UNIVERSIDAD NACIONAL DE LA PLATA

Facultad de Informática

Magíster en Ingeniería de Software

Modelos de Especificación de Requerimientos para la Obtención de Esquemas Conceptuales en un Dominio Restringido: Comparación de Metodologías

Marcelo Martín Marciszack

Director: Mgter. Leandro Antonelli
Codirector: Dra. Roxana Giandini

Tesis presentada a la Facultad de Informática de la Universidad Nacional de La Plata como parte de los requisitos para la obtención del título de Magíster en Ingeniería de Software.

2010

ÍNDICE DE CONTENIDOS

ÍNDICE DE TABLAS

ÍNDICE DE PANTALLAS

ÍNDICE DE FIGURAS

CAPÍTULO 1

RESUMEN *Este capítulo tiene como finalidad servir como introducción, describiendo los motivos que llevaron a la selección del tema del presente trabajo de tesis, el objetivo del mismo y la descripción metodológica seguida, así también, se realiza una aproximación al estado del arte, describiendo la situación actual en referencia a la temática abordada y una descripción y ubicación de los contenidos desarrollados durante el presente trabajo.*

1.1. INTRODUCCIÓN

Muchos son los esfuerzos que se han realizado y se siguen realizando en la actualidad para solucionar los problemas de lo que se ha dado por llamar "la crisis del software", se evidencia en la gran cantidad de metodologías, métodos y herramientas dedicados a capturar los requerimientos con el fin de obtener un esquema conceptual.

Las debilidades de la mayoría de los métodos para la obtención de esquemas conceptuales se reflejan en las primeras etapas del desarrollo. El principal problema derivado de estas debilidades metodológicas radica en la dificultad en determinar si el modelo conceptual refleja fiel y completamente la esencia del dominio [Insfrán 2002].

Existen una gran variedad de trabajos que evidencian que los errores que se cometen en la etapa de especificación de requerimientos, para lo obtención de un esquema conceptual, tienen un costo relativo en relación a su reparación y crecerá en forma exponencial a medida que se avanza en etapas [Boehm 2001]. Un ejemplo que grafica el impacto que tiene una mala especificación de requerimientos y como se propaga en las sucesivas etapas es el propuesto como catarata de errores de Mizuno [Mizuno 1983].

Hay trabajos que describen la importancia de los requerimientos y al gerenciamiento de los mismos, como está descripta en [Antonelli 2002] basándose en la definición de [Ackoff 1974] "Fallamos más a menudo porque resolvemos el problema incorrecto, que porque obtenemos una solución deficiente al problema correcto". La preocupación por definir los requisitos de manera adecuada es extensamente tratada en [Sommerville 1997], donde el eje central es la definición de buenas prácticas en el establecimiento de los mismos, ya que plantea que "el éxito de cualquier proyecto de desarrollo está íntimamente relacionado con la calidad de los requisitos." y que "el proceso de los requisitos es mucho menos homogénea y bien entendido que el proceso de desarrollo de software en su conjunto".

Otros autores manifiestan la importancia de los requerimientos refiriéndose en un mismo sentido como "La Ingeniería de requerimientos se entiende como el proceso de descubrimiento y comunicación de las necesidades de clientes y usuarios y la gestión de los cambios de dichas necesidades" [Duran 2002]. La ingeniería de requerimientos del software es un proceso de búsqueda, refinamiento, modelado y especificación donde se toman como base requisitos de datos, flujo de información y control, y de comportamiento operativo. Goguen plantea que: "uno de los aspectos más importantes de la ingeniería de requerimientos es la comunicación, característica ésta que vuelve el proceso complejo por la alta presencia del factor humano que contiene y es la responsable de que la disciplina contenga aspectos sociales y culturales y no sólo de índole técnica" [Goguen 1994].

Los sistemas de software no existen en forma aislada: se utilizan en un contexto social y organizacional y los requerimientos del sistema de software se deben derivar y restringir de acuerdo a ese contexto. A menudo, satisfacer estos requerimientos es crítico para el éxito del sistema a construir. Una razón de porque muchos sistemas de software se entregan pero que nunca se usan es porque no se tiene en cuenta en forma adecuada todos los requerimientos del sistema incluidos los de su contexto. [Sommerville 2005]

En este sentido los esquemas conceptuales deben procurar establecer una definición sin ambigüedad de lo que se quiere representar. Para cumplir con este objetivo se dispone de un gran número de métodos, técnicas y herramientas que nos servirán de ayuda para lograrlo.

Por lo expuesto en párrafos anteriores, la propuesta de tesis "Modelos de Especificación de Requerimientos para la Obtención de Esquemas Conceptuales en un dominio restringido: Comparación de Metodologías" resulta por demás importante debido al gran impacto que un esquema conceptual tendrá en la correcta especificación de requerimientos.

1.2. PROPUESTA DE TRABAJO

1.2.1. Objetivo

El objetivo del presente trabajo, consiste en la realización de un estudio comparativo de metodologías y herramientas existentes, para la determinación de una técnica de especificación de requerimientos, específicamente la construcción de un esquema conceptual, realizado sobre diferentes dominios de aplicación bajo estudio.

El alcance definido para efectuar las comparaciones de las diferentes metodologías y herramientas será a nivel conceptual y se define desde el establecimiento de dominios objetos de estudio, con determinación de las características a evaluar tanto para las metodologías como para las herramientas de soporte de las mismas, la recolección de las valoraciones de las características evaluadas en los diferentes dominios y el establecimiento de un método

multicriterio para efectuar las comparaciones y de esta manera, determinar cuál combinación de Metodología/Herramienta reúne las mejores características para construir un Esquema Conceptual.

En resumen, el objetivo del presente trabajo, es el de proveer un análisis crítico sobre la aplicación de diferentes metodologías y herramientas para la especificación de requerimientos en la obtención de un esquema conceptual cuyo modelo resultante sea una representación fiel de la realidad que se está representando.

1.2.2. Estado del arte

Si bien hay documentación existente sobre trabajos relacionados con propuestas de especificaciones de requerimientos para la obtención de esquemas conceptuales, aplicando y comparando diversas metodologías, no se evidencia ningún trabajo que realice la comparación simultánea de todas las metodología/herramientas abordadas en el presente trabajo de tesis, y más aún sobre un Dominio de aplicación como lo son las Gramáticas Formales y las Máquinas Abstractas.

Existen trabajos donde se realizan apreciaciones de ventajas, de una Metodología/Herramienta en forma puntual sobre otra determinada, en donde las mismas resultan parciales y están condicionadas por la perspectiva desde donde parte el observador, siendo necesario determinar en forma exhaustiva cuales son los criterios a evaluar y que incluirán las metodologías/herramientas para ser utilizados en el proceso de comparación.

Muchas y variadas son las definiciones del contenido de una correcta especificación de requisitos para la obtención de Esquemas conceptuales. Kontoya [Kontoya 1996] propone que debe formar parte de la misma: el entorno, los usuarios del sistema, los servicios requeridos, las restricciones asociadas; al cuál se lo debe considerar un proceso de captura, análisis y resolución de las ideas, que se producen con diferentes niveles de detalle. Existen también varios estándares definidos, los cuales son utilizados como plantillas para la obtención de una correcta especificación de requerimientos [IEEE/ANSI 830 1984].

Otros trabajos relacionados abordan por estrategia centrarse en la interacción del usuario con el sistema como es el caso de metodologías RUP/UML y una gran cantidad de variantes de metodologías similares. Otros trabajos se centran en un enfoque orientado al cliente con un metamodelo que contiene descripción sobre el Contexto o Universo de Discurso como es el caso del Léxico Extendido del Lenguaje [Leite 1997].

Las ontologías proveen una comprensión compartida y concensuada del conocimiento de un dominio que puede ser comunicada entre personas y sistemas heterogéneos [Muñoz 2002]. Estas ideas fueron desarrolladas en el campo disciplinar de la Inteligencia Artificial (IA) para facilitar el intercambio y reuso del conocimiento. Por lo tanto para un dominio

determinado, una ontología definirá un vocabulario común para los investigadores que necesiten compartir información. Una ontología además debe contener definiciones de los conceptos básicos y sus relaciones las que deben poder ser interpretadas por una máquina.

