sucede expuesta a través de la experiencia | **Proposition_Natural_Sciences** | | Ídem **Concept** |
| **Hypothesis** Proposición que expresa una suposición posible o imposible para extraer una consecuencia | **Proposition_Natural_Sciences** | | Ídem **Concept** |
| **Law** Proposición que expresa una relación entre ciertas magnitudes que intervienen en un fenómeno | **Proposition_Natural_Sciences** | | Ídem **Concept** |
| **Principle** Una de las primeras proposiciones por la que se empieza a estudiar las ciencias o las artes | **Proposition_Natural_Sciences** | | Ídem **Concept** |
| **Theory** Proposición que describe un conjunto de leyes para relacionar un determinado orden de fenómenos | **Proposition** | | Ídem **Concept** |

Tabla A.3: Descripción de la jerarquía **Structural_Knowledge** (cont.)

| CLASE | SUPERCLASE | PROPIEDADES DEFINIDAS | PROPIEDADES HEREDADAS |
|---|---|---|---|
| **Relation**<br>Representa una conexión o correspondencia entre conceptos o propiedades | **Structural_ Knowledge** | **relationName**<br>**domain**: concepto o propiedad origen de la relación<br>**range**: concepto o propiedad destino de la relación<br>**cardinality**: número de conceptos o propiedades que participan en la relación<br>**reflexive**: especifica si la relación tiene la propiedad reflexiva<br>**symmetrical**: especifica si la relación tiene la propiedad simétrica<br>**transitive**: especifica si la relación tiene la propiedad antisimétrica | Ídem **Concept** |
| **Is_Applicable_To**<br>Relación que establece que una acción es aplicable a un cierto objeto | **Relation** | | **idObject**<br>**descriptorObject**<br>**relationName**<br>**domain**<br>**range**<br>**cardinality**<br>**reflexive**<br>**symmetrical**<br>**transitive** |
| **Exist**<br>Relación para establecer la existencia de un objeto | **Relation** | | Ídem **Is_Applicable_To** |

Tabla A.3: Descripción de la jerarquía **Structural_Knowledge** (cont.)

| CLASE | SUPERCLASE | PROPIEDADES DEFINIDAS | PROPIEDADES HEREDADAS |
|---|---|---|---|
| **Are_Related**<br>Relación para establecer que un objeto se relaciona con otro en la aplicación de una acción | **Relation** | | Ídem **Is_Applicable_To** |
| **Subclass_Of**<br>Relación para establecer que un tipo de objeto es una subclase de otro tipo de objeto | **Relation** | | Ídem **Is_Applicable_To** |
| **Is_Of_Class**<br>Relación para establecer que un objeto pertenece a una cierta clase de objetos | **Relation** | | Ídem **Is_Applicable_To** |
| **Requires_Precond**<br>Relación para establecer que una cierta acción requiere para su ejecución del cumplimiento de una cierta precondición | **Relation** | | Ídem **Is_Applicable_To** |
| **Variable**<br>Representa una variable cuyo contenido puede ser un conocimiento estructural | **Structural_ Knowledge** | **variableValue**: valor asociado a la variable | Ídem **Concept** |

Tabla A.4: Descripción de la jerarquía **Procedural_Knowledge**

| CLASE | SUPERCLASE | PROPIEDADES DEFINIDAS | PROPIEDADES HEREDADAS |
|---|---|---|---|
| **Procedural_Knowledge**<br>Objeto de conocimiento basado en la especificación de las fases o pasos sucesivos de una cierta operación | **Knowledge_ Object** | | **idObject**<br>**descriptorObject** |
| **Plan**<br>Describe los pasos o elementos del plan necesarios para resolver una cierta actividad educativa. Cada uno de estos pasos pueden realizarse a través de la ejecución de una cierta acción o de la ejecución ordenada o no de un conjunto de acciones | **Procedural_ Knowledge** | **consistsOfBlocks**: especifica los bloques de acciones y/o acciones simples (**Plan_Element**) de que consta el Plan<br>**costOfPlan**: representa el coste asociado a la ejecución del plan. Se obtendrá a través de una determinada función | Ídem **Procedural_Knowledge** |
| **Plan_Element**<br>Describe un elemento de un plan | **Procedural_ Knowledge** | **relativePositionInPlan**: posición que ocupa dentro del plan de la actividad | Ídem **Procedural_Knowledge** |
| **Compound_Action**<br>Describe un elemento de un plan educativo formado por más de una acción y/o conjunto de acciones | **Plan_Element** | **consistsOfElementPlan**: especifica el conjunto de elementos de tipo **Plan_Element** | **idObject**<br>**descriptorObject**<br>**relativePositionInPlan** |
| **Sequence_Block**<br>Describe una acción compuesta cuyos elementos del plan deben realizarse en un cierto orden y sólo en ese | **Compound_ Action** | | **idObject**<br>**descriptorObject**<br>**relativePositionInPlan**<br>**consistsOfElementPlan** |

Tabla A.4: Descripción de la jerarquía **Procedural_Knowledge** (cont.)

| CLASE | SUPERCLASE | PROPIEDADES DEFINIDAS | PROPIEDADES HEREDADAS |
|---|---|---|---|
| **Bloque_Unordered**<br>Describe una acción compuesta cuyos elementos del plan pueden realizarse en cualquier orden | **Compound_ Action** | | Ídem **Sequence_Block** |
| **Action_Application**<br>Describe un elemento de un plan educativo simple, es decir, la aplicación de una acción concreta | **Plan_Element** | **associatedAction**: especifica la **Punctual_Action** asociada<br>**initialTime** y **finalTime**: establecen el intervalo de tiempo de la aplicación de la acción | Ídem **Compound_Action** |
| **Punctual_Action**<br>Describe cualquier tipo de operación realizada por el estudiante en el entorno específico de su aprendizaje | **Procedural_ Knowledge** | **operatorName**: con el que realizar la acción<br>**preconditions**: que se deben cumplir para que se pueda ejecutar el operador asociado a la acción<br>**consequences**: efecto de la ejecución del operador<br>**achievedGoals**: especifica las metas que se alcanzan al ejecutar una acción<br>**role**: especifica quién puede realizar la acción<br>**executionWay**: propiedad opcional que establecería (normalmente, a través de una función) el modo en que se realizó la acción<br>**qualityFunction**: propiedad opcional que establecería (normalmente, a través de una función) la calidad con que se realizó la acción<br>**usesObjectsAsTool**: propiedad que establece, si fuera necesario, el objeto que se debe usar como herramienta para realizar la acción | Ídem **Procedural_Knowledge** |

Tabla A.4: Descripción de la jerarquía **Procedural_Knowledge** (cont.)

| CLASE | SUPERCLASE | PROPIEDADES DEFINIDAS | PROPIEDADES HEREDADAS |
|---|---|---|---|
| **Condition_On_State**<br>Describe una de las condiciones asociadas al estado de ejecución de una acción. Esta condición puede ser una condición que deba cumplirse para que se ejecute una acción o una condición consecuencia de la ejecución de una acción | **Procedural_ Knowledge** | **entity**: especifica la entidad sobre la que se establece la condición asociada | Ídem **Procedural_Knowledge** |
| **Is_Of_Type**<br>Describe como condición de la acción el tipo de la entidad asociada a esta condición | **Condition_On_ State** | **conditionType**: especifica el tipo establecido en la precondición | **idObject**<br>**descriptorObject**<br>**entity** |
| **To_Be_In_Position**<br>Describe como condición de la acción la posición en la que debe estar la entidad asociada a esta condición | **Condition_On_ State** | **position**: especifica la posición para la entidad asociada a la condición | Ídem **Is_Of_Type** |
| **Exist_Entity**<br>Describe como condición de la acción que la entidad asociada a esta condición exista | **Condition_On_ State** | | Ídem **Is_Of_Type** |
| **Exist_Instances_Of_Class**<br>Describe como condición de la acción que existan instancias de la clase a la que pertenece la entidad asociada a esta condición | **Condition_On_ State** | | Ídem **Is_Of_Type** |

Tabla A.4: Descripción de la jerarquía **Procedural_Knowledge** (cont.)

| CLASE | SUPERCLASE | PROPIEDADES DEFINIDAS | PROPIEDADES HEREDADAS |
|---|---|---|---|
| **Property_Name**<br>Describe como condición de la acción que la entidad asociada a esta condición tenga una determinada propiedad | **Condition_On_ State** | **propertyId**: especifica el nombre de la propiedad establecido en la condición | Ídem **Is_Of_Type** |
| **Property_Value**<br>Describe como condición de la acción que la entidad asociada a esta condición tenga una propiedad con un determinado valor | **Condition_On_ State** | **propertyName**: especifica el nombre de la propiedad establecido en la precondición<br>**propertyValue**: especifica el valor que debe tener la propiedad de esta precondición para que se pueda ejecutar el operador de la acción asociada | Ídem **Is_Of_Type** |
| **Punctual_Action_Element**<br>Describe uno de los elementos de tipo condición de la acción. Este elemento puede ser una precondición o una post-condición | **Procedural_ Knowledge** | **condOnState**: especifica una condición concreta asociada a la ejecución de una acción | Ídem **Procedural_Knowledge** |
| **Precondition**<br>Describe una de las condiciones que se debe cumplir para que se ejecute el operador asociado a una acción | **Punctual_Action_ Element** | | **idObject**<br>**descriptorObject**<br>**condOnState** |
| **Postcondition**<br>Describe una de las consecuencias de ejecutar el operador asociado a una acción | **Punctual_Action_ Element** | | Ídem **Precondition** |

Tabla A.4: Descripción de la jerarquía **Procedural_Knowledge** (cont.)

| CLASE | SUPERCLASE | PROPIEDADES DEFINIDAS | PROPIEDADES HEREDADAS |
|---|---|---|---|
| **Add**<br>Una de las consecuencias de ejecutar el operador asociado a una acción es añadir o modificar una propiedad, una relación o una posición | **Postcondition** | | Ídem **Precondition** |
| **Delete**<br>Una de las consecuencias de ejecutar el operador asociado a una acción es eliminar una propiedad, una relación o una instancia | **Postcondition** | | Ídem **Precondition** |
| **Procedure**<br>Describe una secuencia de acciones | **Procedural_ Knowledge** | **includesSequenceOfActions**: especifica los múltiples elementos de tipo **Action_Application** de que consta el procedimiento | Ídem **Procedural_Knowledge** |
| **Rule**<br>Describe un conjunto de operaciones que deben llevarse a cabo para realizar una cierta inferencia correcta | **Procedural_ Knowledge** | **ruleName** | Ídem **Procedural_Knowledge** |
| **Heuristic** | **Rule** | | **idObject**<br>**descriptorObject**<br>**ruleName** |

Tabla A.5: Descripción de la ontología **Student_State**

| CLASE | SUPERCLASE | PROPIEDADES DEFINIDAS | PROPIEDADES HEREDADAS |
|---|---|---|---|
| **Student_State_Related**<br>Datos obtenidos dinámicamente durante la actividad de aprendizaje o tras su finalización, y que establecen el estado del estudiante y de sus conocimientos | **owl:Thing** | **stateOfStudent**: especifica el estudiante al que se refiere este estado | |
| **Performance_State**<br>Estado de la ejecución del estudiante durante la realización de una cierta actividad pedagógica | **Student_State_ Related** | | **stateOfStudent** |
| **Condition_State**<br>Estado de una condición (precondición o post-condición) asociada a una ejecución concreta de una determinada acción | **Performance_ State** | **refersToCondition**: especifica la condición a la que se refiere este estado | **stateOfStudent** |
| **Precondition_State**<br>Estado de una precondición asociada a una ejecución de una determinada acción | **Condition_State** | **isMet**: especifica si al intentar aplicar el estudiante una cierta acción, se cumple o no la precondición asociada | **stateOfStudent**<br>**refersToCondition** |
| **Postcondition_State**<br>Estado de una precondición asociada a una ejecución de una determinada acción | **Condition_State** | **goalQuality**: especifica la valoración cualitativa con la que se ha alcanzado esta meta o consecuencia. La ejecución de algunas acciones, como por ejemplo la de movimiento del estudiante, pueden dar lugar a la obtención de una meta con una calidad no máxima | Ídem **Precondition_State** |

Tabla A.5: Descripción de la ontología **Student_State** (cont.)

| CLASE | SUPERCLASE | PROPIEDADES DEFINIDAS | PROPIEDADES HEREDADAS |
|---|---|---|---|
| **Action_Execution_State**<br>Especifica información obtenida sobre la acción ejecutada por el estudiante en un instante determinado | **Performance_ State** | **obtainedFromActionTrace**: especifica la traza de la acción a partir de la que se obtiene información sobre su estado<br>**wasApplied**: especifica si la acción fue ejecutada por el estudiante o no, en función del cumplimiento o no de sus precondiciones<br>**consistsOfPrecondState**: especifica el estado de las precondiciones asociadas al intentar ejecutar o al ejecutar la acción asociada<br>**consistsOfConseqState**: especifica el estado de las consecuencias asociadas a una acción ejecutada<br>**isOfActivityExecutionState**: especifica el estado de ejecución de la actividad a la que pertenece este estado de ejecucion de acción | Ídem **Performance_State** |
| **Activity_Execution_State**<br>Especifica el estado de la ejecución actual de una actividad obtenido a partir de ciertos datos tras finalizar dicha actividad | **Performance_ State** | **consistsOfActionExecState**: estados de ejecución de las acciones realizadas en la actividad actual<br>**associatedActivityTrace**: traza de actividad asociada a este estado<br>**obtainedOfActionState**: especifica el estado final de las acciones a partir del que se obtendrá el estado de la ejecución de la actividad correspondiente<br>**mistakeRate**: porcentaje de fallos o de acciones que no coinciden con la planificada durante la ejecución de la actividad<br>**successRate**: porcentaje de éxitos o de acciones que coinciden con la planificada durante la ejecución actual de la actividad<br>**numberOfProvidedExamples**: número de ejemplos proporcionados al estudiante durante la ejecución de la actividad<br>**numberOfVainAttempts**: número de intentos del estudiante antes de realizar las acciones planificadas en la ejecución de la actividad | Ídem **Performance_State** |

Tabla A.5: Descripción de la ontología **Student_State** (cont.)