Así definida una ontología resultará sumamente útil, para la representación de un Esquema Conceptual, y es por tal motivo que será incluida en el presente trabajo de comparación de metodologías.

1.2.3. Descripción metodológica

Para la realización del presente trabajo se establece como dominios a modelar las Gramáticas Formales y Máquinas Abstractas. La elección de estos dominios tiene un doble propósito: por un lado nos permitirá comparar entre si las diferentes metodologías en donde se analizarán, ventajas y desventajas sobre las dimensiones de análisis que se determinarán previo a la realización del ensayo y por otro lado, que dentro de cada uno de los Modelos adoptados en la comparación, se establecerá el grado de correspondencia entre la conceptualización de las Máquinas y Gramáticas, ya que al existir un isomorfismo entre ambos dominios, éste debería continuar en los esquemas conceptuales resultantes de la aplicación de cada una de las metodologías comparadas.

Las metodologías y las herramientas que se compararán en el desarrollo de este trabajo son las siguientes: Léxico Extendido del Lenguaje (LEL), Escenarios y Tarjetas CRC, utilizando como herramienta de descripción al Baseline Mentor Workbench (BMW); desarrollo de ontologías utilizando protege-2000 como herramienta de modelado y edición de Ontologías; casos de uso obtenidos a partir de la metodología Rational Unified Process (RUP) con la utilización de Rational Rose como herramienta de soporte.

Las dimensiones de análisis a definir para efectuar las comparaciones de las metodologías/herramientas, deberán contemplar no sólo aspectos relacionados con los referidos a la representación del conocimiento “Esquema Conceptual”, sino también con todo lo relacionado con el poder expresivo y comunicacional con los expertos en un determinado dominio. Por tal motivo se determinará a través de investigación bibliográfica sobre trabajos previamente realizados cuáles son las cualidades que deben poseer una metodología, y las herramientas de soporte a la misma, para la obtención de un Esquema Conceptual.

1.2.4. Evaluación de la propuesta

Para evaluar la propuesta de trabajo, en cuanto a la valoración de las conclusiones a las que se arribará con la aplicación metodológica en el desarrollo del trabajo, se efectuará a través de la aplicación de la combinación de valoraciones históricas y observacionales, junto con la utilización y aplicación del proceso de análisis jerárquico AHP con la utilización de la herramienta Expert Choice Ver. 11.5.

Esta combinación de valoraciones se manifiesta en varios tramos de la aplicación de la propuesta. En primer lugar lo que se intentará es establecer cuales son las dimensiones de análisis para poder establecer un marco con el cuál efectuar la comparación las diferentes metodologías/herramientas. Estas dimensiones y características deseables que deberán estar contenidas tanto en las metodologías como en las herramientas, para la especificación de un Esquema Conceptual, las cuales, se determinarán a través de una investigación bibliográfica para su definición.

Las dimensiones de análisis detectadas serán validadas en forma previa a la aplicación en el dominio seleccionado, sobre un simple ejemplo de aplicación de manera de independizar el proceso de comparación de las particularidades inherentes al dominio de aplicación.

A su vez, las conclusiones arribadas en el proceso de comparación sobre el simple ejemplo, serán confrontadas con las obtenidas sobre el dominio objeto de estudio como son las Gramáticas Formales y las Máquinas Abstractas el cuál además tiene la particularidad de ser isomorfos, por lo tanto se evaluará además si los modelos resultantes obtenidos a través de las distintas metodologías/herramientas continúan siendo isomorfos entre ellos.

1.3. PUBLICACIONES Y TRABAJOS RELACIONADOS.

Los siguientes artículos publicados y trabajos de investigación son algunos de los resultados obtenidos respecto al tema de esta tesis:

- *Proyecto de Investigación y desarrollo "Modelos de Especificación de Requerimientos para la obtención de esquemas conceptuales en un dominio restringido: Comparación de Metodologías".*
  Proyecto de la Secretaría de Ciencia y Tecnología – Promocional – Código EIPRCO767 Período desde 01 de Enero 2008 al 31 de Diciembre de 2009.
  Director: Leandro Antonelli – Codirector: Marcelo Marciszack
  Disposición SCYT Nº 132 / 08

- *Publicación del trabajo "Construcción de esquemas conceptuales para la elicitación de requerimientos con ontologías utilizando Protégé"* - aceptado en el Encuentro de Investigadores y Docentes de Ingeniería V ENIDI
  Los Reyunos – Mendoza – Argentina 11 al 13 de Noviembre de 2009
  Autores: Marciszack, Marcelo – Perez Cota, Manuel – Antonelli, Leandro – Giandini, Roxana – Cárdenas, Marina
  Obra en CD – ISBN: 978-950-42-0121-2
- *Publicación del trabajo "Construcción de una Ontología utilizando Protégé para la elicitación de Requerimientos"* en el Congreso Nacional Información y Comunicación para la Sociedad del Conocimiento Organizado por el Centro de Ingenieros de Córdoba.

Córdoba – Argentina 16, 17 y 18 de junio de 2009
Autores: Marciszack, Marcelo – Perez Cota, Manuel – Antonelli, Leandro – Cárdenas, Marina
Pág. 55 ISBN: 978-987-24343-2-8 Obra impresa y CD
Centro de Ingenieros Córdoba - Jorge Sarmiento Editor
http://www.cnit2009.org.ar/

- *Publicación del Artículo "Construcción de una Ontología para gramáticas formales y máquinas abstractas utilizando protege para la elicitación de requerimientos"* – aceptado en el Área de Ingeniería de Software y bases de datos, en el XI Workshop de Investigadores en Ciencias de la Computación WICC 2009.
  San Juan, Argentina 7, 8 de Mayo de 2009.
  Autores: Marciszack, Marcelo – Perez Cota, Manuel – Antonelli, Leandro – Giandini, Roxana – Cárdenas, Marina
  Obra en CD – ISBN: 978-950-605-570-7

- *Publicación del Artículo "Construcción de un modelo conceptual para gramáticas formales y máquinas abstractas con ontologías utilizando Protégé"* aceptado en el X workshop de Investigadores en Ciencias de la Computación. WICC 2008. Organizado por la Facultad de Ingeniería de la Universidad Nacional de la Pampa y la Red UNCI.
  General Pico, La Pampa, 05 y 06 de mayo de 2008.
  Autores: Marciszack, Marcelo – Cárdenas, Marina - Vázquez, Juan Carlos J. – Castillo, Julio Javier
  Obra en CD ISBN 978-950-863-101-5

- *Publicación del trabajo "Construcción de un Modelo Conceptual para Gramáticas Formales y Máquinas Abstractas con UML usando Racional"* en el Congreso Nacional de Estudiantes de Ingeniería en Sistemas de Información - CNEISI 2008 Universidad Tecnológica Nacional – Facultad Regional La Plata
  Autor: Torres Díaz, Melisa Daniela, bajo la Dirección de Marciszack, Marcelo en el marco del Proyecto Promocional *"Modelos de Especificación de Requerimientos para la obtención de esquemas conceptuales en un dominio restringido: Comparación de Metodologías"*.

- *Publicación del trabajo "Modelos de especificación de requerimientos para la obtención de esquemas conceptuales en un dominio restringido"*
  Congreso Nacional de Estudiantes de Ingeniería en Sistemas de Información – CNEISI 2007

Universidad Tecnológica Nacional – Facultad Regional Córdoba
Autor: Fernández Taurant, Juan Pablo, Bajo la Dirección de Marciszack, Marcelo Martín

1.4. ORGANIZACIÓN DE LA TESIS

Para facilitar la comprensión general del trabajo, el mismo se ha dividido en ocho capítulos y siete anexos. A continuación se desarrolla una breve descripción de la temática desarrollada en cada uno de los mismos.