| CLASE | SUPERCLASE | PROPIEDADES DEFINIDAS | PROPIEDADES HEREDADAS |
|---|---|---|---|
| **Activity_Execution_State**<br>Especifica el estado de la ejecución actual de una actividad obtenido a partir de ciertos datos obtenidos al finalizar dicha actividad | **Performance_ State** | **numberOfProvidedProblems**: número de problemas proporcionados al estudiante para que ejecute correctamente una actividad<br>**numberOfReplannings**: número de veces que el estudiante realizó una acción equivocada pero que, a partir de ella, aún es posible llegar a la meta de la actividad con un nuevo plan<br>**numberOfCorrectAnswers**: número de veces que el estudiante realizó una acción que coincidía con la del plan de la actividad<br>**numberOfIncorrectAnswers**: especifica el número de veces que el estudiante realizó una acción que no coincidía con la del plan de la actividad<br>**numberOfBasicQuestions**, **numberOfGeneralQuestions**, **numberOfAdvancedQuestions**: número de preguntas básicas, generales o avanzadas respectivamente, que el estudiante ha formulado durante la ejecución de la actividad<br>**numberOfHints**: número de pistas proporcionadas por el módulo de tutoría al estudiante durante la ejecución de la actividad<br>**devotedTime**: tiempo que ha requerido el estudiante para realizar la ejecución actual de la actividad<br>**executionFactor**: valor que sirve para evaluar cuantitativamente la ejecución actual de la actividad por el estudiante. Se obtiene a partir de la evaluación de las acciones ejecutadas por el estudiante durante la actividad<br>**costActivityExecution**: especifica el número de acciones realizadas en la ejecución de la actividad | **stateOfStudent** |

Tabla A.5: Descripción de la ontología **Student_State** (cont.)

| CLASE | SUPERCLASE | PROPIEDADES DEFINIDAS | PROPIEDADES HEREDADAS |
|---|---|---|---|
| **Activity_Execution_State**<br>Especifica el estado de la ejecución actual de una actividad obtenido a partir de ciertos datos obtenidos al finalizar dicha actividad | **Performance_ State** | **assessmentActivityExecution**: especifica una valoración cualitativa del conocimiento que posee el estudiante de la actividad basada exclusivamente en la ejecución actual<br>**nextActionPlan**: especifica cuál es la siguiente acción del plan que corresponde ejecutar | Ídem **Performance_State** |
| **Session_State**<br>Especifica el estado de la sesión actual en la que se encuentra un estudiante | **Performance_ State** | **duration**: establece la duración de la sesión actual, si está limitada o, por el contrario, si la sesión es indefinida<br>**achievementDegree**: porcentaje de sesión realizado hasta el momento<br>**started**: especifica si el estudiante ya ha iniciado la sesión<br>**isRelatedToSessionTrace**: establece la traza asociada a la sesión | Ídem **Performance_State** |
| **Pedagogical_State**<br>Especifica el estado de aprendizaje del estudiante relativo a la realización de un cierto plan de estudios | **Studen_State_ Related** | **completed**: especifica si se ha realizado o no un cierto plan de estudios<br>**numberOfTimes**: numero de veces que se ha realizado un cierto plan de estudios<br>**averageGrade**: valoración media asociada a la realización de un cierto plan de estudios | Ídem **Performance_State** |

Tabla A.5: Descripción de la ontología **Student_State** (cont.)

| CLASE | SUPERCLASE | PROPIEDADES DEFINIDAS | PROPIEDADES HEREDADAS |
|---|---|---|---|
| **Activity_State**<br>Especifica el estado pedagógico actual de una cierta actividad relativo a un estudiante. Este estado viene dado por un conjunto de datos generales asociados a la actividad y obtenidos de las sucesivas ejecuciones de la misma por el estudiante | **Pedagogical_State** | **belongsToActivity**: nombre de la actividad a la que está asociado este estado<br>**basicQuestionAverage**, **generalQuestionAverage**, **advancedQuestionAverage**: valor medio de preguntas básicas, generales o avanzadas respectivamente, que el estudiante ha formulado durante las ejecuciones de la actividad<br>**replanningAverage**: valor medio de replanificaciones realizadas durante las sucesivas ejecuciones de la actividad por el estudiante<br>**hintAverage**: valor medio de pistas ofrecidas por el módulo de tutoría al estudiante durante las diversas ejecuciones de la actividad<br>**factorActivityKnowledge**: evaluación del conocimiento que el estudiante posee de una actividad. Se obtiene de las diversas ejecuciones realizadas por el estudiante de la actividad<br>**obtainedFromExecutions**: estados de ejecución de la actividad a partir de los que se obtiene el estado de esta actividad<br>**stateExperienceActivity**: estado de experiencia de la actividad alcanzado por el estudiante hasta la actualidad | **stateOfStudent**<br>**completed**<br>**numberOfTimes**<br>**averageGrade** |
| **Course_State**<br>Especifica el estado pedagógico actual de un curso perteneciente a un plan de estudios | **Pedagogical_State** | **belongsToCourse**: nombre del curso al que está asociado este estado<br>**isComposedOfPhases**: fases de las que está compuesta el curso | Ídem **Activity_State** |

Tabla A.5: Descripción de la ontología **Student_State** (cont.)

| CLASE | SUPERCLASE | PROPIEDADES DEFINIDAS | PROPIEDADES HEREDADAS |
|---|---|---|---|
| **Phase_State**<br>Especifica el estado pedagógico actual del estudiante respecto a una cierta fase de un curso | **Pedagogical_State** | **belongsToPhase**: nombre de la fase a la que está asociada este estado<br>**isComposedOfActivities**: actividades de las que está compuesta la fase | Ídem **Activity_State** |
| **Syllabus_State**<br>Especifica el estado pedagógico actual del estudiante respecto a un cierto plan de estudios | **Pedagogical_State** | **belongsToCurriculum**: nombre del plan de estudios al que está asociado este estado<br>**isComposedOfCourses**: cursos de los que está compuesta el plan de estudios | Ídem **Activity_State** |
| **Capacity_State**<br>Especifica la valoración de ciertas capacidades del estudiante. Se obtienen tras la ejecución de la actividad de un plan de estudios | **Student_State_ Related** | | Ídem **Performance_State** |
| **Memory_Assessment**<br>Especifica la valoración de la capacidad de memoria del estudiante | **Capacity_State** | **memoryLevel**: establece una valoración cualitativa de la capacidad de memoria del estudiante tras la ejecución de una actividad (muy pobre, pobre, normal, buena, muy buena, excelente, etc.) | Ídem **Performance_State** |
| **Reasoning_Assessment**<br>Especifica la valoración de la capacidad de razonamiento del estudiante | **Capacity_State** | **reasoningLevel**: establece una valoración cualitativa de la capacidad de razonamiento que posee el estudiante tras la ejecución de una actividad (muy bajo, bajo, normal, alto, muy alto, etc.) | Ídem **Performance_State** |
| **Learning_Speed_Assessment**<br>Especifica la valoración de la velocidad de aprendizaje del estudiante | **Capacity_State** | **learningSpeed**: establece una valoración cualitativa de la velocidad de aprendizaje mostrada por el estudiante tras la ejecución de una actividad (muy lenta, lenta, normal, alta, muy alta, excelente. | Ídem **Performance_State** |

Tabla A.5: Descripción de la ontología **Student_State** (cont.)

| CLASE | SUPERCLASE | PROPIEDADES DEFINIDAS | PROPIEDADES HEREDADAS |
|---|---|---|---|
| **Emotional_State**<br>Especifica una valoración dinámica del estado emocional del estudiante que varía continuamente durante la ejecución de la actividad | **Student_State_ Related** | **emotionalStateObtained**: establece una valoración del estado emocional (nerviosismo, alegría, etc.) que refleja el estudiante. Este estado se obtiene a partir del chequeo de una o más variables durante un cierto intervalo de tiempo (traza emocional)<br>**intensityObtained**: establece la intensidad del estado emocional. Se obtiene en la realización de una actividad a partir de la traza emocional del estudiante durante un cierto intervalo de tiempo<br>**comesFromEmotionalTrace**: es la traza emocional (traza de una o más variables emocionales) del estudiante. Se obtiene durante un cierto intervalo de tiempo para valorar su estado emocional | Ídem **Performace_State** |
| **Experience_State**<br>Especifica una valoración del grado de experiencia adquirido por el estudiante al finalizar una actividad | **Student_State_ Related** | **experienceLevel**: valoración cualitativa del grado de experiencia adquirido por el estudiante tras la realización de una actividad (Novato, Principiante, Intermedio, Avanzado, etc.) | Ídem **Performance_State** |
| **Objective_State**<br>Establece dinámicamente si el estudiante ha alcanzado o no un objetivo durante el transcurso de una cierta actividad educativa | **Student_State_ Related** | **valuedObjective**: objetivo de aprendizaje valorado para el estudiante<br>**adquired**: establece la valoración del objetivo: alcanzado, no alcanzado o desconocido<br>**isAssumed**: establece si el estado de objetivo es asumido como estado inicial<br>**inferBy**: establece el motivo (tipo de contradicción), si lo hubiera, inferido para este estado. Esta propiedad facilita la tarea de resolución de contradicciones del método de diagnóstico<br>**associatedObjectiveTrace**: establece la traza del objetivo a partir de la que se obtiene su estado | Ídem **Performance_State** |

Tabla A.5: Descripción de la ontología **Student_State** (cont.)

| CLASE | SUPERCLASE | PROPIEDADES DEFINIDAS | PROPIEDADES HEREDADAS |
|---|---|---|---|
| **Didactic_Objective_State**<br>Especifica para un estudiante la valoración obtenida de un objetivo didáctico durante la realización de una cierta actividad | **Objective_State** | | **stateOfStudent**<br>**valuedObjective**<br>**associatedObjectiveTrace**<br>**acquired**<br>**isAssumed**<br>**inferBy** |
| **Specific_Objective_State**<br>Especifica la valoración obtenida para el estudiante de un objetivo específico durante la realización de una actividad | **Objective_State** | **levelCurrentReliability**: establece el número de veces que, hasta el instante actual, el estudiante ha alcanzado (o no) el objetivo específico. Un objetivo específico puede ser alcanzado tras la realización de varias actividades distintas o varias veces en la misma ejecución de una actividad | Ídem **Didactic_Objective_State** |
| **Knowledge_Object_State**<br>Especifica dinámicamente para un estudiante la valoración obtenida de un objeto de conocimiento, estructural o procedimental, durante la realización de una actividad educativa | **Student_State_ Related** | **valuedObject**: establece el objeto de conocimiento al que pertenece este estado<br>**objectKnowledgeAssessment**: establece una valoración cualitativa de la bondad en la adquisición de un objeto de conocimiento | Ídem **Performance_State** |
| **Structural_Knowledge_State**<br>Especifica dinámicamente para un estudiante la valoración obtenida de un objeto de conocimiento estructural durante la realización de una actividad educativa | **Knowledge_Object_ State** | | **stateOfStudent**<br>**valuedObject**<br>**objectKnowledgeAssessment** |

Tabla A.5: Descripción de la ontología **Student_State** (cont.)

| CLASE | SUPERCLASE | PROPIEDADES DEFINIDAS | PROPIEDADES HEREDADAS |
|---|---|---|---|
| **Procedural_Knowledge_State**<br>Especifica dinámicamente para un estudiante la valoración obtenida de un objeto de conocimiento procedimental durante la realización de una actividad educativa | **Knowledge_Object_State** | **numberAppliedWell**: número de veces que se aplicó bien el conocimiento procedimental<br>**numberAppliedBad**: número de veces que se aplicó mal el conocimiento procedimental<br>**applicationTime**: intervalo de tiempo durante el que se aplicó el conocimiento procedimental | Ídem **Structural_Knowledge_State** |
| **Actions_State**<br>Especifica para un estudiante la valoración obtenida de las acciones realizadas tras finalizar el desarrollo de una actividad educativa | **Procedural_Knowledge_State** | **belongsToActivityTrace**: establece la traza de ejecución de la actividad a la que pertenecen las acciones valoradas<br>**consistsOf**: establece los estados de cada una de las acciones que forman parte de esta valoración<br>**performanceFactor**: establece una evaluación cuantitativa de las acciones realizadas durante una actividad. Se obtiene al finalizar la actividad y a partir de la valoración o estado obtenido para cada una de sus acciones | **stateOfStudent**<br>**valuedObject**<br>**objectKnowledgeAssessment**<br>**numberAppliedWell**<br>**numberAppliedBad**<br>**applicationTime** |

Tabla A.5: Descripción de la ontología **Student_State** (cont.)

| CLASE | SUPERCLASE | PROPIEDADES DEFINIDAS | PROPIEDADES HEREDADAS |
|---|---|---|---|
| **Action_State**<br>Establece dinámicamente para un estudiante la valoración obtenida de una acción realizada en una actividad de aprendizaje | **Procedural_ Knowledge_ State** | **numberAppliedNotInPlan**: número de veces que el operador asociado a la acción aplicada no está en el plan de la actividad<br>**numberOfCorrects**: número de veces que se aplicó la acción y coinciden tanto el operador como los argumentos con la siguiente en el plan de la actividad<br>**numberOpNotArgs**: número de veces que se aplicó la acción coincidiendo su operador con el siguiente en el plan de la actividad pero no sus argumentos<br>**numberNotSeq**: número de veces que se aplicó la acción coincidiendo su operador y sus argumentos no con la siguiente en el plan de la actividad pero sí con otra<br>**numberNotSeqOpNotArgs**: número de veces que se aplicó la acción, no coincide con la siguiente en el plan, sí coincide su operador con el de otra acción perteneciente al plan pero no sus argumentos<br>**numberNotMetPrecond**: número de veces que no se aplicó el operador asociado a la acción porque no se cumplía alguna de las precondiciones<br>**numberNotAppliedForObject**: número de veces que no se aplicó el operador asociado a la acción porque no se cumple la siguiente precondición específica: el (los) objeto(s) al que se intentaba aplicar la acción no es correcto acorde al entorno de la actividad | Ídem **Actions_State** |

Tabla A.6: Descripción de la ontología **Student_Trace**

| CLASE | SUPERCLASE | PROPIEDADES DEFINIDAS | PROPIEDADES HEREDADAS |
|---|---|---|---|
| **Student_Trace_Related**<br>Aspectos de la traza o registro temporal de la ejecución de un estudiante durante una actividad pedagógica | **owl:Thing** | **belongsToStudent**: especifica el estudiante al que se refiere la traza<br>**initialTime**: especifica el tiempo en el que comienza a registrarse la traza<br>**finalTime**: especifica el tiempo en el que termina el registro de la traza | |
| **Action_Trace**<br>Describe el registro temporal de una cierta acción realizada durante el aprendizaje de un estudiante | **Student_Trace_ Related** | **associatedSpecificAction**: especifica la acción concreta que el estudiante ha ejecutado en esta traza<br>**associatedActionState**: especifica el estado de ejecución concreto de la acción correspondiente | **belongsToStudent**<br>**initialTime**<br>**finalTime** |
| **Procedure_Trace**<br>Describe el registro temporal de una secuencia de acciones (procedimiento) llevada a cabo por un estudiante durante su aprendizaje | **Student_Trace_ Related** | **actionTraceSequence**: especifica la secuencia de trazas de acción de que consta la traza de procedimiento asociada | Ídem **Action_Trace** |
| **Activity_Trace**<br>Describe el registro temporal de una cierta actividad realizada por el estudiante durante su aprendizaje | **Student_Trace_ Related** | **consistsOfProcedureTraces**: especifica las trazas de procedimiento de que consta<br>**idActivity**: identificador de la actividad asociada<br>**belongsToSession**: especifica la traza de sesión a la que está vinculada la traza de actividad<br>**associatedActivityState**: especifica el estado de ejecución concreto de la actividad correspondiente | Ídem **Action_Trace** |

Tabla A.6: Descripción de la ontología **Student_Trace** (cont.)

| CLASE | SUPERCLASE | PROPIEDADES DEFINIDAS | PROPIEDADES HEREDADAS |
|---|---|---|---|
| **Session_Trace**<br>Describe el registro temporal de una sesión de aprendizaje de un estudiante | **Student_Trace_ Related** | **consistsOfActivityTraces**: especifica las trazas de actividades que constituyen la traza de la sesión<br>**idSession**: identificador de la sesión asociada a la traza<br>**associatedSessionState**: especifica el estado de ejecución concreto de la sesión correspondiente | Ídem **Action_Trace** |
| **Variable_Trace**<br>Describe el registro temporal de alguno de los aspectos que, durante el aprendizaje del estudiante, están continuamente variando. Por ejemplo, la mirada | **Student_Trace_ Related** | **tracedVariable**: concepto asociado a la variable trazada | Ídem **Action_Trace** |
| **Emotional_Variable_Trace**<br>Describe el registro temporal de alguna variable que debe ser chequeada con una determinada frecuencia para obtener el valor de un aspecto del estado emocional del estudiante (por ejemplo, su nivel de estrés) | **Variable_Trace** | | **belongsToStudent**<br>**initialTime**<br>**finalTime**<br>**tracedVariable** |
| **Emotional_Trace**<br>Especifica el registro temporal de una cierta emoción | **Student_Trace_ Related** | **associatedEmotionalState**: especifica el estado de una cierta emoción medida en el estudiante durante su aprendizaje<br>**containsEmotionalVariableTraces**: especifica una o más trazas de variables emocionales que es necesario registrar para obtener una medida de algún aspecto emocional del estudiante como, por ejemplo, de su nivel de estrés, distracción, etc. | Ídem **Action_Trace** |

Tabla A.6: Descripción de la ontología **Student_Trace** (cont.)

| CLASE | SUPERCLASE | PROPIEDADES DEFINIDAS | PROPIEDADES HEREDADAS |
|---|---|---|---|
| **Objective_Trace**<br>Especifica el registro temporal de un cierto estado de objetivo durante el aprendizaje de un estudiante | **Student_Trace_ Related** | **isOfActionTrace**: especifica la traza de acción que ha permitido alcanzar esta traza de objetivo<br>**associatedObjectiveState**: especifica el estado de adquisición (alcanzado o no) de un cierto objetivo asociado al instante de esta traza | Ídem **Action_Trace** |
| **Objective_History_Objetivo**<br>Especifica el registro temporal de los sucesivos estados por los que pasa un cierto objetivo durante el aprendizaje de un estudiante. Esta información es útil durante la fase de diagnóstico pedagógico del estudiante | **Student_Trace_ Related** | **isConstitutedByObjectiveTraces**: especifica una o más trazas de las que consta la traza del histórico de un determinado objetivo | Ídem **Action_Trace** |

Tabla A.7: Descripción de la ontología **Student_Monitoring**

| CLASE | SUPERCLASE | PROPIEDADES DEFINIDAS | PROPIEDADES HEREDADAS |
|---|---|---|---|
| **Student_Monitoring_Strategy** Describe aquellas características que definen cómo el módulo de tutoría realizará el seguimiento del estudiante durante su aprendizaje | **owl:Thing** | **monitoringOfStudent**: especifica el estudiante objeto de seguimiento por parte del módulo de tutoría | |
| **Variable_Monitoring_Strategy** Describe las características que definirán el seguimiento de ciertos aspectos variables del estudiante durante su aprendizaje (por ejemplo, su mirada) | **Student_Monitoring_ Strategy** | **samplingFrequency**:especifica el intervalo de tiempo regular entre varios chequeos realizados del valor de una cierta variable observada en el comportamiento del estudiante<br>**samplingVariable**: especifica la variable examinada en el comportamiento del estudiante durante su aprendizaje | **monitoringOfStudent** |

## A.2 Descripción de la extensión de la ontología de Modelado del Estudiante para el prototipo de demostración I

Para adaptar la ontología general de Modelado del estudiante a entornos de aprendizaje de GUIs, ha sido necesaria su extensión y especialización. En las tablas siguientes se detallan las modificaciones realizadas en las jerarquías de la ontología general para su aplicación a este tipo de entornos.

Tabla A.8: Descripción de subclases añadidas a la jerarquía **Object**

| CLASE | SUPERCLASE | PROPIEDADES DEFINIDAS | PROPIEDADES HEREDADAS |
|---|---|---|---|
| **Computer_Interface_Object**<br>Objeto intrínseco al computador que puede ser manejado por un estudiante desde un punto de vista lógico | **Object** | | **descriptorObject**<br>**objectPosition**<br>**idObject** |
| **File**<br>Conjunto de datos con un nombre asociado para poder ser referenciado de forma lógica | **Computer_Interface_Object** | | Ídem **Computer_Interface_Object** |
| **Directory**<br>Objeto que contiene información jerarquizada sobre uno o más archivos o directorios. Tiene asociado un nombre para ser referenciado de forma lógica | **Computer_Interface_Object** | | Ídem **Computer_Interface_Object** |
| **Widget**<br>Objeto usado para diseñar interfaces gráficas de usuario | **Computer_Interface_Object** | | Ídem **Computer_Interface_Object** |
| **Container**<br>Objeto gráfico usado para ubicar otros componentes estableciéndose así una jerarquía de objetos contenedores. El contenedor de nivel superior viene definido por el marco ventana | **Widget** | | Ídem **Computer_Interface_Object** |

Tabla A.8: Descripción de subclases añadidas a la jerarquía **Object** (cont.)

| CLASE | SUPERCLASE | PROPIEDADES DEFINIDAS | PROPIEDADES HEREDADAS |
|---|---|---|---|
| **Container_Window**<br>Objeto gráfico de una aplicación usado como contenedor de nivel superior que incluirá otros componentes gráficos | **Container** | **height** y **width**: establecen el tamaño del contenedor<br>**xCoordinate** y **yCoordinate**: establecen las coordenadas bidimensionales de la ventana | Ídem **Computer_Interface_Object** |
| **Frame**<br>Ventana con una barra de título y con los botones para su manipulación que puede visualizarse en cualquier parte del escritorio | **Container_ Window** | | **descriptorObject**<br>**idObject**<br>**objectPosition**<br>**height**<br>**width**<br>**xCoordinate**<br>**yCoordinate** |
| **Window**<br>Ventana sin una barra de título y sin los botones para su manipulación, que puede visualizarse en cualquier parte del escritorio | **Container_ Window** | | Ídem **Frame** |
| **Dialog**<br>Objeto contenedor que permite visualizar una caja de diálogo personalizada | **Container_ Window** | | Ídem **Frame** |

Tabla A.8: Descripción de subclases añadidas a la jerarquía **Object** (cont.)

| CLASE | SUPERCLASE | PROPIEDADES DEFINIDAS | PROPIEDADES HEREDADAS |
|---|---|---|---|
| **Component**<br>Objeto contenedor atómico o destinado a contener otros componentes que pertenece a un contenedor de nivel superior | **Container** | **alignmentX** y **alignmentY**: establecen la localización del componente con respecto al contenedor al que pertenece | Ídem **Computer_Interface_Object** |
| **Menu_Bar**<br>Componente a la que se añaden alineados uno o más menús que puede seleccionar el usuario | **Component** | **isBorderPainted**: establece si el borde de la barra debe pintarse<br>**isSelected**: establece si la barra tiene un componente seleccionado | **descriptorObject**<br>**idObject**<br>**objectPosition**<br>**alignmentX**<br>**alignmentY** |
| **Popup_Menu**<br>Menú que aparece cuando el usuario selecciona un elemento de una barra menús | **Component** | **invoker**: componente que lo invoca<br>**listMenuItem**: elementos del menú | Ídem **Menu_Bar** |
| **File_Chooser**<br>Son un tipo de cajas de diálogo estándar que permiten al usuario seleccionar una unidad de disco, un directorio, una extensión de archivo y un nombre de archivo | **Component** | **chooseFileFilter**: lista de filtros a elegir<br>**acceptAllFileFilter**: establece si el filtro de "todos los archivos"se incluye en la lista ‚Archivos de tipo"<br>**multiSelectionEnabled**: establece si puede ser seleccionado más de un archivo a la vez<br>**selectedFiles**: nombre de los archivos seleccionados cuando está permitida su selección múltiple<br>**currentDirectory**: nombre del directorio actual en la selección | Ídem **Menu_Bar** |

Tabla A.8: Descripción de subclases añadidas a la jerarquía **Object** (cont.)

| CLASE | SUPERCLASE | PROPIEDADES DEFINIDAS | PROPIEDADES HEREDADAS |
|---|---|---|---|
| **Combo_Box**<br>Componente formado por una caja de texto y una lista desplegable de la que se puede seleccionar un valor | **Component** | **isEditable**: establece si el usuario puede escribir información en la caja de texto o bien, seleccionar un elemento de la lista (lista editable) o sólo puede seleccionar elementos de la lista (lista no editable)<br>**maximumRowCount**: establece el número de filas de elementos que muestra la lista | Ídem **Menu_Bar** |
| **List**<br>Componente que visualiza un conjunto de elementos de los que el usuario puede elegir uno | **Component** | **listData**: establece los elementos de la lista<br>**selectedValue**: establece el valor o valores de la lista seleccionados<br>**selectionMode**: establece si se pueden seleccionar uno o más elementos de la lista<br>**layoutOrientation**: establece si los elementos de la lista son visualizados en una sola columna o en múltiples columnas | Ídem **Menu_Bar** |
| **Abstract_Button**<br>Componente que define el comportamiento común de los botones asociados a una interfaz gráfica y a los elementos de un menú | **Component** | **textButton**: establece el texto asociado al botón | Ídem **Menu_Bar** |
| **Button**<br>Botón de pulsación que permite al usuario ejecutar su acción asociada | **Component** | | **descriptorObject**<br>**idObject**<br>**objectPosition**<br>**alignmentX**<br>**alignmentY**<br>**textButton** |

Tabla A.8: Descripción de subclases añadidas a la jerarquía **Object** (cont.)

| CLASE | SUPERCLASE | PROPIEDADES DEFINIDAS | PROPIEDADES HEREDADAS |
|---|---|---|---|
| **Menu_Item**<br>Elemento de un menú (elemento de una barra de menús u opción de un menú desplegable) | **Abstract_Button** | | Ídem **Button** |
| **Menu**<br>Menú de una barra de menús. Está formado principalmente por elementos **Menu_Item** | **Menu_Item** | | Ídem **Button** |
| **Radio_Button_Menu_Item**<br>Componente (opción) que forma parte de un grupo de elementos de un menú desplegable y de los cuales sólo puede ser seleccionado uno de ellos | **Menu_Item** | | Ídem **Button** |

Tabla A.9: Descripción de subclases añadidas a la jerarquía **Knowledge_Object**

| CLASE | SUPERCLASE | PROPIEDADES DEFINIDAS | PROPIEDADES HEREDADAS |
|---|---|---|---|
| **Position**<br>Describe la situación de una persona u objeto en el espacio geométrico | **Concept** | | Ídem **Concept** |
| **Absolute_Position**<br>Describe la situación de un objeto en un espacio bidimensional mediante coordenadas | **Position**<br>**Variable** | **coordenatex**: especifica la coordenada respecto al eje X<br>**coordenatey**: especifica la coordenada respecto al eje Y | Ídem **Concept**<br>**representedConcept**<br>**variableValue** |
| **Object_Relative_Position**<br>Describe la situación de objeto en el espacio bidimensional tomando como referencia un determinado objeto | **Position** | **positionObject**: especifica el objeto respecto al cual se referencia la posición | Ídem **Concept** |
| **Relation**<br>Representa una conexión o correspondencia entre conceptos o propiedades (ya definido) | **Structural_**<br>**Knowledge** | **domain**<br>**range**<br>**cardinality**<br>**reflexive**<br>**symmetrical**<br>**transitive** | Ídem **Concept** |

Tabla A.9: Descripción de subclases añadidas a la jerarquía **Knowledge_Object** (cont.)

| CLASE | SUPERCLASE | PROPIEDADES DEFINIDAS | PROPIEDADES HEREDADAS |
|---|---|---|---|
| **Is_Son_Of**<br>Establece una relación jerárquica ascendente entre objetos. Por ejemplo, entre un subdirectorio de un determinado directorio padre y éste | **Relation** | | **descriptorObject**<br>**idObject**<br>**domain**<br>**range**<br>**cardinality**<br>**reflexive**<br>**symmetrical**<br>**transitive** |
| **Is_Father_Of**<br>Establece una relación jerárquica descendente entre objetos. Por ejemplo, entre un directorio y uno de los archivos o subdirectorios que contiene | **Relation** | | Ídem **Is_Son_Of** |
| **Aggregation**<br>Relación que establece la anexión de un objeto en otro | **Relation** | | Ídem **Is_Son_Of** |
| **Activation**<br>Relación que establece la habilitación o no de un objeto en un instante de la navegación | **Relation** | | Ídem **Is_Son_Of** |
| **Activated**<br>Relación que establece si un objeto está habilitado en un instante de la navegación | **Activation** | | Ídem **Is_Son_Of** |

Tabla A.9: Descripción de subclases añadidas a la jerarquía **Knowledge_Object** (cont.)

| CLASE | SUPERCLASE | PROPIEDADES DEFINIDAS | PROPIEDADES HEREDADAS |
|---|---|---|---|
| **Inhibited**<br>Relación que establece si un objeto no está habilitado en un instante de la navegación | **Activation** | | Ídem **Is_Son_Of** |
| **Selection**<br>Relación que establece la selección o no de un objeto de la aplicación | **Relation** | | Ídem **Is_Son_Of** |
| **Selected**<br>Relación que establece la selección de un objeto de la aplicación | **Selection** | | Ídem **Is_Son_Of** |
| **Not_Selected**<br>Relación que establece que un objeto de la aplicación no está seleccionado | **Selection** | | Ídem **Is_Son_Of** |
| **Navegate_To**<br>Relación que establece la conexión de un objeto con otro de la aplicación | **Relation** | | Ídem **Is_Son_Of** |
| **To_Be_In_Absolute_Position**<br>Describe como condición de la acción la posición absoluta en la que debe estar la entidad asociada a esta condición | **To_Be_In_Position** | **position**: establece la posición absoluta definida como coordenadas donde se sitúa la entidad asociada a esta condición | **idObject**<br>**descriptorObject**<br>**entity** |

Tabla A.9: Descripción de subclases añadidas a la jerarquía **Knowledge_Object** (cont.)

| CLASE | SUPERCLASE | PROPIEDADES DEFINIDAS | PROPIEDADES HEREDADAS |
|---|---|---|---|
| **To_Be_In_Object_Relative_Position** Describe como condición de la acción la posición relativa a un objeto en la que debe estar la entidad asociada a esta condición | **To_Be_In_Position** | | Ídem **To_Be_In_Absolute_Position** |

## A.3 Descripción de la extensión de la ontología de Modelado del Estudiante para el prototipo de demostración II

Para adaptar la ontología general de Modelado del estudiante a entornos Virtuales de Entrenamiento de tipo procedimental, ha sido necesaria su extensión y especialización. En las tablas siguientes se detallan las modificaciones realizadas en las jerarquías de la ontología general para su aplicación a este tipo de entornos.