CAPÍTULO 2: Este capítulo describe la importancia que tienen los esquemas conceptuales, con el fin de abstraer la esencia de un dominio a modelar, sirviendo a la vez para una correcta y completa especificación de los requerimientos que debe cumplir. Se describe el objetivo de los esquemas conceptuales, y se definen formalmente a los requerimientos y el impacto que tiene una incorrecta definición de los mismos en futuras etapas del proceso de desarrollo de software en la construcción de los sistemas de información. Se presentan las características esenciales que debe poseer un modelo, construyendo un instrumento que nos permita evidenciar, evaluar y comparar las distintas metodologías/herramientas en la construcción de un Esquema Conceptual.

CAPÍTULO 3: En este capítulo se presenta una metodología orientada al cliente que se fundamenta en base a una documentación integrada en una estructura denominada Client Oriented Requeriments Baseline, la cual se basa en el Léxico Extendido del Lenguaje (LEL) para el modelado del vocabulario, y mediante el uso de Escenarios para representar su comportamiento. LEL y Escenarios serán utilizados para capturar la esencia del dominio, utilizando el Baseline Mentor Workbench (BMW) como herramienta de soporte.

CAPÍTULO 4: En este apartado se presentan las Ontologías, partiendo desde su descripción, comportamiento, metodologías, lenguajes de representación y herramientas disponibles. Se procede a la selección del conjunto de Metodología, Lenguaje y Herramienta, los que serán utilizados para la comparación de metodologías para el desarrollo de la Ontología de manera de construir un Esquema conceptual.

CAPÍTULO 5: Se parte de una revisión histórica de la evolución de la metodología seleccionada RUP/UML (Rational Unified Process), se describen sus características principales y estructura del proceso describiendo a sus componentes. Se presenta y se selecciona como herramienta de modelado a Rational, ya que es la que incluye la descripción del contexto a través de la definición de las reglas del negocio, describiendo su funcionamiento.

CAPÍTULO 6: En este capítulo se presentan los aspectos metodológicos seguidos en el proceso de comparación de las diferentes metodologías/herramientas bajo estudio en el presenta trabajo. Se enuncian y analizan las particularidades de los dominios seleccionados. Se describe la metodología seleccionada para establecer la comparación de las diferentes metodologías/herramientas es el proceso de análisis Jerárquico (AHP). Se visualizan los aspectos de los criterios de evaluación seleccionados y la operatoria seguida para la asignación de los mismos.

CAPÍTULO 7: En este capítulo se efectúa la valoración de acuerdo a los criterios metodológicos propuestos en el capítulo anterior sobre ambos dominios objetos bajo estudio. Se construyen los esquemas conceptuales para ambos dominios a través de la aplicación de las diferentes metodologías/herramientas los cuales quedan plasmados en diferentes anexos. Se presentan los resultados de la aplicación del método de comparación multicriterio AHP sobre las diferentes metodologías/herramientas de acuerdo a los criterios y dimensiones de análisis descriptas para la evaluación de las mismas en ambos dominios.

CAPÍTULO 8: En este capítulo se evalúan las diferentes metodologías/herramientas objetos de comparación. Se procede a establecer las conclusiones en referencia a: a) Comparación para cada una de las metodologías/herramientas de los resultados obtenidos en los diferentes dominios Sistema Docentes y Máquinas Abstractas / Gramáticas Formales. b) Comparación entre las metodologías/herramientas de acuerdo a la aplicación del método AHP seleccionado para la comparación multicriterio. c) Mantenimiento del isomorfismo en los modelos obtenidos. A partir de estas tres dimensiones de análisis se establece la conclusión general del trabajo y los futuros trabajos de Investigación. Se elabora una conclusión final sobre las metodologías/herramientas objeto de estudio del presente trabajo de tesis

ANEXOS

ANEXO I: Planilla de valoración de criterios seleccionados

Descripción del instrumento utilizado en el proceso de comparación de las Herramientas / Metodologías.

ANEXO II: LEL/Escenarios/CRC – BMW: Aplicación al Modelo de Docentes

Aplicación completa del proceso metodológico y uso de la herramienta, sobre un simple dominio de aplicación

ANEXO III: Ontologías – Protégé - Aplicación al Modelo de Docentes

Aplicación completa del proceso metodológico y uso de la herramienta, sobre un simple dominio de aplicación.

ANEXO IV: RUP/UML – Rational - Aplicación al Modelo de Docentes

Aplicación completa del proceso metodológico y uso de la herramienta, sobre un simple dominio de aplicación

ANEXO V: LEL/Escenarios/CRC – BMW: Aplicación a Gramáticas Formales y Máquinas Abstractas

Aplicación completa del proceso metodológico y uso de la herramienta, sobre el dominio bajo estudio.

ANEXO VI: Ontologías – Protégé: Aplicación a Gramáticas Formales y Máquinas Abstractas

Aplicación completa del proceso metodológico y uso de la herramienta, sobre el dominio bajo estudio.

ANEXO VII: RUP/UML – Rational: Aplicación a Gramáticas Formales y Máquinas Abstractas

Aplicación completa del proceso metodológico y uso de la herramienta, sobre el dominio bajo estudio.

CAPÍTULO 2

RESUMEN *Este capítulo comienza con el abordaje de la importancia que tienen los Esquemas Conceptuales, con el fin de abstraer la esencia de un dominio a modelar, sirviendo a la vez para una correcta y completa especificación de los requerimientos que debe cumplir. Se describe el objetivo de los Esquemas Conceptuales y se definen formalmente a los requerimientos y el impacto que tiene una incorrecta definición de los mismos en futuras etapas del proceso de desarrollo de software en la construcción de los sistemas de información.*
Se presentan las características esenciales que debe poseer un modelo, construyendo un instrumento de evaluación que nos permita evidenciar, evaluar y comparar las distintas metodologías/herramientas en la construcción de un Esquema Conceptual.

2.1. ESQUEMAS CONCEPTUALES

Es posible encontrar, como describiremos a continuación, un conjunto de definiciones o aproximaciones a ellas, de lo que representa un esquema conceptual, con diversos significados, ya sea por el enfoque u óptica disciplinar desde donde se lo aborde, o incluso dentro de una misma área disciplinar.

2.1.1. Definición

Comenzamos con la definición más general y abarcativa desde una concepción filosófica en donde, un esquema conceptual, surge básicamente por dos razones:

- La primera es la facultad intrínseca de percibir representación (Modelos de la Realidad)
- La segunda es la facultad de conocer un objeto a través de tales representaciones.

Lo primero a resaltar, es que desde esta óptica, la capacidad de crear esquemas conceptuales es una característica universal, pero cuya aplicación puede verse dificultada por la forma en que los mismos son transmitidos y percibidos por las diferentes personas.

Desde la óptica disciplinar de los Sistemas de Información y los sistemas de software asociados, un *Esquema Conceptual* será definido como un modelo de representación de la realidad, sobre un dominio de problema determinado, el cual deberá incluir además, el lenguaje utilizado en su definición, de manera que no existan ambigüedades, de manera de reducir el "gap" semántico, entre el constructor del modelo y los usuarios del mismo.

En este contexto el presente trabajo, se focaliza con la visión aportada por [Insfrán 2002b] en donde un Esquema Conceptual es interpretado como un refinamiento de los requerimientos de usuario a través de los requisitos funcionales que resultarán en especificaciones más detalladas que constituirán dicho esquema.

2.1.2. Propuestas de los esquemas conceptuales

Durante años para realizar las actividades de modelado de los sistemas de información se ha atacado el problema de establecer un mismo significado para los términos utilizados dentro de una empresa, o sea el de establecer un esquema conceptual único para todos los actores y todas las actividades dentro de una organización, siendo éste el propósito del modelado del esquema conceptual [Sesé 2006].

En las últimas tres décadas hemos asistido a una gran proliferación de notaciones, lenguajes, técnicas, metodologías y herramientas para realizar esta tarea. Lamentablemente a pesar de los esfuerzos de los distintos organismos de estandarización, son variadas las definiciones que podemos encontrar en ANSI: *American Nacional Standards Institute*, ISO: *International Organization for Standardization,* NIST: US *Nacional Institute of Standard and Technology*, IFIP: *International Federation for Information Processing*, IEEE: *Institute of Electrical and Electronics Engineers. Software Engineering Standard committee of the Computer Society*

Ejemplificaremos ahora cómo es que no existen consensos, ni consistencia terminológica, ni siquiera en el nombre de lo que produce como resultado del esquema conceptual [Sesé 2006].

El nombre más utilizado es el nombre de Esquema Conceptual el cuál fue propuesto por [ANSI 1975] pero también es referido como Modelo de Empresa (business Model), y Modelo de datos (Data Model) [Verrijn-Stuart 2001], o como Modelo Conceptual [Shanks 2003], también como Modelo de Datos Semántico [Wand 1998], u Ontología [Guarino 1998], o Diagrama de Clases [Booch 1998].

Tal como se plantea en [Sesé 2006], lo anterior, no deja de ser una situación al menos irónica, ya que, si tenemos en cuenta, que estos diferentes autores de grupos de investigación persiguen el desarrollo de métodos, que permitan la definición consistente y compartida, del significado de términos.

En [IEEE 1999]: "*The conceptual schema provides a single integrated definition of the concepts relevant to an enterprise, unbiased toward any particular application. The primary objective of this conceptual schema is to provide a consistent definition of the meanings and interrelationship of concepts. This definition can then by used to integrate, share, and manage the integrity of the concepts"*.

"El esquema conceptual ofrece una única e integrada definición de los conceptos relevantes para una empresa, sin estar condicionada a una aplicación en particular. El principal objetivo de este esquema conceptual es proporcionar una definición coherente de significados y la interrelación de conceptos. Esta definición puede servir para integrar, compartir y gestionar la integridad de los conceptos."

O como consta en el informe de la ISO [Griethuysen 1982] en donde se define al esquema conceptual como*: "A consistent collection of sentences expressing the necessary propositions that hold for a universe of discourse".*

"Una coherente colección de frases que expresan las propuestas necesarias para que mantenga un universo de discurso."

En dónde se explica como universo de discurso: "*The collection of all objects (entities) that ever have been, are, or ever will be in a selected portion of real world or postulated world of interest that is being described*".

"La recopilación de todos los objetos (entidades) que nunca han sido, son o nunca serán seleccionados en una porción del mundo real o mundo postulado de interés que está siendo descrito."

De las consideraciones sobre esquemas conceptuales anteriormente propuestas por IEEE e ISO, podemos inferir que muchos autores interpretan que entre el mundo real y el esquema conceptual existe una correspondencia directa e independiente de la percepción de la personas. Esta postura es calificada como realismo ingenuo "naive realism" por [Hirschheim 1995]. Otros autores prefieren reflejar esta problemática flexibilizando la definición considerando que: la representación de la realidad que supone un esquema conceptual, no refleja a la realidad en sí misma, si no que a la percepción que un grupo de personas tiene sobre la misma [Shanks 2003].

2.1.3. Validación de los esquemas conceptuales

Un Esquema Conceptual constituye el núcleo central en lo que refiere a la formalización de los requerimientos a cumplir, ya sea cuando en forma particular hablamos de la construcción de software asociado a un sistema de información, o bien cuando generalizamos la definición haciéndola extensiva en el modelado de un determinado dominio de aplicación, por este motivo parece lógico asegurarse de que es correcto antes de comenzar a construir los artefactos de software.

Sin embargo, la actividad de validación de los esquemas conceptuales ha sido poco investigada y bastante descuidada en la mayor parte de las metodologías propuestas.

Analizaremos una de las metodologías de mayor prestigio de la actualidad, la metodología RUP/UML, de los autores Booch, Jacobson y Rumbaugh. De la lectura de sus libros se puede inferir que los autores presuponen que el significado de los términos es objetivo y tienen por si mismo la suficiente precisión para ser capturado por los analistas, lo cual esta postura inicial los lleva a prestar muy poca atención a la validación de los requerimientos por parte de los usuarios. Por ejemplo en Booch y Rumbaugh, ni siquiera mencionan el problema de validación de los requerimientos, y sólo Jacobson le dedica alguna

atención mediante la utilización de su técnica de casos de uso. Esta técnica posteriormente se incorporó al Lenguaje UML [Booch 1998], sin embargo esta técnica no ataca en forma precisa ni explícita el problema de la validación de los requerimientos de información, esto es, de alguna manera la verificación de la coherencia de los significados atribuidos a los distintos términos del esquema conceptual por los diferentes usuarios. Estos autores proponen que el modelo conceptual, o como lo denominan modelo de dominio "*domain model*" sea la vista del modelo que se ocupe de validar los requerimientos.

La validación de los requerimientos de información es crítica en el desarrollo de los sistemas de información y por ende en el sistema de software asociados a estos. Sin embargo es un problema poco estudiado [Sesé 2006] en donde se afirma que la validación de los esquemas conceptuales no está madura, y se requiere mayor investigación en dicha área.

El problema de la validación de requerimientos es tratada en [Loucopoulos 1995] en donde básicamente proponen que existen dos maneras que son: por un lado la revisión del mismo por parte del usuario y por otro lado, la descripción de las transacciones.

La primer manera consiste en que los usuarios comprendan el significado del esquema e identifiquen las discrepancias entre este y la realidad tal como la perciben. Sin embargo, este procedimiento presenta algunos inconvenientes, que es que el usuario debe familiarizarse con la notación que se está utilizando para especificar el esquema conceptual, lo cuál aunque se utilicen notaciones gráficas, no siempre es fácil conseguir que el usuario entienda, con el nivel adecuado de precisión, el alcance de cada uno de los símbolos utilizados. Otro inconveniente adicional y más grave es que los usuarios no son capaces de percibir su universo de discurso (dominio del Modelo) al mismo nivel de abstracción en el que se presentan en los esquemas conceptuales. Otra característica que resulta importante es que si la validación del modelo conceptual puede ser realizada en forma completa, o lo que es lo mismo, si el modelo es capaz de reflejar todo el conjunto de datos, acciones y funciones que se realicen sobre el mismo.

La segunda manera de validar el esquema conceptual es la realización de un "*test de transacciones*", esta técnica consiste en la comprobación de que el esquema conceptual soporta todas las transacciones que el usuario hará sobre él.

Parece ser obvio que antes de comenzar con la construcción del software, se compruebe de manera acabada y exhaustiva que el modelo conceptual es capaz de soportar toda y cada una de las transacciones que se efectuarán sobre él, pero es necesario hacerlo dado que las debilidades de algunas metodologías radican precisamente en algunas de las siguientes limitaciones: la primera, es que las transacciones se presentan el forma descriptivas. La segunda, que se de posiblemente por causa de la primera, no proponen ninguna manera de realizar la tarea. Y la tercera, que asume que la validación la asume el analista.

2.2. REQUERIMIENTOS E INGENIERÍA DE REQUERIMIENTOS.

2.2.1. Ingeniería de requerimientos

En la actualidad, son muchos los procesos de desarrollo de software que existen. Con el pasar de los años, la Ingeniería de Software ha introducido y popularizado una serie de estándares para medir y certificar la calidad, tanto del sistema a desarrollar, como del proceso de desarrollo en sí. Se han publicado muchos libros y artículos relacionados con este tema, con el modelado de procesos del negocio y la reingeniería. Un número creciente de herramientas automatizadas han surgido para ayudar a definir y aplicar un proceso de desarrollo de software efectivo. Hoy en día la economía global depende más de sistemas automatizados que en épocas pasadas; esto ha llevado a los equipos de desarrollo a enfrentarse con una nueva década de procesos y estándares de calidad.

La Ingeniería de Requerimientos cumple un papel primordial en el proceso de producción de software, ya que enfoca un área fundamental: la definición de lo que se desea producir. Su principal tarea consiste en la generación de especificaciones correctas que describan con claridad, sin ambigüedades, en forma consistente y compacta, el comportamiento del sistema; de esta manera, se pretende minimizar los problemas relacionados al desarrollo de sistemas.