Tabla A.10: Descripción de subclases de **Structural_Knowledge** añadidas para EVs

| CLASE | SUPERCLASE | PROPIEDADES DEFINIDAS | PROPIEDADES HEREDADAS |
|---|---|---|---|
| **Map**<br>Conjunto de escenarios que constituyen en el espacio geométrico el medio en el que se desarrolla la actividad | **Concept** | **consistsOfScenarios**: especifica el conjunto de escenarios del mapa | **descriptorObject**<br>**idObject** |
| **Locating_In_Mental_Map**<br>Orientación mental de una persona tomando ciertos puntos u objetos como referencia | **Concept** | **locatingOrigin**: especifica la posición inicial del camino durante el cuál el estudiante debe ubicarse mentalmente<br>**locatingDestination**: especifica la posición final del camino durante el cuál el estudiante ubicarse | Ídem **Map** |
| **Eyesight**<br>Describe el campo que se descubre desde un punto | **Concept** | | Ídem **Mapa** |
| **Absolute_Position**<br>Describe la situación de una persona u objeto en el espacio geométrico mediante coordenadas tridimensionales | **Position**<br>**Variable** | **coordenatex**: especifica la coordenada respecto al eje X<br>**coordenatey**: especifica la coordenada respecto al eje Y<br>**coordenatez**: especifica la coordenada respecto al eje Z | Ídem **Map**<br>**representedConcept**<br>**variableValue** |
| **Area_Position**<br>Describe la situación de una persona u objeto en el espacio geométrico referida a un cierto subescenario (área) | **Position** | **positionArea**: especifica el subescenario respecto al cual se referencia la posición | Ídem **Map** |

Tabla A.10: Descripción de subclases de **Structural_Knowledge** añadidas para EVs (cont.)

| CLASE | SUPERCLASE | PROPIEDADES DEFINIDAS | PROPIEDADES HEREDADAS |
|---|---|---|---|
| **Virtual_Reality_Object**<br>Describe un objeto característico de espacios virtuales | **Object** | | **descriptorObject**<br>**idObject**<br>**objectPosition** |
| **Scenario**<br>Describe el lugar geométrico donde se desarrolla una cierta actividad virtual | **Virtual_Reality_Object** | **isComposedOfSubscenarios**: especifica las partes en las que divide el escenario<br>**isConnectedOfSubscenarios**: identifica los posibles escenarios a los que se puede acceder a partir de éste | Ídem **Virtual_Reality_Object** |
| **Subscenario**<br>Describe uno de los subespacios geométricos pertenecientes a un escenario en el que se puede realizar una cierta actividad virtual | **Virtual_Reality_Object** | **containsObjects**: especifica los objetos geométricos que pertenecen al subescenario<br>**isConnectedToSubscenarios**: identifica los posibles subescenarios a los que se puede acceder a partir de éste | Ídem **Virtual_Reality_Object** |
| **Geometric_Object**<br>Describe un objeto perteneciente a un escenario en el espacio geométrico de un entorno virtual | **Virtual_Reality_Object** | **belongsToSubscenario**: especifica el subescenario al que pertenece el objeto geométrico<br>**isManipulable**: especifica si el objeto de un cierto subescenario se puede o no manejar | Ídem **Virtual_Reality_Object** |
| **Simple_Object**<br>Describe un objeto geométrico sencillo, no divisible en otros objetos geométricos | **Geometric_Object** | | **descriptorObject**<br>**idObject**<br>**objectPosition**<br>**isManipulable** |

Tabla A.10: Descripción de subclases de **Structural_Knowledge** añadidas para EVs (cont.)

| CLASE | SUPERCLASE | PROPIEDADES DEFINIDAS | PROPIEDADES HEREDADAS |
|---|---|---|---|
| **Compound_Object**<br>Describe un objeto geométrico que está formado por más de un objeto simple | **Geometric_Object** | **isComposedOfObjects**: especifica los objetos simples geométricos de los que está constituido | Ídem **Simple_Object** |
| **Part_Of**<br>Relación para establecer que un objeto forma parte de otro objeto | **Relation** | | Ídem **Is_Son_Of** |

Tabla A.11: Descripción de subclases de **Procedural_Knowledge** añadidas para EVs

| CLASE | SUPERCLASE | PROPIEDADES DEFINIDAS | PROPIEDADES HEREDADAS |
|---|---|---|---|
| **Way**<br>Dirección que se sigue para llegar a un lugar a partir de otro | **Concept** | **startOfWay**: especifica la posición inicial del camino<br>**endOfWay**: especifica la posición final del camino | **descriptorObject**<br>**idObject** |
| **Strech**<br>Parte en la que se divide un camino | **Concept** | **pointStrechBeginning**: especifica la posición inicial del tramo<br>**pointStrechEnd**: especifica la posición final del tramo | Ídem **Way** |
| **Trajectory**<br>Línea que se sigue en el espacio por un punto que se mueve | **Concept** | **distance**: distancia a recorrer en la trayectoria<br>**startOfTrajectory**: especifica la posición inicial de la trayectoria<br>**endOfTrajectory**: especifica la posición final de la trayectoria<br>**speed**: especifica la velocidad de la trayectoria | Ídem **Way** |
| **To_Be_In_Area**<br>Describe como condición de la acción la posición relativa a un subescenario o área en la que debe estar la entidad asociada a esta condición | **To_Be_In_Position** | | **descriptorObject**<br>**idObject**<br>**entity**<br>**position** |

Tabla A.12: Descripción de subclases de **Punctual_Action** añadidas para EVs

| CLASE | SUPERCLASE | PROPIEDADES DEFINIDAS | PROPIEDADES HEREDADAS |
|---|---|---|---|
| **According_To_Involved_Characters**<br>Describe una acción de acuerdo al número de participantes que la realizan | **Punctual_Action** | | **descriptorObject**<br>**idObject**<br>**operatorName**<br>**preconditions**<br>**consequences**<br>**usesObjectsAsTool**<br>**executionWay**<br>**qualityFunction**<br>**role**<br>**achievedGoals** |
| **According_To_Interaction_With_Object**<br>Describe una acción de acuerdo a la existencia o no de contacto del individuo con algún objeto al realizarla | **Punctual_Action** | | Ídem<br>**According_To_Involved_Characters** |
| **According_To_Fulfilment_Level**<br>Describe una acción según la posibilidad de diagnosticar la calidad en la obtención de la meta de la acción o no | **Punctual_Action** | | Ídem<br>**According_To_Involved_Characters** |
| **According_To_Communication_Type**<br>Describe una acción de acuerdo al modo de comunicación, verbal o no verbal, existente entre personas durante la acción | **Punctual_Action** | | Ídem<br>**According_To_Involved_Characters** |

Tabla A.13: Descripción de acciones **According_To_Communication_Type** añadidas para EVs

| CLASE | SUPERCLASE | PROPIEDADES DEFINIDAS | PROPIEDADES HEREDADAS |
|---|---|---|---|
| **According_To_Communication_Type**<br>Describe una acción de acuerdo al modo de comunicación, verbal o no verbal, existente durante su ejecución | **Punctual_Action** | | **descriptorObject**<br>**idObject**<br>**operatorName**<br>**preconditions**<br>**consequences**<br>**usesObjectsAsTool**<br>**executionWay**<br>**qualityFunction**<br>**achievedGoals**<br>**role** |
| **Verbal**<br>Describe una acción en la que existe comunicación hablada entre varias personas | **According_To_Communication_Type**<br>**Not_Interaction_With_Object**<br>**Comunication** | | Ídem<br>**According_To_Communication_Type** |
| **Order**<br>Describe una acción verbal, por ejemplo, cuando el tutor proporciona una pista o una instrucción al estudiante en la realización de una actividad | **Verbal** | **order**: especifica la orden proporcionada en este tipo de accción<br>**orderObjects**: especifica los objetos de conocimiento involucrados en la orden | Ídem<br>**According_To_Communication_Type** |

Tabla A.13: Descripción de acciones **According_To_Communication_Type** añadidas para EVs (cont.)

| CLASE | SUPERCLASE | PROPIEDADES DEFINIDAS | PROPIEDADES HEREDADAS |
|---|---|---|---|
| **Ask**<br>Describe la acción verbal de preguntar. Por ejemplo, cuando un estudiante plantea una pregunta acerca de la siguiente acción a realizar en una actividad | **Verbal** | **question**: especifica la pregunta proporcionada en este tipo de accción<br>**questionObjects**: especifica los objetos de conocimiento involucrados en la pregunta | Ídem **According_To_Communication_Type** |
| **Answer**<br>Describe la acción verbal de responder | **Verbal** | **answer**: especifica la respuesta proporcionada en este tipo de accción | Ídem **According_To_Communication_Type** |
| **Not_Verbal**<br>Describe una acción en la que no existe comunicación hablada entre varias personas (por ejemplo, acciones gesticulares como saludar) | **According_To_ Communication_Type Not_Interaction_ With_Object** | | Ídem **According_To_Communication_Type** |
| **Gesture**<br>Describe una acción en la que se realiza un gesto, por ejemplo, facial o con las manos | **Not_Verbal** | | Ídem **According_To_Communication_Type** |

Tabla A.13: Descripción de acciones **According_To_Communication_Type** añadidas para EVs (cont.)

| CLASE | SUPERCLASE | PROPIEDADES DEFINIDAS | PROPIEDADES HEREDADAS |
|---|---|---|---|
| **Gesture_Facial**<br>Describe una acción en la que se realiza un gesto con la cara. Puede ser una acción del tipo mirar (individual) o mostrar una emoción, afirmar, negar etc., (individuales o colectivas) | **Gesture** | | Ídem<br>**According_To_Communication_Type** |
| **Gesture_With_Hands**<br>Describe una acción colectiva, en la que se realiza un gesto con las manos como saludar, apuntar, etc. | **Gesture**<br>**Communication** | | Ídem<br>**According_To_Communication_Type** |

Tabla A.14: Descripción de acciones **According_To_Interaction_With_Object** añadidas para EVs

| CLASE | SUPERCLASE | PROPIEDADES DEFINIDAS | PROPIEDADES HEREDADAS |
|---|---|---|---|
| **According_To_Interaction_With_Object**<br>Describe una acción de acuerdo a la existencia o no de algún tipo de interacción del individuo con objetos al realizarla | **Punctual_Action** | | **descriptorObject**<br>**idObject**<br>**operatorName**<br>**preconditions**<br>**consequences**<br>**usesObjectsAsTool**<br>**executionWay**<br>**qualityFunction**<br>**achievedGoals**<br>**role** |
| **Interaction_With_Object**<br>Describe una acción que implica el contacto del individuo que realiza la acción con algún objeto | **According_To_Interaction_With_Object**<br>**Individual** | **isAppliedToObjects**: especifica los objetos con los que interacciona el individuo en esta acción | Ídem<br>**According_To_Interaction_With_Object** |
| **Modifies_Object_State**<br>Describe una acción que implica una alteración del estado del objeto que se está manipulando | **Interaction_With_**<br>**Object** | | Ídem<br>**According_To_Interaction_With_Object**<br>**isAppliedToObjects** |
| **Not_Modifies_Object_Position**<br>Describe una acción que no implica la modificación de la posición del objeto que se está manipulando | **Modifies_Object_ State**<br>**Action_With_Total_**<br>**Level_Fulfilment** | | Ídem **Modifies_Object_State** |

Tabla A.14: Descripción de subclases de acciones **According_To_Interaction_With_Object** añadidas para EVs (cont.)

| CLASE | SUPERCLASE | PROPIEDADES DEFINIDAS | PROPIEDADES HEREDADAS |
|---|---|---|---|
| **Push**<br>Describe la acción de pulsar un botón. Por ejemplo, pulsar el botón izquierdo del ratón para realizar la selección de un objeto gráfico | **Not_Modifies_Object_ Position** | | Ídem **Modifies_Object_State** |
| **Open**<br>Describe la acción de apertura de un objeto, por ejemplo una puerta | **Not_Modifies_Object_ Position** | | Ídem **Modifies_Object_State** |
| **Close**<br>Describe la acción de cerrar un objeto, por ejemplo una puerta | **Not_Modifies_Object_ Position** | | Ídem **Modifies_Object_State** |
| **Modifies_Object_Position**<br>Describe una acción que implica la modificación de la posición del objeto que se está manipulando | **Modifies_Position**<br>**Modifies_Object_ State** | | Ídem **Modifies_Object_State**<br>**originPos destinationPos** |
| **Move_2D_Object**<br>Describe una acción que implica la modificación de la posición del objeto en un espacio bidimensional. Por ejemplo, empujar un objeto o arrastrar un objeto | **Modifies_Object_ Position** | | Ídem **Modifies_Object_Position** |

Tabla A.14: Descripción de subclases de acciones **According_To_Interaction_With_Object** añadidas para EVs (cont.)

| CLASE | SUPERCLASE | PROPIEDADES DEFINIDAS | PROPIEDADES HEREDADAS |
|---|---|---|---|
| **Drag_Object**<br>Describe una acción que implica la modificación de la posición del objeto que se está arrastrando. Por ejemplo, al arrastrar el ratón de una posición inicial a una final en un GUI | **Modifies_Object_ Position** | | Ídem **Modifies_Object_Position** |
| **Move_3D_Object**<br>Describe la acción de desplazamiento del estudiante con el objeto con el que interacciona en un espacio 3D | **Modifies_Object_ Position Individual** | | Ídem **Modifies_Object_Position** |
| **Modifies_Relation_With_Object**<br>Describe una acción que implica la modificación de la relación entre el estudiante y un objeto o entre los objetos que se están manipulando | **Interaction_With_Object Action_With_Total_ Level_Fulfilment** | | Ídem **Modifies_Object_Position** |
| **Modifies_Relation_Among_Objects**<br>Describe una acción que implica modificar la relación entre objetos (por ejemplo, echar o insertar un objeto en otro) | **Modifies_Relation_ With_Object** | | Ídem **Modifies_Object_State** |

Tabla A.14: Descripción de subclases de acciones **According_To_Interaction_With_Object** añadidas para EVs (cont.)

| CLASE | SUPERCLASE | PROPIEDADES DEFINIDAS | PROPIEDADES HEREDADAS |
|---|---|---|---|
| **Object_Inside_Other**<br>Describe una acción que implica introducir un objeto en otro (por ejemplo, echar o insertar un objeto en otro) | **Modifies_Relation_ Among_Objects** | | Ídem **Modifies_Object_State** |
| **Put_In**<br>Describe una acción que implica introducir un objeto en el interior de otro | **Object_Inside_Other** | | Ídem **Modifies_Object_State** |
| **Get_Out_Object_Of_Other**<br>Describe una acción que implica extraer un objeto de otro | **Modifies_Relation_ Among_Objects** | | Ídem **Modifies_Object_State** |
| **Modifies_Relation_Student_Object**<br>Describe una acción que implica la modificación de la relación entre el estudiante y el objeto que se está manipulando | **Modifies_Relation_ With_Objeto** | **actionPos**: especifica la posición del objeto involucrado en la acción | Ídem **Modifies_Object_State** |
| **Pickup_Object**<br>Describe una acción que implica coger un objeto de una determinada posición | **Modifies_Relation_ Student_Object** | | Ídem **Modifies_Object_State** **actionPos** |
| **Drop_Object**<br>Describe una acción que implica dejar un objeto de una determinada posición | **Modifies_Relation_ Student_Object** | | Ídem **Modifies_Object_State** **actionPos** |

Tabla A.14: Descripción de subclases de acciones **According_To_Interaction_With_Object** añadidas para EVs (cont.)

| CLASE | SUPERCLASE | PROPIEDADES DEFINIDAS | PROPIEDADES HEREDADAS |
|---|---|---|---|
| **Without_Effect_On_Object**<br>Describe una acción de interacción con un objeto sin efecto sobre él | **Interaction_With_Object**<br>**Action_With_Total_**<br>**Level_Fulfilment** | | Ídem **Modifies_Object_State**<br>**actionPos** |
| **Crash_Into_Object**<br>Describe una acción de interacción con un objeto con el que un estudiante choca pero sin efecto sobre el objeto | **Without_Effect_On_**<br>**Object** | | Ídem **Modifies_Object_State**<br>**actionPos** |
| **Not_Interaction_With_Object**<br>Describe una acción que no implica el contacto del individuo que realiza la acción con objetos | **According_To_Interaction_**<br>**With_Object** | | Ídem<br>**According_To_Interaction_With_Object** |
| **Modifies_Student_Position**<br>Describe una acción que no implica interacción con objeto pero sí modificación de la posición del estudiante | **Not_Interaction_With_**<br>**Object**<br>**Individual**<br>**Modifies_Position** | | Ídem **Not_Interaction_With_Object**<br>**originPos**<br>**destinationPos** |
| **Move**<br>Describe una acción sin interacción con objeto y en la que el estudiante se desplaza | **Modifies_Student_**<br>**Position** | | Ídem **Modifies_Student_Position** |

Tabla A.14: Descripción de subclases de acciones **According_To_Interaction_With_Object** añadidas para EVs (cont.)

| CLASE | SUPERCLASE | PROPIEDADES DEFINIDAS | PROPIEDADES HEREDADAS |
|---|---|---|---|
| **Movement_In_Position**<br>Describe una acción sin interacción con objeto y con movimiento del estudiante sin modificar su posición actual (por ejemplo, agacharse, levantarse, saltar, etc) | **Not_Interaction_With_Object**<br>**Action_With_Total_Level_Fulfilment Individual** | | Ídem **Not_Interaction_With_Object** |

Tabla A.15: Descripción de acciones **According_To_Involved_Characters** añadidas para EVs

| CLASE | SUPERCLASE | PROPIEDADES DEFINIDAS | PROPIEDADES HEREDADAS |
|---|---|---|---|
| **According_To_Involved_Characters**<br>Describe una acción de acuerdo al número de participantes al realizarla | **Punctual_Action** | | **descriptorObject**<br>**idObject**<br>**operatorName**<br>**preconditions**<br>**consequences**<br>**usesObjectsAsTool**<br>**executionWay**<br>**qualityFunction**<br>**role**<br>**achievedGoals** |
| **Individual**<br>Describe una acción en la que participa sólo un personaje | **According_To_Involved_Characters** | | Ídem<br>**According_To_Involved_Characters** |
| **Collective**<br>Describe una acción en la que participan varios personajes | **According_To_Involved_Characters** | | Ídem<br>**According_To_Involved_Characters** |
| **Collaborative**<br>Describe una acción colectiva en la que existe coordinación entre los personajes involucrados en ella. Por ejemplo, dar un objeto a otra persona, transportar un objeto entre varias personas, etc. | **Collective** | | Ídem<br>**According_To_Involved_Characters** |

Tabla A.15: Descripción de subclases de acciones **According_To_Involved_Characters** añadidas para EVs (cont.)

| CLASE | SUPERCLASE | PROPIEDADES DEFINIDAS | PROPIEDADES HEREDADAS |
|---|---|---|---|
| **Collaborative_Modifies_Object_Position**<br>Describe una acción colaborativa en la que hay interacción con un objeto al menos y se modifica su posición | **Collaborative** | | Ídem **Modifies_Object_Position** |
| **Transport_Object_Among_Several**<br>Describe una acción colaborativa en la que varias personas transportan un objeto | **Collaborative_Modifies_ Object_Position** | | Ídem **Modifies_Object_Position** |
| **Give_Object_To_Other**<br>Describe una acción colaborativa en la que un individuo entrega un objeto a otro | **Collaborative_Modifies_ Object_Position** | | Ídem **Modifies_Object_Position** |
| **Communication**<br>Describe una acción colectiva en la que existe comunicación entre los personajes involucrados en ella (por ejemplo, gesticular con las manos o acciones verbales como dar una orden, preguntar, responder, etc. | **Collective Action_With_Total_ Level_Fulfilment** | | Ídem **According_To_Involved_Characters** |

Tabla A.16: Descripción de acciones **According_To_Fulfilment_Level** añadidas para EVs

| CLASE | SUPERCLASE | PROPIEDADES DEFINIDAS | PROPIEDADES HEREDADAS |
|---|---|---|---|
| **According_To_Fulfilment_Level**<br>Describe una acción según la posibilidad de diagnosticar la calidad en la obtención de alguna de sus metas o no | **Punctual_Action** | | **descriptorObject**<br>**idObject**<br>**operatorName**<br>**preconditions**<br>**consequences**<br>**usesObjectsAsTool**<br>**executionWay**<br>**qualityFunction**<br>**role**<br>**achievedGoals** |
| **Action_With_Total_Level_Fulfilment**<br>Describe una acción cuya ejecución implicará un diagnóstico en la obtención de sus metas como alcanzadas o no alcanzadas | **According_To_Fulfilment_Level** | | Ídem **According_With_Total_Level_Fulfilment** |
| **Action_With_Partial_Level_Fulfilment**<br>Describe una acción cuya ejecución implicará un diagnóstico de alguna de sus metas con un cierto grado de calidad que dependerá del seguimiento o evaluación de determinados parámetros | **According_To_Fulfilment_Level** | | Ídem **According_To_Fulfilment_Level** |

Tabla A.17: Descripción de subclases de **Performance_State** añadidas para EVs

| CLASE | SUPERCLASE | PROPIEDADES DEFINIDAS | PROPIEDADES HEREDADAS |
|---|---|---|---|
| **Strech_State**<br>Datos obtenidos dinámicamente durante el seguimiento (en tiempo de ejecución) y la evaluación (finalización) del desplazamiento del estudiante en un tramo de su trayectoria | **Performance_ State** | **containsStrech**: especifica el tramo al que pertenecen los datos obtenidos dinámicamente<br>**isRelatedToStrechTrace**: especifica la traza del tramo correspondiente<br>**studentStrechDistance**: especifica la longitud del trayecto realizado por el estudiante en un tramo (entre sus dos puntos extremos)<br>**distanceFactor**: valor de comparación de la longitud del trayecto realizado por el estudiante con respecto a la longitud del tramo óptimo[a]<br>**studentStrechTime**: especifica el tiempo que ha tardado el estudiante en realizar el tramo<br>**timeFactor**: valor de comparación del tiempo que ha tardo en realizar el tramo el estudiante con respecto al tiempo que se ha supuesto razonable para realizar el tramo óptimo<br>[b]<br>**averageDistance**: factor que, mediante un algoritmo ([]), realiza una comparación parcial de los puntos que componen el tramo realizado por el estudiante con respecto a los puntos calculados del tramo óptimo, según la velocidad seguida por el estudiante en el tramo | **stateOfStudent** |

[a] El factor distancia viene definido por la fórmula siguiente: $\text{Factor\_Distancia} = \frac{\text{Distancia\_Tramo\_Alumno}}{\text{Distancia\_Tramo\_Optima}}$ []

[b] El factor tiempo viene definido por la fórmula siguiente: $\text{Factor\_Tiempo} = \frac{\text{Tiempo\_Tramo\_Alumno}}{\text{Tiempo\_Tramo\_Optima}}$ [].

Tabla A.17: Descripción de subclases de **Performance_State** añadidas para EVs (cont.)

| CLASE | SUPERCLASE | PROPIEDADES DEFINIDAS | PROPIEDADES HEREDADAS |
|---|---|---|---|
| **Trajectory_State**<br>Datos obtenidos dinámicamente durante el seguimiento (en tiempo de ejecución) y la evaluación (finalización) del desplazamiento del estudiante durante el desarrollo de una cierta actividad de aprendizaje | **Performance_ State** | **isRelatedToTrajectoryTrace**: especifica la traza de la trayectoria a la que pertenecen los datos obtenidos dinámicamente<br>**accumulatedAverageDistance**: representa la separación real que existe entre los tramos de la trayectoria del estudiante y la óptima. Este factor se obtiene como el total de las distancias medias de cada uno de los tramos de la trayectoria<br>**trajectoryFactor**: especifica un valor para evaluar cuantitativamente la trayectoria al finalizar la actividad. Este factor se obtiene a partir de la distancia media acumulada (DMA), el factor distancia (FD) y el factor tiempo (FT) que se han ido acumulando en los tramos realizados durante la trayectoria[a]<br>**movementAssessment**: especifica una valoración cualitativa asociada a la trayectoria realizada por el estudiante y dependiente del tipo de actividad ejecutada y del entorno virtual. Esta evaluación se realizará tras un estudio de los resultados obtenidos en pruebas sucesivas | **stateOfStudent** |

[a] El factor trayectoria se obtiene con la siguiente fórmula []: $Factor\_Trayectoria = DMA * 0,7 + |1 - FD| * 0,2 + |1 - FT| * 0,1$

Tabla A.18: Descripción de subclases de **Student_Trace** añadidas para EVs

| CLASE | SUPERCLASE | PROPIEDADES DEFINIDAS | PROPIEDADES HEREDADAS |
|---|---|---|---|
| **Student_Action_Trace**<br>Describe el registro temporal de la acción del estudiante; cuándo y qué tipo de acción ha realizado | **Action_Trace** | | **belongsToStudent**<br>**initialTime**<br>**finalTime**<br>**associatedSpecificAction**<br>**associatedActionState** |
| **Movement_Action_Trace**<br>Describe el registro temporal del tipo de acción de desplazamiento ha realizado el estudiante durante su trayectoria | **Student_Action_Trace** | **associatedTrajectory**: especifica la traza de la trayectoria del estudiante a la que pertenece esta traza de movimiento | Ídem **Student_Action_Trace** |
| **Tutor_Action_Trace**<br>Describe el registro temporal del tipo de acción que ha realizado el tutor durante el aprendizaje del estudiante | **Action_Trace** | | Ídem **Student_Action_Trace** |
| **Position_Trace**<br>Describe el registro temporal de la posición del estudiante en un cierto intervalo de tiempo. Durante el aprendizaje, este parámetro continuamente variará | **Variable_Trace** | | **belongsToStudent**<br>**initialTime**<br>**finalTime**<br>**tracedVariable**[a]<br>[a]Esta propiedad tiene restringido sus valores a instancias del concepto **Position** |

Tabla A.18: Descripción de subclases de **Student_Trace** añadidas para EVs (cont.)

| CLASE | SUPERCLASE | PROPIEDADES DEFINIDAS | PROPIEDADES HEREDADAS |
|---|---|---|---|
| **Strech_Trace**<br>Describe el registro temporal de uno de los tramos de que consta la trayectoria del estudiante durante una actividad de aprendizaje | **Student_Trace_ Related** | **associatedStrechState**: especifica el estado de tramo asociada a esta traza<br>**belongsToStrech**: especifica el tramo al que corresponde esta traza<br>**consistsOfPosition**: especifica las dos trazas de posición de que consta la traza de la trayectoria del estudiante<br>**consistsOfStrechTraces**: especifica las trazas de los tramos de que consta la trayectoria seguida por el estudiante<br>**associatedTrajectoryState**: especifica el estado (valoración) de la trayectoria obtenido tras finalizar esta traza<br>**isComparedWith**: especifica la trayectoria óptima con la que se compara la trayectoria del estudiante | **belongsToStudent**<br>**initialTime**<br>**finalTime** |
| **Trajectory_Trace**<br>Especifica la traza de la trayectoria seguida por el estudiante durante el aprendizaje de una cierta actividad | **Student_Trace_ Related** | | Ídem **Strech_Trace** |

Tabla A.19: Descripción de subclases de **Student_Monitoring_Strategy** añadidas para EVs

| CLASE | SUPERCLASE | PROPIEDADES DEFINIDAS | PROPIEDADES HEREDADAS |
|---|---|---|---|
| **Position_Monitoring_Strategy** Describe las características relativas al seguimiento de la posición del estudiante en su desplazamiento durante el aprendizaje | **Variable_Monitoring_ Strategy** | | **samplingFrequency** **samplingVariable**[a] [a]La variable muestreada está restringida a instancias del concepto **Position** |

# REFERENCIAS


(2003a). Ims learning design information model. version 1.0 final specification. Technical report, IMS Global Consortium.