A continuación se darán algunas definiciones para ingeniería de requerimientos.

"Ingeniería de Requerimientos es la disciplina para desarrollar una especificación completa, consistente y no ambigua, la cual servirá como base para acuerdos comunes entre todas las partes involucradas y en dónde se describen las funciones que realizará el sistema" [Boehm 2001].

"Ingeniería de Requerimientos es el proceso por el cual se transforman los requerimientos declarados por los clientes, ya sean hablados o escritos, a especificaciones precisas, no ambiguas, consistentes y completas del comportamiento del sistema, incluyendo funciones, interfaces, rendimiento y limitaciones". [ANSI/IEEE 1984].

"Es el proceso mediante el cual se intercambian diferentes puntos de vista para recopilar y modelar lo que el sistema va a realizar. Este proceso utiliza una combinación de métodos, herramientas y actores, cuyo producto es un modelo del cual se genera un documento de requerimientos" [Leite 1997].

"Ingeniería de requerimientos es un enfoque sistémico para recolectar, organizar y documentar los requerimientos del sistema; es también el proceso que establece y mantiene acuerdos sobre los cambios de requerimientos, entre los clientes y el equipo del proyecto" [Pressman 1993].

Estudios realizados muestran que un gran porcentaje de los proyectos de software fracasan por no realizar un estudio previo de requisitos. Otros factores como falta de participación del usuario, requerimientos incompletos y el cambio a los requerimientos, también ocupan sitiales altos en los motivos de fracasos.

2.2.2. Requerimientos

Es habitual en nuestra área disciplinar, encontrar diferentes significados para una misma palabra o concepto. De las muchas definiciones que existen para requerimiento, a continuación se presenta la definición que aparece en el glosario de la IEEE STD-610 [ANSI/IEEE 1990]:

1. Una condición o necesidad de un usuario para resolver un problema o alcanzar un objetivo.
2. Una condición o capacidad que debe estar presente en un sistema o componentes de sistema para satisfacer un contrato, estándar, especificación u otro documento formal.
3. Una representación documentada de una condición o capacidad como en 1) o 2).

Los requerimientos de un sistema, son definidos en las primeras etapas de desarrollo del mismo, y nos describen la manera en que el sistema deberá comportarse en forma conjunta con las propiedades o atributos del mismo. Ellos pueden tomar la forma de restricciones sobre los procesos de desarrollo del sistema [Sommerville 1999]. Por lo tanto, los requisitos se podrían describir de la siguiente manera:

- Facilidad a nivel de usuario
- Definición del Sistema a nivel general
- Restricciones específicas en el sistema
- Restricciones sobre el desarrollo del sistema

Algunos autores sugieren que los requerimientos deben mostrar solo lo que debe hacer el sistema, y como lo hace. Esta es una atractiva idea pero que no es simple de implementar en la práctica.

En este sentido es común encontrar una subclasificación de los requerimientos, entre requerimientos funcionales y no funcionales.

2.2.2.1. Requerimientos funcionales

Los requerimientos funcionales definen las funciones que el sistema será capaz de realizar. Describen las transformaciones que el sistema realiza sobre las entradas para producir salidas.

2.2.2.2. Requerimientos no funcionales

Los requerimientos no funcionales tienen que ver con características que de una u otra forma puedan limitar el sistema, como por ejemplo, el rendimiento (en tiempo y espacio), interfaces de usuario, fiabilidad (robustez del sistema, disponibilidad de equipo), mantenimiento, seguridad, portabilidad, estándares, etc.

2.2.3. Características de los requerimientos

Las características de un requerimiento son sus propiedades principales. Un conjunto de requerimientos en estado de madurez, deben presentar una serie de características tanto individualmente como en grupo.

A continuación se presentan las características relevadas de diferentes autores que resultan más importantes, tomando como referencia lo publicado por [Leue 2000] y [Wiegers 1999b]:

Necesario: Un requerimiento es necesario si su omisión provoca una deficiencia en el sistema a construir, y además su capacidad, características físicas o factor de calidad no pueden ser reemplazados por otras capacidades del producto o del proceso.

Correcto: Si y solo si para cada uno de los requerimientos detectados son los que el conjunto de funcionalidades del sistema software debe poseer.

Conciso: Un requerimiento es conciso si es fácil de leer y entender. Su redacción debe ser simple y clara para aquellos que vayan a consultarlo en un futuro.

Completo: Un requerimiento está completo si no necesita ampliar detalles en su redacción, es decir, si se proporciona la información suficiente para su comprensión. Por otra parte una SRS está completa si se encuentran identificados todos los requerimientos individuales.

Consistente: Un requerimiento es consistente si no es contradictorio con otro requerimiento. Esta regla de consistencia debe darse entre todos los requerimientos de la SRS

No ambiguo: Un requerimiento no es ambiguo cuando tiene una sola interpretación. El lenguaje usado en su definición, no debe causar confusiones al lector.

Verificable: Un requerimiento es verificable cuando puede ser cuantificado de manera que permita hacer uso de los siguientes métodos de verificación: inspección, análisis, demostración o pruebas.

Trazable: Si el origen de cada requerimiento es claro, conciso y fácilmente referenciable

Modificable: Si y solo si su estructura y estilo son tales que cualquier cambio necesario de efectuar puede ser realizado en forma fácil, completa y consistente.

2.2.4. Necesidad de una buena definición de los requerimientos

En el desarrollo de sistemas de software, asociados a los sistemas de información han tenido una constante en común desde la década del 60, que han tenido muchos errores en su concepción. Han sido entregados en forma tardía, no han hecho lo que el usuario realmente quería, y han tenido funcionalidades que nunca han sido utilizadas por los usuarios [Sommerville 1999].

La importancia de los requerimientos y a la gestión de los mismos está descripta en [Antonelli 2002] basándose en la definición de [Ackoff 1974] *"Fallamos más a menudo porque resolvemos el problema incorrecto, que porque obtenemos una solución deficiente al problema correcto"*.

Pocas veces existe una sola razón o una única solución al problema de una correcta especificación de requerimientos, pero la mejor contribución para minimizar estos inconvenientes está en una correcta definición los mismos.

Una especificación de requerimientos define que servicios el sistema debe proporcionar y cuáles son las limitaciones en operación. Los problemas más comunes que surgen con la especificación de requerimientos son:

- Los requerimientos no reflejan las reales necesidades del usuario para el sistema.
- Los requerimientos son incompletos y/o inconsistentes.
- Antes de hacer cambios los requerimientos deben ser aceptados.
- Hay malentendidos entre los clientes, quienes desarrollan los requisitos del sistema, los que construyen el sistema y los que hacen el mantenimiento del sistema.

2.2.5. Dificultades para definir los requerimientos

A continuación se enuncian una serie de puntos que deben ser tenidos en cuenta para obtener una correcta especificación de requerimientos.

- Los requerimientos no son obvios y vienen de muchas fuentes.
- Son difíciles de expresar en palabras (el lenguaje natural es ambiguo).
- Existen muchos tipos de requerimientos y diferentes niveles de detalle.
- La cantidad de requerimientos en un proyecto puede ser difícil de manejar.

- Nunca son iguales. Algunos son más difíciles, más riesgosos, más importantes o más estables que otros.
- Los requerimientos están relacionados unos con otros, y a su vez se relacionan con otras partes del proceso.
- Cada requerimiento tiene propiedades únicas y abarcan áreas funcionales específicas.
- Un requerimiento puede cambiar a lo largo del ciclo de desarrollo.
- Son difíciles de cuantificar, ya que cada conjunto de requerimientos es particular para cada proyecto.