(2003b). Learning design specification. version 1.0 final specification. Technical report, IMS Global Consortium.

Akhras, F. N. and Self, J. A. (2000). System intelligence in constructivist learning. *International Journal of Artificial Intelligence in Education*, 11:334–376.

Alem, L. and R., K. (1996). Intelligent simulation environment an application to air traffic control training. In *Procs. of the first international conference on Advancing simulation technology and training, SimTecT'96*, pages 323–328.

Amokrane, K., Lourdeaux, D., Barthès, J., and Burkhardt, J. (2008a). An intelligent tutoring system for training and learning in a virtual environment for high-risk sites. In *20th IEEE International Conference on Tools with Artificial Intelligence*, pages 185–193.

Amokrane, K., Lourdeaux, D., and Burkhardt, J. (2008b). Hera: Learner tracking in a virtual environment. *The International Journal of Virtual Reality*, 7(3):23–30.

Amokrane, K., Lourdeaux, D., and Burkhardt, J. (2008c). Learner behavior tracking in a virtual environment. In *Proc. of Virtual Reality International Conference*.

Amorim, R. R., Lama, M., Sánchez, E., Riera, A., and Vila, X. A. (2006). A learning design ontology based on the ims specification. *Journal of Educational Technology and Society*, 9:38–57.

Anderson, J. R. (1983). *The Architecture of Cognition*. Harvard University Press, Cambridge, MA, USA.

Anderson, J. R., Boyle, C. F., Corbett, A. T., and Lewis, M. W. (1990). Cognitive modeling and intelligent tutoring. *Artif. Intell.*, 42(1):7–49.

Anderson, J. R., Boyle, C. F., and Yost, G. (1985). The geometry tutor. In *Byte*, pages 1–7.

Anderson, L. W., Krathwohl, D. R., and Bloom, B. S. (2001). *A Taxonomy for Learning, Teaching, and Assessing: A Revision of Bloom's Taxonomy of Educational Objectives*. Pearson Education Group, Boston, MA, USA.

Barr, A., B. M. y. A. R. (1975). The computer as a tutorial laboratory: The stanford bip project. Technical Report 260, Stanford Univ., CA. Inst. for Mathematical Studies in Social Science.

Bechhofer, S., Horrocks, I., Goble, C. A., and Stevens, R. (2001). Oiled: A reason-able ontology editor for the semantic web. In *KI/ÖGAI*, pages 396–408.

Beck, J., Stern, M., and Woolf, B. P. (1997). Using the student model to control problem difficulty. In *In Proceedings of the 6th International Conference on User Modeling*, pages 277–288. Springer-Verlag.

Beck, J. and Woolf, B. P. (1998). Using a learning agent with a student model. In *ITS '98: Proceedings of the 4th International Conference on Intelligent Tutoring Systems*, pages 6–15, London, UK. Springer-Verlag.

Bernaras, A., Laresgoiti, I., and Corera, J. M. (1996). Building and reusing ontologies for electrical network applications. In *ECAI*, pages 298–302.

Berners-Lee, T., Hendler, J., and Lassila, O. The semantic web (berners-lee et. al 2001).

Blázquez, M., M., F.-L., García-Pinar, J. M., and Gomez-Perez, A. (1997). Building ontologies at the knowledge level using the ontology design environment. In *11th International Workshop on Knowledge Acquisition, Modeling and Management (KAW'98)*, Banff, Canada.

Bloom, B. S., Engelhart, M. D., Furst, E. J., Hill, W. H., and Krathwohl, D. R. (1956). *Taxonomy of Educational Objectives: Handbook 1 Cognitive Domain*. Longmans, Green and Co. Ltd., London, UK, USA.

Borst, W. N. (1997). *Construction of Engineering Ontologies*. PhD thesis, University of Tweenty, Enschede, The Netherlands.

Brajnik, G. and Tasso, C. (1994). A shell for developing non-monotonic user modeling systems. *Int. J. Hum.-Comput. Stud.*, 40(1):31–62.

Bray, T., Paoli, J., Sperberg-McQueen, C. M., Maler, E., and Yergeau, F. (2004). Extensible markup language (XML) 1.0 (third edition), W3C recommendation. Technical report, W3C.

Brecht, B. and Jones, M. (1988). Student models: The genetic graph approach. *Int. Journal of Man-Machine Studies*, 28:483–504.

Bredeweg, B. and Breuker, J. (1993). "Device Models" for model-based diagnosis of student behaviour. In Ohlsson, S., editor, *Proceedings of the World Conference on AI and Education*, Charlottesville, VA. AACE.

Bresciani, P., Giorgini, P., Giunchiglia, F., Mylopoulos, J., and Perini, A. (2004). Tropos: An agent-oriented software development methodology. *Autonomous Agents and Multi-Agent Systems*, 8(3):203–236.

Breuker, J. (1990). Eurohelp: developing intelligent help systems, esprit project p280, final report. Technical report, University of Leeds, UK.

Brickley, D. and Guha, R. (2002). RDF Vocabulary Description Language 1.0: RDF Schema, W3C Working Draft. Technical report, W3C, http://www.w3.org./TR/rdf-schema/.

Brown, A. L. (1987). Metacognition, executive control, self-regulation, and other more mysterious mechanisms. In Weinert, F. and Kluwe, R., editors, *Metacognition, motivation, and understanding*, pages 65–116. Erlbaum, Hillsdale, NJ.

Brown, J. S. and VanLehn, K. (1988). Repair theory: A generative theory of bugs in procedural skills. In Collins, A. and Smith, E. E., editors, *Readings in Cognitive Science: A Perspective from Psychology and Artificial Intelligence*, pages 338–361. Kaufmann, San Mateo, CA.

Buche, C., Querrec, R., Loor, P. D., and Chevaillier, P. (2003). Mascaret: Pedagogical multi-agents system for virtual environment for training. *Cyberworlds, International Conference on*, 0:423.

Bull, S., Greer, J., Mccalla, G., Kettel, L., and Bowes, J. (2001). User modelling in i-help: What, why, when and how. In *Vassileva (Eds.), UM2001, User Modeling: Proceedings of the Eighth International Conference*, pages 117–126. Springer.

Bull, S. and Smith, M. (1997). A pair of student models to encourage collaboration. In *Proceedings of the Sixth international Conference UM97*, pages 339–341. Chia Laguna, Italy.

Burton, R. B. (1982). *Diagnosing bugs in a simple procedural skill*, chapter Intelligent tutoring systems, pages 157–183. Academic-Press.

Burton, R. R. and Brown, J. S. (1976). A tutoring and student modelling paradigm for gaming environments. *SIGCSE Bull.*, 8(1):236–246.

Carbonell, J. R. (1970). Ai in cat: An artificial intelligence approach to computer-assisted instruction. *IEEE Transactions on Man-Machine Systems*, 11(4):190–202.

Carmona, C., Castillo, G., and Millán, E. (2008). Designing a dynamic bayesian network for modeling students' learning styles. In *ICALT*, pages 346–350.

Carr, B. and Coldstein, I. (1977). Overlays: a theory of modelling for computer-aided instruction. *International Journal of Man-Machine Studies*, 5:215–236.

Carrasco, M. (2000). E-learning el futuro de la formación on-line. Centro de Diseño Industrial y Artístico de Cataluña.

Chaudhri, V., Farquhar, A., Fikes, R., Karp, P., and Rice, J. (1998). Open knowledge base connectivity (okbc) specification document 2.0.3. sri international and stanford university (ksl). Technical report. See http://www.ai.sri.com/ okbc/okbc-2-0-3.pdf.

Chen, W. and Mizoguchi, R. (1999). Communication content ontology for learner model agent in multi-agent architecture. In *Advanced Research in Computers and Communication in Education, Proc. ICCE'99*, pages 95–102.

Chen, W. and Mizoguchi, R. (2004). Learner model ontology and leaner model agent, cognitive support for learning - imagining the unknown. *Cognitive Support for Learning -Imagining the Unknown*, 2004:189–200.

Clancey, W. The frame of reference problem in the design of intelligent machines, booktitle = The twenty-second Carnegie Symposium on Cognition: Architecturcs for intelligence, year = 1991, editor = K. VanLehn, pages = 357–424, publisher = Lawrence Erlbaum Associates, Inc., address = Hillsdale, NJ,.

Clancey, W. J. (1986). Qualitative student models. pages 381–450.

Collins, A., Brown, J. S., and Newman, S. E. (1989). Cognitive apprenticeship: Teaching the crafts of reading, writing and mathematics. In Resnick, L. B., editor, *Knowing, learning, and instruction: Essays in honor of Robert Glaser*, pages 453–494. Lawrence Erlbaum Associates, Hillsdale, NJ, CA.

Collins, J. A., Greer, J. E., and Huang, S. X. (1996). Adaptive assessment using granularity hierarchies and bayesian nets. In *ITS '96: Proceedings of the Third International Conference on Intelligent Tutoring Systems*, pages 569–577, London, UK. Springer-Verlag.

Conati, C. and Maclaren, H. (2004). Evaluating a probabilistic model of student affect. In *Intelligent Tutoring Systems*, pages 55–66.

Conati, C. and VanLehn, K. (1996). Pola: a student modeling framework for probabilistic on-line assessment of problem solving performance.

Corcho, Ó., Fernández-López, M., Gómez-Pérez, A., and Vicente, Ó. (2002). Webode: An integrated workbench for ontology representation, reasoning, and exchange. In *EKAW*, pages 138–153.

Crowder, N. A. (1959). Automatic tutoring by means of intrinsic programming. In Galanter, E. H., editor, *Automatic teaching: The state of the art*, pages 109–116. Wiley, J. and Sons, Inc., New York.

Dave, H. (1970). *Developing and Writing Behavioral Objectives*. Educational Innovators Press, Tucson, AZ, USA.

Davis, R. (1984). Diagnostic reasoning based on structure and behavior. *Artif. Intell.*, 24(1-3):347–410.

de Antonio, A., Ramírez, J., Imbert, R., and Méndez, G. (2005). Intelligent virtual environments for training: An agent-based approach. In *CEEMAS*, pages 82–91.

De Bruijn, J., Bussler, C., Domingue, J., Fensel, D., Hepp, M., Keller, U., Kifer, M., König-Ries, B., Kopecky, J., Lara, R., Lausen, H., Oren, E., Polleres, A., Roman, D., Scicluna, J., and Stollberg, M. (2005a). Web service modeling ontology (wsmo). W3C Member Submission.

De Bruijn, J., Lausen, H., Krummenacher, R., Polleres, A., Predoiu, L., Kifer, M., and Fensel, D. (2005b). The web service modeling language wsml. Technical report, DERI.

de Kleer, J. (1986). An assumption-based tms. *Artif. Intell.*, 28(2):127–162.

de Kleer, J. and Williams, B. C. (1987). Diagnosing multiple faults. *Artif. Intell.*, 32(1):97–130.

De Kleer, J. and Williams, B. C. (1989). Diagnosis with behavioral modes. In *IJCAI'89: Proceedings of the 11th international joint conference on Artificial intelligence*, pages 1324–1330, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

de Koning, K. and Bredeweg, B. (1998). Using gde in educational systems. In *In Proceedings of the European Conference on Artificial Intelligence (ECAI)*, pages 279–283.

de Koning, K., Bredeweg, B., Breuker, J., and Wielinga, B. J. (2000). Model-based reasoning about learner behaviour. *Artif. Intell.*, 117(2):173–229.

de Koning, K., Breuker, J. A., and Bredeweg, B. (1995). Cognitive diagnosis revisited. In Greer, J., editor, *Proceedings of Artificial Intelligence in Education*, pages 115–122. AACE, Charlottesville, VA.

Dellschaft, K., Engelbrecht, H., Monte, J., Rutenbeck, S., and Staab, S. (2008). Cicero: Tracking design rationale in collaborative ontology engineering. In *ESWC*, pages 782–786.

Derry, S. J. (1992). Metacognitive models of learning and instructional systems design. *Adaptative Learning Environments: Foundations and Frontiers*, 85.

Devedzic, V., Jerinic, L., and Radovic, D. (1999). The get-bits model of intelligent tutoring systems. *Journal of Interactive Learning Research*, 11:411–434.

Dillenbourg, P. and Self, J. (1992). A framework for learner modelling. *Interactive Learning Environments*, 2:111–137.

Dimitrova, V., Self, A., and Brna, P. (1999). Style-olm an interactive diagnosis tool in a terminology learning environment. In *Proceedings of the workshop Open, Interactive, and other Overt Approaches to Learner Modelling, AIED'99*.

Ding, L., Kolari, P., Ding, Z., and Avancha, S. (2005). *Using Ontologies in the Semantic Web: A Survey*. Springer.

Domingue, J. (1998). Tadzebao and webonto: Discussing, browsing, editing ontologies on the web. In *11th Knowledge Acquisition for Knowledge-Based Systems Workshop*.

Dongming Xu, H. W. and Wang, M. (2005). A conceptual model of personalized virtual learning environments. *Expert Syst. Appl.*, 29(3):525–534.

Doyle, J. (1987). A truth maintenance system. pages 259–279.

El-Kechaï, N. (2007). *Suivi et assistance des apprenants dans les environnements virtuels de formation*. PhD thesis, Université de Maine.

El-Kechaï, N. and Després, C. (2006). A plan recognition process, based on a task model, for detecting learner's erroneous actions. In *Intelligent Tutoring Systems, ITS 2006*, pages 329–338, Jhongli, Taïwan.

Elsom-Cook, M. (1990). *Guided Discovery Tutoring: a Framework for ICAI Research*. Paul Chapman, London, UK.

Engler, M., Vrandecic, D., and Sure, Y. (2006). A tool for diligent argumentation: Experiences, requirements and design. In *WETICE*, pages 370–375.

Farquhar, A., Fikes, R., and Rice, J. (1997). The ontolingua server: a tool for collaborative ontology construction. *International Journal of Human-Computer Studies*, pages 707–727.

Farrell, R. G., Anderson, J. R., and Reiser, B. J. (1984). An interactive computer-based tutor for lisp. In *AAAI*, pages 106–109.

Felder, R. M. and Silverman, L. K.

Fernández-López, M., Gómez Pérez, A., and Juristo, N. (1997). Methontology: from ontological art towards ontological engineering. In *Proceedings of the AAAI97 Spring Symposium*, pages 33–40, Stanford, USA.

Fernández López, M., Gómez Pérez, A., and Rojas Amaya, M. D. (2000). Ontology's crossed life cycles. In *EKAW '00: Proceedings of the 12th European Workshop on Knowledge Acquisition, Modeling and Management*, pages 65–79, London, UK. Springer-Verlag.

Forbus, K. D. and de Kleer, J. (1993). *Building problem solvers*. MIT Press, Cambridge, MA, USA.

Gagné, R., Briggs, L., and Wagner, W. (1989). *Principles of Instructional Design*. Rinehart and Winston.

Gascueña, J. M., Fernández-Caballero, A., and González, P. (2006). Domain ontology for personalized e-learning in educational systems. In *ICALT*, pages 456–458.

Geneserth, M. R. and Fikes, R. E. (1992). Knowledge interchange format version 3.0 reference manual. Technical Report Logic Group Report Logic-92-1, Computer Science Department, Stanford University.

Gilmore, D. and Self, J. (1988). The application of machine learning to intelligent tutoring systems. In Self, J., editor, *Artificial Intelligence and Human Learning: Intelligent Computer-Aided Instruction*, pages 179–196. Chapman and Hall, London.

Giraffa, L., Nunes, M. A., and Viccari, R. M. (1997). Multi-ecolological: A learning environment using multi-agent architecture. In *MASTA'97. Multi-Agent System: Theory and Applications*.

Gomez-Perez, A., Angele, J., Fernandez-Lopez, M., Christophides, V., Stutt, A., and Sure, Y. (2002). A survey on ontology tools. OntoWeb deliverable 1.3, Universidad Politecnica de Madrid.

Gomez-Perez, A., Corcho, O., and Fernandez-Lopez, M., editors (2003). *Ontological Engineering: With Examples from the Areas of Knowledge Management, E-Commerce and the Semantic Web. Advanced Information and Knowledge Processing (First Edition)*. Springer-Verlag, Heidelberg.

Gómez Pérez, A. and Rojas Amaya, M. D. (1999). Ontological reengineering for reuse. In *EKAW '99: Proceedings of the 11th European Workshop on Knowledge Acquisition, Modeling and Management*, pages 139–156, London, UK. Springer-Verlag.

Gonzalez, C., Burguillo, J. C., and M., L. (2006). A qualitative comparison of techniques for student modeling in intelligent tutoring systems. In *Frontiers in Education Conference, 36th Annual*, pages 13–18. INSTICC -Institute for Systems and Technologies of Information, Control and Communication.

Greenwood, E. (1973). *Metodología de la investigación social*. Paidós.

Gruber, T. (2008). Ontology (computer science) - definition in encyclopedia of database systems.

Gruber, T. R. (1993). A translation approach to portable ontology specifications. *Knowledge Acquisition*, 5(2):199–220.

Grüninger, M. and Fox, M. (1995). Methodology for the Design and Evaluation of Ontologies. In Skuce, D., editor, *IJCAI'95, Workshop on Basic Ontological Issues in Knowledge Sharing, April 13, 1995*, pages 6.1–6.10.

Grüninger, M. and Fox, M. S. (1994). The role of competency questions in enterprise engineering. In *Proceedings of the IFIP WG5.7 Workshop on Benchmarking - Theory and Practice*.

Gürer, D. W., desJardins, M., and Schlager, M. (1995). Aaai technical report, sri international. Technical report.

Hamscher, W. (1990). Xed: diagnosing devices with hierarchic structure and known component failure modes. In *Proceedings of the sixth conference on Artificial intelligence applications*, pages 48–54, Piscataway, NJ, USA. IEEE Press.

Harrow, A. J. (1972). *A Taxonomy of the Psychomotor Domain: A Guide for Developing Behavioral Objectives*. David McKay Company, Inc., New York, NY, USA.

Hasse, P., Lewen, H., Studer, R., and Erdmann, M. (2008). The neon ontology engineering toolkit.

Hayashi, S., Bourdeau, J., and Mizoguchi, R. (2008). Toward establishing an ontological structure for the accumulation of learning/instructional design knowledge. In *Sixth International Workshop on Ontologies and Semantic Web for E-Learning, SWEL'08*, pages 1–10.

Hayashi, S., Bourdeau, J., and Mizoguchi, R. (2009). Structuring learning/instructional strategies through a state-based modeling. In *AIED*, pages 215–222.

Hayashi, Y., Bourdeau, J., and Mizoguchi, R. (2006). Ontological support for a theory-eclectic approach to instructional and learning design. In *EC-TEL*, pages 155–169.

Hayes, P., editor (2004). *RDF Semantics*. W3C Recommendation. World Wide Web Consortium.

Heim, M. (2000). *Virtual Realism*. Oxford University Press, Inc., New York, NY, USA.

Hernández, Y., Arroyo-Figueroa, G., and Sucar, L. E. (2008). Evaluating a probabilistic model for affective behavior in an intelligent tutoring system. In *ICALT*, pages 408–412.

Herzog, C. and Zierl, H. (1994). Fuzzy techniques for understanding students' solutions in an intelligent tutoring system. In *Proceedings of the World Conference on Educational Multimedia and Hypermedia*.

Hobbs, J. R. (1985). Granularity. In *IJCAI*, pages 432–435.

Hodge, G. (2000). Systems of knowledge organization for digital libraries: Beyond traditional authority files. The Digital Library Federation, Washington, D.C.

Hollnagel, E. (1993). The phenotype of erroneous actions. *International Journal of Man-Machine Studies*, 39:1–32.

Holmes, B. and Gardner, J. R. (2006). *E-Learning: Concepts and Practice*. Sage Publications Ltd.

Holt, P., Dubs, S., Jones, M., and Greer, J. (1994). *Student Modelling: The Key to Individualized Knowledge-Based Instruction, NATO ASI, Series F*, chapter The state of student modelling, pages 3–35. Lecture Notes in Computer Science. Springer-Verlag.

Horrocks, I. (2003). Implementation and optimization techniques. pages 306–346.

Hospers, M., Kroezen, E., Nijholt, A., Den Akker, R., and Heylen, D. (2003). An agent-based intelligent tutoring system for nurse education. In *In Applications of Intelligent Agents in Health*, pages 143–159. Birkhauser Publishing Ltd.

Huang, X., McCalla, G., Greer, J., and Neufeld, E. (1991a). Revising deductive knowledge and stereotypical knowledge in a student model. *User Modeling and User-Adapted Interaction*, 1(1):87–115.

Huang, X., McCalla, G. I., and Neufeld, E. (1991b). Using attention in belief revision. In *Proceedings of the Ninth National Conference on Artificial Intelligence*, pages 275–280.

Hustadt, U., Motik, B., and Sattler, U. (2004). Reducing shiq-description logic to disjunctive datalog programs. In *KR*, pages 152–162.

Ikeda, M., Kono, Y., and Misoguchi, R. (1993). Nonmonotonic model inference. In *Proceedings of the Thirteenth International Joint Conference on Artificial Intelligence, IJCAI'93*.

Ikeda, M. and Mizoguchi, R. (1992). Nonmonotonic model inference. technical report of artificial intelligence research groupgroup, isir osaka univ., ai-tr-92-10. Technical report.

Imbert, R., Sánchez, L., de Antonio, A., Méndez, G., and Ramírez, J. (2007). A multiagent extension for virtual reality based intelligent tutoring systems. volume 0, pages 82–84, Los Alamitos, CA, USA. IEEE Computer Society.

Jerinic, L. and Devedzic, V. (2000). The friendly intelligent tutoring environment. *SIGCHI Bull.*, 32(1):83–94.

Johnson, W. L. (2007). Serious use of a serious game for language learning. In *AIED*, pages 67–74.

Johnson, W. L. and Soloway, E. (1984). Proust: Knowledge-based program understanding. In *ICSE*, pages 369–380.

Johnson, W. L. and Valente, A. (2008). Tactical language and culture training systems: Using artificial intelligence to teach foreign languages and cultures. In *AAAI*, pages 1632–1639.

Jonassen, D. H. and Wilson, B. G. (1993). Constructivist uses of expert systems to support learning. *Journal of Computer-Based Instruction*, 20(3):86–94.

Jones, J., Millington, M., and Virvou, M. (2000). An assumption-based truth maintenance system in active aid for unix users. *Artif. Intell. Rev.*, 14(3):229–252.

Kalyanpur, A., Parsia, B., Sirin, E., Cuenca Grau, B., and Hendler, J. (2005). Swoop: A web ontology editing browser. *Journal of Web Semantics*, 4:2005.

Kass, R. (1989). Student modeling in intelligent tutoring systems – implications for user modeling. In Kobsa, A. and Wahlster, W., editors, *User Models in Dialog Systems*, pages 386–410. Springer Verlag, Symbolic Computation Series, Berlin Heidelberg New York Tokyo.

Katz, S. and Lesgold, A. (1992). Approaches to student modelling in the sherlock tutors. In *Proc. of the Third International Workshop on User Modeling*, pages 205–230, Dagstuhl, Germany.

Katz, S., Lesgold, A., Eggan, G., and Gordin, M. (1994). Modeling the student in sherlock i. In Greer, J., editor, *Proc. NATO Advanced Research*, pages 99–125, Berlin, Germany. Springer-Verlag.

Kifer, M., Lausen, G., and Wu, J. (1995). Logical foundations of object-oriented and frame-based languages. *Journal of the Association for Computing Machinery*, 42(4).

Klyne, G. and Carroll, J. J. (2004). Resource description framework (rdf): Concepts and abstract syntax. Technical report, W3C.

Kobsa, A. (2001). Generic user modeling systems. *User Model. User-Adapt. Interact.*, 11(1-2):49–63.

Koch, M. (2002). Interoperable community platforms and identity management in the university domain. *International Journal on Media Management*, 4(1):21–30.

Kono, Y., Ikeda, M., and Mizoguchi, R. (1994). Themis: a nonmonotonic inductive student modeling system. *J. Artif. Intell. Educ.*, 5(3):371–413.

Koper, R. and Olivier, B. (2004). Representing the learning design of units of learning. *Educational Technology and Society*, 7:97–111.

Krathwohl, D., Bloom, B. S., and Masia, B. B. (1973). *Taxonomy of Educational Objectives: The Classification of Educational Goals. Handbook II: The Affective Domain*. David McKay Company, Inc., New York, NY, USA.

Leidner, D. E. and Jarvenpaa, S. L. (1995). The use of information technology to enhance management school education: a theoretical view. *MIS Q.*, 19(3):265–291.

Lenat, D. B. and Guha, R. V. (1989). *Building Large Knowledge-Based Systems; Representation and Inference in the Cyc Project*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA.

Lewis, W. J., Rickel, J., and Lester, J. (2000). Animated pedagogical agents: Face-to-face interaction in interactive learning environments. *INTERNATIONAL JOURNAL OF ARTIFICIAL INTELLIGENCE IN EDUCATION*, 11:47–78.

London, B. and Clancey, W. J. (1982). Plan recognition strategies in student modeling: Prediction and description. In *AAAI*, pages 335–338.

Los Arcos, J., Muller, W., Fuente, O., Orúe, L., Arroyo, E., Leaznibarrutia, I., and Santander, J. (2000). *Intelligent Tutoring Systems*, chapter LAHYSTOTRAIN: Integration of Virtual Environments and ITS for Surgery Training, pages 43–52. Lecture Notes in Computer Science. Springer Berlin / Heidelberg.

Luke, S. and Heflin, J. (2000). Shoe 1.01. proposed specification, shoe project. university of maryland, 2000. Technical report. http://www.cs.umd.edu/projects/plus/SHOE/spec1.01.htm.

MacGregor, R. M. (1991). Inside the loom description classifier. *SIGART Bulletin*, 2(3):88–92.

Maedche, A. and Staab, S. (2003). Kaon - the karlsruhe ontology and semantic web meta project. *KI*, 17(3):27–.

Manola, F. and Miller, E., editors (2004). *RDF Primer*. W3C Recommendation. World Wide Web Consortium.

Mantovani, F., Galimberti, G. e., and Mantovani, F. (2003). Vr learning: Potential and challenges for the use of 3d environments in education and training.

Martin, B. (1999). Constraint-based student modeling: Representing student knowledge. In *New Zealand Computer Science Research Students' Conference*, pages 22–29.

Martin, J. and Vanlehn, K. (1995). Student assessment using bayesian nets. *International Journal of Human-Computer Studies*, 42:575–591.

Martins, J. P. (1991). The truth, the whole truth, and nothing but the truth: An indexed bibliography to the literature of truth maintenance systems. *AI Mag.*, 11(5):7–25.

McBride, B. Beckett, D. (2004). Rdf/xml syntax specification. W3C Recommendation.

Mccalla, G., Vassileva, J., Greer, J., and Bull, S. (2000). Active learner modelling. In *Intelligent Tutoring Systems*, pages 53–62. Springer-Verlag.

McCalla, G. I. and Greer, J. E. (1994). Granularity-based reasoning and belief revision in student models. *Student Modelling: The Key to Individualized Knowledge-Based Intruction*, 125:39–62.

McCalla, G. I., Greer, J. E., and the SCENT Research Team (1988). Intelligent advising in problem solving domains: The scent-3 architecture. In *Proceedings International Conference on Intelligent Tutoring Systems*, pages 124–131, Montreal.

Méndez, G. and de Antonio, A. (2005). Using intelligent agents to support collaborative virtual environments for training. *WSEAS Transactions on Computers, n. 10, vol. 4, pp. 1373-80. ISSN 1109-2750. October, 2005.*, 4(10):1373–1380.

Mili, F., Barr, J., Harris, M., and Pittiglio, L. (2008). Nursing training: 3d game with learning objectives. In *ACHI*, pages 236–242.

Millán, E., Pérez-de-la Cruz, J. L., and Suárez, E. (2000). Adaptive bayesian networks for multilevel student modelling. In *Intelligent Tutoring Systems*, pages 534–543.

Mislevy, R. J. and Gitomer, D. H. (1996). The role of probability-based inference in an intelligent tutoring system. *User Modeling and User-Adapted Interaction*, 5(3-4):253–282.

Mitrovic, A. (1998). Experiences in implementing constraint-based modeling in sql-tutor. In *ITS '98: Proceedings of the 4th International Conference on Intelligent Tutoring Systems*, pages 414–423, London, UK. Springer-Verlag.