2.2.6. Rol e importancia de los requerimientos

Los principales beneficios que se obtienen de una correcta especificación de requerimientos son:

- Permite gestionar las necesidades del proyecto en forma estructurada: Cada actividad del desarrollo del proceso consiste de una serie de pasos organizados y bien definidos.
- Mejora la capacidad de predecir cronogramas de proyectos, así como sus resultados: proporciona un punto de partida para controles subsecuentes y actividades de mantenimiento, tales como estimación de costos, tiempo y recursos necesarios.
- Disminuye los costos y retrasos del proyecto: muchos estudios han demostrado que reparar errores por un mal desarrollo no descubierto a tiempo, es sumamente caro; especialmente aquellas decisiones tomadas durante la especificación de requerimientos.
- Mejora la calidad del software: la calidad en el software tiene que ver con cumplir un conjunto de requerimientos (funcionalidad, facilidad de uso, confiabilidad, desempeño, etc.).
- Mejora la comunicación entre equipos: La especificación de requerimientos representa una forma de consenso entre clientes y desarrolladores. Si este consenso no ocurre, el proyecto no será exitoso.
- Evita rechazos de usuarios finales: la especificación de requerimientos compromete al cliente a considerar sus requerimientos cuidadosamente y revisarlos dentro del marco del problema, por lo que se le involucra durante todo el desarrollo del proyecto.

En otras palabras “hacer un mejor trabajo definiendo y especificando software no sólo vale la pena sino que también es posible y ventajoso en costo” [Loucopoulos 1995]]. Para esto se deben tener en cuenta los siguientes aspectos:

- Cuanto más tarde en el ciclo de vida se detecta un error, más cuesta repararlo.

- Muchos errores permanecen latentes y no son detectados hasta bastante después de la etapa en que se cometieron.
- Se están cometiendo demasiados errores.
- Los errores de requerimientos son típicamente: hechos, incorrectos, omisiones, inconsistencias y ambigüedades.
- Los errores en los requerimientos pueden detectarse.

2.2.7. Perspectivas sobre la especificación de los requerimientos

La especificación de requerimientos debe atender a mejorar la gestión del cambio en la organización, integrar visiones dentro de la misma y vincular los Sistemas de Información con la estrategia organizacional. Respecto a esto, la especificación de requerimientos se comporta con una identidad aún mayor dentro del proceso de desarrollo de los sistemas de información y del sistema de software asociado a ellos, en lo que se ha dado en llamar Ingeniería de Requerimientos, el cuál se lo considera como un camino para gerenciar el cambio, involucrando las siguientes actividades:

- Comprensión conceptual del status actual.
- Definición del cambio.
- Implementación del cambio.
- Integración de esta nueva implementación.

De acuerdo a las normas ISO, un proceso es: un curso finito, único, de acontecimientos, definido por su propósito o por sus efectos, ejecutado bajo condiciones dadas. Basados en este concepto podemos definir el desarrollo de software como una secuencia de procesos, donde la determinación de requerimientos de software es un subproceso de este desarrollo.

El proceso de desarrollo de software involucra la generación de varios modelos y como dijimos anteriormente, puede verse como una serie de pasos que están orientados por objetivos y pueden considerarse transiciones entre representaciones o refinamientos de esas representaciones.

Sin importar el modelo en el que se genere, el desarrollo lo podemos tomar como una actividad o proceso de diseño que involucra:

- requerimientos a satisfacer
- output : un documento de especificación de diseño
- objetivo del diseñador: un diseño que su implementación satisfacerá los requerimientos
- el diseñador no conoce ningún diseño que satisface los requerimientos

2.2.8. Análisis de requerimientos

Antes de describir qué pasos deben cumplirse en esta actividad, debemos tener una definición clara del término "Problema".

"Un problema puede ser definido como la diferencia entre las cosas como se perciben y las cosas como se desean". Aquí vemos la importancia que tiene una buena comunicación entre desarrolladores y clientes; de esta comunicación con el cliente depende que entendamos sus necesidades.

A través de la definición de problema, podemos ver entonces que la actividad de "Análisis del Problema" tiene por objetivo que se comprendan los problemas del dominio de especificación, se evalúen las necesidades iniciales de todos los involucrados en el proyecto y que se proponga una solución de alto nivel para resolverlo.

Durante el análisis del problema, se realizan una serie de pasos para garantizar un acuerdo entre los involucrados, basados en los problemas reales del dominio de aplicación.

Estos pasos son los siguientes:

Comprender el problema que se está resolviendo: Es importante determinar quién tiene el problema realmente, considerar dicho problema desde una variedad de perspectivas y explorar muchas soluciones desde diferentes puntos de vista.

Construir un vocabulario común: Se debe confeccionar un glosario en dónde se definan todos los términos que tengan significados comunes (sinónimos) y que serán utilizados durante el proyecto. La creación de un glosario es sumamente beneficiosa ya que reduce los términos ambiguos desde el principio, ahorra tiempo, asegura que todos los participantes de una reunión están en la misma página, además de ser reutilizable en otros proyectos.

Identificar a los afectados por el sistema: Identificar a todos los afectados evita que existan sorpresas a medida que avanza el proyecto. Las necesidades de cada afectado, son discutidas y sometidas a debate durante la ingeniería de requerimientos, aunque esto no garantiza que vaya a estar disponible toda la información necesaria para especificar un sistema adecuado.

Para saber quiénes son las personas, departamentos, organizaciones internas o externas que se verán afectadas por el sistema, debemos realizar algunas preguntas.

- ¿Quién usará el sistema que se va a construir?
- ¿Quién desarrollará el sistema?
- ¿Quién probará el sistema?
- ¿Quién documentará el sistema?
- ¿Quién dará soporte al sistema?

- ¿Quién dará mantenimiento al sistema?
- ¿Quién mercadeará, venderá, y/o distribuirá el sistema?
- ¿Quién se beneficiará por el retorno de inversión del sistema?

Como vemos, debe conocerse la opinión de todo aquél que de una u otra forma está involucrado con el sistema, ya sea directa o indirectamente.

Definir los límites y restricciones del sistema: Este punto es importante pues debemos saber lo que se está construyendo, y lo que no se está construyendo, para así entender la estrategia del producto a corto y largo plazo. Debe determinarse cualquier restricción ambiental, presupuestaria, de tiempo, técnica y de factibilidad que limite el sistema que se va a construir.

2.3. MODELOS

2.3.1. Concepto

Un modelo es una representación de un objeto, sistema o idea, de forma diferente al de la entidad misma. El propósito de los modelos es ayudarnos a explicar, entender mejorar un sistema. Un modelo de un objeto puede ser una réplica exacta de éste o una abstracción de las propiedades dominantes del objeto [Torres 2006].

El uso de modelos no es algo nuevo. El hombre siempre ha tratado de representar y expresar ideas y objetos para tratar de entender y manipular su medio.

Un requerimiento básico para cualquier modelo, es que debe describir al sistema con suficiente detalle para hacer predicciones válidas sobre el comportamiento del sistema. Más generalmente, las características del modelo deben corresponder a algunas características del sistema modelado.

Un modelo se utiliza como *ayuda para el pensamiento* al organizar, clasificar y clarificar conceptos confusos e identificar inconsistencias. Al realizar un análisis de sistemas, se crea un modelo del sistema que muestre las entidades, las interrelaciones, etc. La adecuada construcción de un modelo ayuda a organizar, evaluar y examinar la validez de pensamientos.

Al explicar ideas o conceptos complejos, los lenguajes verbales a menudo presentan ambigüedades e imprecisiones. Un modelo es la representación concisa de una situación; por eso representa un medio de *comunicación* más eficiente y efectivo.

Un modelo deberá representar fiel y completamente los requisitos de los usuarios. Casi siempre estos requisitos son expresados de forma escasamente estructurados sin establecer ninguna correspondencia entre éstos y los demás elementos del modelo [Insfrán 2002].

El modelado es una parte central de todas las actividades que conducen a la producción de buen software. Construimos modelos para comunicar la estructura deseada y el

comportamiento de nuestro sistema. Construimos modelos para visualizar y controlar la arquitectura del sistema. Construimos modelos para comprender mejor el sistema que estamos construyendo, muchas veces descubriendo oportunidades para la simplificación y la reutilización [Booch 2006].