Mitrovic, A. and Ohlsson, S. (1999). Evaluation of a constraint-based tutor for a database language. *International Journal of Artificial Intelligence in Education*, 10:238–256.

Mizoguchi, R. and Bourdeau, J. (2000). Using ontological engineering to overcome common aied problems. *Journal of Artificial Intelligence and Education*, 11:107–121.

Mizoguchi, R., Hayashi, Y., and Bourdeau, J. (2007). Inside theory-aware and standards-compliant authoring system. In *SWEL, Fifth International Workshop on Ontologies and Semantic Web for E-Learning, AIED 2007*, Marina del Rey, CA, USA.

Motik, B. and Studer, R. (2004). Kaon2 -a scalable reasoning tool for the semantic web. In *Procs. of the 2nd European Semantic Web Conference (ESWC 2005)*, pages 152–162.

Motta, E. (1999). *Reusable Components for Knowledge Modelling: Principles and Case Studies in Parametric Design*. IOSS Press.

Mozetic, I. (1991). Hierarchical model-based diagnosis. In *International Journal of Man-Machine Studies*, pages 64–75. Morgan Kaufmann.

Müller-Wittig, W., Voss, G., Bockholt, U., and Los Arcos, J. L. (2001). Enhanced training environment for minimally invasive surgery. In *WETICE*, pages 269–272.

Neches, R., Fikes, R., Finin, T., Gruber, T., Patil, R., Senator, T., and Swartout, W. (1991). Enabling Technology for Knowledge Sharing. *AI Magazine*, 12(3):36–56.

Niu, X. (2002). Purpose based learner modelling. In *Proceedings of the Graduate Symposium, CS Dept., University os Sakatchewan, Canada*.

Noguez, J. and Huesca, G. (2008). LaSiTo: A Lathe Simulated Virtual Laboratory. In *Procs. of the 38th IEEE International Conference on Frontiers in Education, FIE'08*.

Noguez, J. and Sucar, L. E. (2005a). A probabilistic relational student model for virtual laboratories. *Mexican International Conference on Computer Science*, 0:2–9.

Noguez, J. and Sucar, L. E. (2005b). A semi-open learning environment for virtual laboratories. In *MICAI*, pages 1185–1194.

Norman, D. A. (1987). Some observations on mental models. pages 241–244.

Noy, N. F., Sintek, S., Decker, S., Crubézy, M., Fergerson, R. W., and Musen, M. A. (2001). Creating semantic web contents with protégé-2000. *IEEE Intelligent Systems*, 16(2):60–71.

Ohlsson, S. (1986). Some principles of intelligent tutoring. *Instructional Science*, 14(3):293–326.

Ohlsson, S. (1994). Constraint-based student modelling. *Student Modelling: The Key to Individualized Knowledge-Based Instruction*, 125:167–190.

Ortony, A., L., G., and Collins, A. (1988). *The Cognitive Structure of Emotions*. Cambridge University Press.

Paiva, A. and Self, J. (1995). Tagus – a user and learner modeling workbench. *User Modeling and User-Adapted Interaction*, 4(3):197–226.

Petrushin, V. A. (1995). Intelligent tutoring systems: architecture and methods of implementation. a survey. *Journal of computer and system sciences international*, 33:117–139.

Pinto, H. S., Staab, S., and Tempich, C. (2004). Diligent: Towards a fine-grained methodology for distributed, loosely-controlled and evolving engineering of ontologies. In *In Proceedings of the 16th European Conference on Artificial Intelligence (ECAI 2004)*, pages 393–397. IOS Press.

Priest, T. and Young, R. M. (1988). Methods for evaluating micro-theory systems. In Self, J., editor, *Artificial Intelligence and Human Learning: Intelligent Computer-Aided Instruction*, pages 124–137. Chapman and Hall, London.

Querrec, R., Buche, C., Maffre, É., and Chevaillier, P. (2004). Multiagents systems for virtual environment for training. application to fire-fighting. *International Journal of Computers and Applications*, 1:25–34.

Quinlan, J. R. (1984). Learning efficient classification procedures and their application to chess end games. In Michalski, R. S., Carbonell, J. G., and Mitchell, T. M., editors, *Machine Learning: An Artificial Intelligence Approach*, pages 463–482. Springer, Berlin, Heidelberg.

Raggett, D., Le Hors, A., and Jacobs, I. (1999). Html 4.01 specification. W3C Recommendation.

Rake, S. T. and Smith, L. D. R. (1988). Use of an assumption-based truth maintenance system to record and resolve ambiguity in cardiac angiograms. In *Proceedings of the 4th International Conference on Pattern Recognition*, pages 656–666, London, UK. Springer-Verlag.

Reiter, R. (1987). A theory of diagnosis from first principles. *Artif. Intell.*, 32(1):57–95.

Reye, J. (1996). A belief net backbone for student modelling. In *Intelligent Tutoring Systems*, pages 596–604.

Reye, J. (1998). Two-phase updating of student models based on dynamic belief networks. In *Intelligent Tutoring Systems*, pages 274–283.

Rickel, J. and Lewis, W. J. (1998). Animated agents for procedural training in virtual reality: Perception, cognition, and motor control. *Applied Artificial Intelligence*, 13:343–382.

Riding, R. J. and Cheema, J. (1991). A framework for learner modelling. *Cognitive styles - An overview and integration*, 11:193–215.

Rodríguez-Artacho, M., Verdejo, M. F., Mayorga, J. I., and Calero, M. Y. (1999). Using a high-level language to describe and create web-based learning scenarios. In *29th ASEE/IEEE Frontiers in Education Conference*.

S., Y., Jürgen, A., and Steffen, S. (2002). Ontoedit: Guiding ontology development by methodology and inferencing. In *CoopIS/DOA/ODBASE*, pages 1205–1222.

Sánchez, J. A. (1995). Los objetivos educativos.

Schraw, G. and Dennison, R. S. (1994). Assessing metacognitive awareness. *Contemporary Educational Psychology*, 19:460–475.

Schreiber, G., Akkermans, H., Anjewierden, A., Dehoog, R., Shadbolt, N., Vandevelde, W., and Wielinga, B. (1999). *Knowledge Engineering and Management: The CommonKADS Methodology*. The MIT Press.

Self, J. (1993). Model-based cognitive diagnosis. *User Modeling and User-Adapted Interaction*, 3(1):89–106.

Self, J. A. (1992). Cognitive diagnosis for tutoring systems. In *ECAI*, pages 699–703.

Semmel, G. S., Davis, S. R., Leucht, K. W., Rowe, D. A., Kelly, A. O., and Bölöni, L. (2005). Launch commit criteria monitoring agent. In *AAMAS '05: Proceedings of the fourth international joint conference on Autonomous agents and multiagent systems*, pages 3–10, New York, NY, USA. ACM.

Shapiro, E. Y. (1983). *Algorithmic Program DeBugging*. MIT Press, Cambridge, MA, USA.

Simpson, E. J. (1972). The classification of educational objectives in the psychomotor domain. *The psychomotor domain*, 3:43–56.

Sintek, M. and Decker, S. (2002). Triple - a query, inference, and transformation language for the semantic web. In *International Semantic Web Conference*, pages 364–378.

Sirin, E., Parsia, B., Grau, B., Kalyanpur, A., and Katz, Y. (2007). Pellet: A practical owl-dl reasoner. *Web Semantics: Science, Services and Agents on the World Wide Web*, 5(2):51–53.

Skinner, B. F. (1950). Are theories of learning necessary? *Psychological Review*, 57:193–216.

Slater, M. (1999). Measuring presence: A response to the witmer and singer presence questionnaire.

Sleeman, D. and Brown, J. S. (1982). *Intelligent Tutoring Systems*. Academic Press, London.

Sleeman, D. H., Kelly, A. E., Martinak, R., Ward, R. D., and Moore, J. L. (1989). Studies of diagnosis and remediation with high school algebra students. *Cognitive Science*, 13(4):551–568.

Sleeman, D. H. and Smith, M. J. (1981). Modelling student's problem solving. *Artif. Intell.*, 16(2):171–188.

Smith, M. K., Welty, C., and McGuinness, D. L. Owl web ontology language guide. w3c recommendation.

Soldato, T. D. (1992). Detecting and reacting to the learner's motivational state. In *ITS '92: Proceedings of the Second International Conference on Intelligent Tutoring Systems*, pages 567–574, London, UK. Springer-Verlag.

Staab, S., Studer, R., Schnurr, H. P., and Sure, Y. (2001). Knowledge processes and ontologies. *IEEE Intelligent Systems*, 16(1):26–34.

Struss, P. and Dressler, O. (1989). "physical negation": integrating fault models into the general diagnostic engine. In *IJCAI'89: Proceedings of the 11th international joint conference on Artificial intelligence*, pages 1318–1323, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

Studer, R., Benjamins, V. R., and Fensel, D. (1998). Knowledge engineering: Principles and methods. *Data Knowl. Eng.*, 25(1-2):161–197.

Suárez-Figueroa, M. C. (2010). *Neon Methodology for Building Ontology Networks: Specification, Scheduling and Reuse*. PhD thesis, Universidad Politécnica de Madrid, Madrid, España.

Suárez-Figueroa, M. C., Blomqvist, E. D'Aquin, M., Espinoza, M., Gómez-Pérez, A., Lewen, H., Mozetic, I., Palma, R., Poveda, M., Sini, M., Villazón-Terrazas, B., Zablith, F., and Dzbor, M. (2009). Neon deliverable 5.4.2. revision and extension of the neon methodology for building contextualized ontology networks. neon project. Technical report. See http://www.neon-project.org.

Suárez-Figueroa, M. C., de Cea, A., Buil, C., Caracciolo, C., Dzbor, Gómez-Pérez, A., Herrero, G., Lewen, H., Montiel-Ponsoda, E., and Presutti, V. (2007). Neon deliverable 5.3.1. neon development process and ontology life cicle. neon project. Technical report. See http://www.neon-project.org.

Suárez-Figueroa, M. C., de Cea, A., Buil, C., Dellschaft, K., Fernández-López, M., García, A., Gómez-Pérez, A., Herrero, G., Montiel-Ponsoda, E., Sabou, M., Villazon-Terrazas, B., and Yufei, Z. (2008a). Neon deliverable 5.4.1. neon methodology for building contextualized ontology networks. neon project. Technical report. See http://www.neon-project.org.

Suárez-Figueroa, M. C., Fernández-López, M., Gómez-Pérez, A., Dellschaft, K., Lewen, H., and Dzbor, M. (2008b). Neon deliverable 5.3.2. revision and extension of the neon development process and ontology life cicle. neon project. Technical report. See http://www.neon-project.org.

Suarez Figueroa, M. C. and Gómez-Pérez, A. (2008). First attempt towards a standard glossary of ontology engineering terminology. In *8th International Conference on Terminology and Knowledge Engineering (TKE'08)*.

Suárez-Figueroa, M. C., Gómez-Pérez, A., Poveda, M., Ramos, J. A., Euzenat, J., and Le Duc, C. (2010). Neon deliverable 5.4.3. revision and extension of the neon methodology for building contextualized ontology networks. neon project. Technical report. See http://www.neon-project.org.

Sucar, L. E., Noguez, J., and Huesca, G. (2007). A semi-open learning environment for mobile robotics. *International journal of online engineering*, pages 1–6.

Swartout, B., Ramesh, P., Knight, K., and Russ, T. (1997). Toward distributed use of large-scale ontologies. *AAAI Symposium on Ontological Engineering*, pages 138–148.

Torres, E. G., Iida, T., and Watanabe, S. (1994). Measuring the student knowledge state in concept learning: An approximate student model. *IEICE transactions on information and systems*, 77(10):1170–1178.

Tsybenko, Y. (1993). Intelligent tutoring system for solving problems in geometry. In *Proceedings of ICCE'93, Taiwan*, pages 386–389.

Tsybenko, Y. (1995). Device models in student modeling. In Greer, J., editor, *Proceedings of International Conference on Artificial IntelliCence in Education, AIED'95*, pages 525–532. AACE.

Tyugu, E. (1984). *Conceptual Programming*. Nauka, Moscow, Russian.

Uschold, M. and Jasper, R. (1999). A framework for understanding and classifying ontology applications. In *Twelfth Workshop on Knowledge Acquisition Modeling and Management KAW 99*.

VanLehn, K. (1982). Bugs are not enough: Empirical studies of bugs, impasses, and repairs in procedural skills. *Journal of Mathematical Behaviour*, 3:3–72.

VanLehn, K. (1988). *Foundations of Intelligent Tutoring Systems*, chapter Student Modeling, pages 55–78. Hillsdale. N. J. Lawrence Erlbaum Associates.

Vassileva, J. (1992). A three-dimensional perspective on the current trends in student modelling. In Brusilovsky, P. and Stefanuk, V., editors, *Proc. of the East-West Conference on Emerging Computer Technologies in Education*, pages 315–320. Moscow, Russia.

Vassileva, J., Mccalla, G., and Greer, J. (2003). Multi-agent multi-user modeling in i-help. *User Modeling and User-Adapted Interaction*, 13(1-2):179–210.

Verbert, K. and Duval, E. (2004). Towards a global component architecture for learning objects: A comparative analysis of learning object content models. In *In Proceedings of the EDMEDIA 2004 World Conference on Educational Multimedia, Hypermedia and Telecommunications*, pages 202–208.

Weiten, M. (2009). *Semantic Knowledge Management*, chapter OntoSTUDIO as a Ontology Engineering Environment , pages 51–60. Springer-Verlag Berlin / Heidelberg.

Wenger, E. (1987). *Artificial intelligence and tutoring systems: computational and cognitive approaches to the communication of knowledge*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA.

White, B., Frederiksen, J., Forbus, K., Whalley, P., Everett, J., Ureel, L., Brokowski, M., Baher, J., and Kuehne, S. (1990). Causal model progressions as a foundation for intelligent learning environments.

Wielinga, B. J., Schreiber, A. T., and Breuker, J. A. (1992). Kads: a modelling approach to knowledge engineering. *Knowl. Acquis.*, 4(1):5–53.

Woolf, B. (1984). Context dependent planning in a machine tutor. Technical report, Amherst, MA, USA.

Xiaolin, N., McCalla, G., and Vassileva, J. (2004). Purpose-based expert finding in a portfolio management system. *Computational Intelligence*, 20(4):548–561.

Zhiping, L., Tianwei, X., and Yu, S. (2008). A web-based personalized intelligent tutoring system. In *CSSE '08: Proceedings of the 2008 International Conference on Computer Science and Software Engineering*, pages 446–449.