2.3.2. Clasificación

Los modelos pueden clasificarse de diversas maneras. Existen muchos modelos físicos tales como el modelo de un avión o, más generalmente, una réplica a escala de un sistema. Existen modelos esquemáticos que abarcan dibujos, mapas y diagramas. Existen modelos simbólicos, de los cuales los que están basados en las matemáticas o en un código de computadora, desempeñan funciones importantes en el diseño y estudio de la simulación de sistemas por medio de computadora.

Algunos modelos son estáticos; otros, dinámicos. Un modelo estático omite ya sea un reconocimiento del tiempo o describe un instante del estado de un sistema en determinado momento. En contraste, un modelo dinámico reconoce explícitamente el transcurso del tiempo. Además de proporcionar una secuencia de instantes del sistema en el transcurso del tiempo, algunos modelos dinámicos especifican relaciones entre los estados de un sistema en diferentes momentos.

2.3.3. Necesidad y costo del nivel de detalle

Cuando se construye un modelo, constantemente nos encontramos frente al problema de equilibrar la necesidad del detalle estructural con la de hacer manejable el problema para las técnicas de solución aplicables al dominio del problema. Siendo un formalismo, un modelo necesariamente es una abstracción. Sin embargo, cuanto más detallado sea un modelo en forma explícita, mejor será la semejanza del modelo con la realidad. Otra razón para incluir el detalle es que se ofrecen mayores oportunidades para estudiar el comportamiento del sistema cuando una relación estructural dentro del modelo altera con el propósito del sistema. Primero, puede considerarse un mayor número de combinaciones de los cambios estructurales y, segundo, puede estudiarse un mayor número de aspectos del comportamiento del dominio.

Por otra parte, un gran nivel de detalle, dificulta la comprensión de las soluciones propuestas a los requerimientos a satisfacer, incrementando el costo de la solución propuesta Sin embargo, el factor que sirve de límite en la utilización del detalle, es que a menudo no se tiene suficiente información sobre los propósitos del dominio en estudio, como para especificar características que debe satisfacer.

Todo modelo debe limitar el detalle en algún aspecto. Al hacer la descripción de un sistema en lugar del detalle, se hacen suposiciones sobre el comportamiento del sistema. Es

deseable que estas suposiciones no contradigan el comportamiento observable del sistema, debiendo comprobarlas comparándolas con el objeto de estudio.

2.3.4. Importancia de modelar

Hay límites a la capacidad humana de comprender la complejidad. A través del modelado reducimos el problema que se está estudiando, centrándonos en un aspecto cada vez. Atacar a un problema difícil dividiéndolo en una serie de problemas más pequeños que se puede resolver.

Un modelo escogido adecuadamente puede permitir al modelador trabajar con mayores niveles de abstracción. Cuanta más envergadura tenga el sistema a construir, hay más probabilidades que se fracase si no se construye el modelo adecuado. Todos los sistemas útiles e interesantes tienen la tendencia natural de hacerse más complejos con el paso del tiempo. Así que, aunque al inicio se puede pensar que no es necesario modelar, cuando el sistema evolucione de descubrirá la real necesidad y si no se lo ha hecho entonces será demasiado tarde [Booch 2006].

La fase de modelado de requisitos tiene una doble finalidad, que no podemos perder de vista en ningún momento. Por un lado debe permitir entender cabalmente las necesidades del usuario en el dominio del problema y representarlo de una forma sencilla, completa y sin ambigüedades. Por otro lado toda la información de requisitos capturada debe tener su representación en el Modelo Conceptual [Insfrán 2002].

2.3.5. Principios del modelado

En las disciplinas ingenieriles el uso del modelado tiene una larga y rica experiencia, la que sugiere cuatro principios básicos:

- La elección de cuál es el modelo que se utilizará para representar la realidad, tendrá una incidencia directa en la forma que tomará la solución.
- Todo modelo puede ser obtenido con diferentes modelos de precisión (nivel de granularidad)
- Todo modelo deben reflejar todas las características esenciales de la realidad (requisitos funcionales).
- Un único modelo o vista no es suficiente. Resulta más ventajoso disponer de un conjunto de vistas diferentes, que reflejarán múltiples puntos de vista.

2.3.6. Uso de metodologías en el modelado

Ahora se presenta la utilización de una metodología definida, la cual sirve para establecer un esquema conceptual sobre un determinado dominio. El término metodología,

sugiere la existencia y descripción de métodos estructurados, los cuales tienen por objetivos ayudar a desarrollar modelos de sistemas en forma sistemática.

Los métodos estructurados proveen un camino para el analista de manera de explorar algunos aspectos del sistema en detalle y sumar claridad a la visión difusa de los requerimientos por parte del cliente, en forma conjunta con las restricciones propias del dominio.

Los métodos estructurados, sin embargo, deben ocuparse no solo por rescatar la esencia a través de una percepción vaga de la noción del sistema, sino que también ocuparse de las actividades a realizar en la elaboración de los mismos.

Sin una metodología, pueden seleccionarse y aplicarse un conjunto de técnicas y notaciones en forma ad-hoc. Aún sin disponer de una orientación sobre qué aspectos del sistema a modelar, y que pasos aplicar para desarrollar el modelo. Esto es la diferencia principal de disponer de una metodología a limitarse a representar un conjunto de notaciones o técnicas.

En resumen se puede concluir que el objetivo de disponer de una metodología para el modelado en la representación de un esquema conceptual, servirá para clarificar y si la metodología lo contempla validar los requerimientos del usuario.

2.3.7. Características deseables de las metodologías/herramientas

A continuación se describirán una serie de aspectos deseables que deberán contener las metodologías/Herramientas, utilizadas para el establecimiento de un modelo.

2.3.7.1. Modelado iterativo y evolutivo

Las actividades de elicitación, especificación y validación, son repetidas varias veces en un proceso iterativo, en donde los requerimientos se van refinando y evolucionando a medido que avanza la construcción del modelo. Por lo tanto en cada iteración el modelo debe permitir identificar el origen del requerimiento, y el versionado actual de los mismos.

2.3.7.2. Diferentes vistas

A continuación se presentan diferentes modelos o vistas, que son importantes que estén presentes en cualquier conjunto de Metodologías/Herramientas para facilitar la comprensión del sistema, ellas son:

- **Vista Estática:** Esta vista debe proporcionar, y especificar con detalle, las propiedades estáticas, de manera de soportar toda la funcionalidad requerida al modelo.

- **Vista Dinámica:** Explicita el ciclo de vida de sus objetos, y las interrelaciones que se producen entre los mismos. Puede estar constituido por la secuencia válida que caracterice su comportamiento, incluyendo la interrelación entre los distintos objetos.
- **Vista funcional:** Debe especificar en forma declarativa como cada servicio ante un estímulo, se producen los cambios de estados en sus atributos.

2.3.7.3. Identificación y trazabilidad de los requerimientos

Para todos los requerimientos funcionales, debe ser posible hacer un seguimiento durante todo el proceso de modelado desde la identificación y formulación por parte del usuario hasta su efectivo cumplimiento plasmado en la funcionalidad del modelo.

2.3.7.4. Documentación proporcionada por el modelo

El producto o salida de esta etapa debe servir con un doble propósito. Por un lado de estar destinado al cliente/usuario de manera de certificar y validar los requisitos a satisfacer y por otro lado uno eminentemente técnico, que es el de servir como insumo en las restantes etapas de la construcción del sistema de software.

2.3.7.5. Flexibilidad para cambio de requisitos

El modelo debe ser flexible permitiendo introducir cambios, en donde la herramienta deberá realizar en forma automática, o lo más automática posible atender el impacto que producirá en el resto del sistema.

2.3.7.6. Reducir las ambigüedades

El lenguaje natural es inherentemente ambiguo, por lo tanto se deberá procurar llevar a una notación que permita reducir la ambigüedad del lenguaje del usuario, y en lo posible unificar el léxico empleado por el usuario.

2.3.7.7. Traducir a lenguaje técnico los requerimientos

Debe permitir, ser tratados a los efectos de llevarlos a un lenguaje que se vaya aproximando al lenguaje técnico, utilizado en las próximas etapas.

2.4. CRITERIOS DE COMPARACIÓN

A continuación se describirán las diferentes dimensiones de análisis que se utilizarán de manera de permitir realizar una comparación de las diferentes Metodologías/Herramientas de manera de poder construir un instrumento de evaluación que permita efectuar tal comparación.

Este conjunto de criterios a ser valorados, ha sido obtenido en base a las consideraciones sobre esquemas conceptuales, requerimientos y modelos vertidos en el presente trabajo, contrastándolo con los trabajos realizados por otros autores como lo tratado en [Robertson 1997], [Olsina 1999], [Andriano 2006]

2.4.1. Dimensiones de análisis

2.4.1.1. Metodología

- Claridad Conceptual.
- Potencialidad para abstraer esencia del dominio.
- Identificación de la fuente
- Reducción de ambigüedades sobre conceptos y manejo de sinónimos
- Facilidad de aplicación y flexibilidad para adopción de criterios de diseño
- Facilidad de entendimiento por parte del usuario del Dominio.
- Mantenibilidad del modelo
- Reutilización del modelo.
- Documentación del modelo.
- Jerarquización de los requerimientos del modelo
- Validación del modelo resultante.
- Versionado en proceso Iterativo.
- Facilidad de trazabilidad de requerimientos.
- Producto como insumo para la construcción del sistema modelado.

2.4.1.2. Herramienta

- Facilidad de instalación y configuración.
- Curva de Aprendizaje de la Herramienta
- Capturar en forma fiel y precisa la abstracción del modelo.
- Facilidad de introducir cambios en el diseño.
- Soporte a proceso iterativo y manejo de versionado.
- Visualización a través de diferentes vistas del modelo.
- Uso de notaciones y simbología que faciliten el entendimiento del usuario.

- Uso de notaciones y simbología que faciliten el entendimiento del equipo de desarrollo en la construcción del modelo.
- Facilidad de mapeo directo del modelo para la construcción del sistema.
- Portabilidad de la herramienta.

CAPÍTULO 3

RESUMEN *En este capítulo se presenta una metodología orientada al cliente que se fundamenta en base a una documentación integrada en una estructura denominada Client Oriented Requeriments Baseline, la cual se basa en el Léxico Extendido del Lenguaje (LEL) para el modelado del vocabulario, y mediante el uso de Escenarios para representar su comportamiento. LEL y Escenarios serán utilizados para capturar la esencia del dominio, utilizando el Baseline Mentor Workbench (BMW) como herramienta de soporte.*

3.1. CLIENT ORIENTED REQUIREMENTS BASELINE

3.1.1. Marco conceptual

Este modelo propone un enfoque de captura de requerimientos para la obtención de un modelo conceptual, siendo la idea principal que el usuario y el desarrollador compartan el mismo lenguaje. Leite sugiere que en particular el uso del lenguaje propio del usuario mejora considerablemente esta comunicación [Leite 1989]. En el proceso de Ingeniería de Requerimientos y la validación de los mismos a lo largo del ciclo de vida de desarrollo se requiere una fuerte de interacción con el usuario [Loucopoulos 1995], tarea que se facilita con el uso del vocabulario común entre el usuario (experto del dominio) y el analista.

Los requerimientos necesitan de un modelo del contexto del dominio que le den un marco. Leite propone un modelo de contexto en dos etapas [Leite 1997]. Primero sugiere comprender el lenguaje del dominio del problema y luego estudiar la dinámica de éste. Por tal motivo desarrolló el LEL (Léxico Extendido del Lenguaje) y un modelo particular de escenarios para atacar cada uno de los problemas. Estas técnicas resultan apropiadas tanto para el Ingeniero de Requerimientos como para el experto del dominio.

Por esta razón se plantea trabajar con una metodología orientada al cliente propuesta por Leite [Leite 1997], que trabaja con una documentación integrada en una estructura llamada Requirements Baseline [Leite 1995; Leite 1997] como soporte del proceso de desarrollo de software basándose en el LEL (Léxico extendido del lenguaje) como elemento principal para modelar el vocabulario del sistema, y mediante el uso de escenarios para representar su comportamiento.

El Léxico extendido del Lenguaje y los escenarios se utilizan para capturar y abstraer conocimiento del dominio en el cual se utilizará un sistema de software. A partir de ambos se

obtienen en forma sistemática tarjetas de colaboraciones y responsabilidades de las clases (CRC).

3.1.2. Requirements baseline

Es un enfoque orientado al cliente propuesto por Leite [Leite 1997], que puede verse en la Figura 3.1, y se basa en un metamodelo que contiene descripciones sobre el universo de discurso y el sistema de software que será construido dentro de ese universo de discurso. Estas descripciones son escritas en lenguaje natural siguiendo patrones determinados y relacionadas entre sí. El uso de lenguaje natural posibilita validar en todo momento con los usuarios del dominio, las especificaciones obtenidas en el análisis de requerimientos. Por otra parte la Requirements Baseline es una estructura que evoluciona dinámicamente a lo largo del proceso de desarrollo, acompañando las tareas de mantenimiento y en forma independiente del proceso de desarrollo que se utilice.

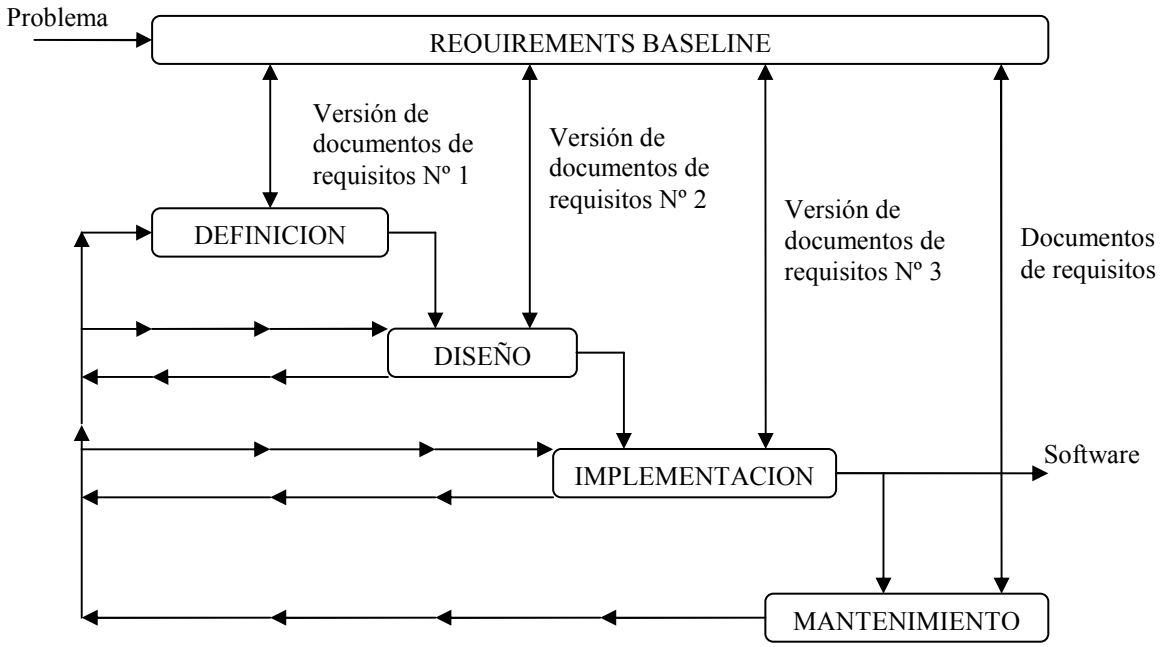


Figura 3.1. Requirements baseline y el proceso de desarrollo de software [Leite 1995]

3.1.3. Vistas del modelo

La Requirements Baseline, de acuerdo con [Leite 1997], se compone de cinco vistas:

- ***Vista del modelo léxico LEL***

  La vista léxica está basada en un metamodelo que brinda soporte a la elicitación del lenguaje de los requerimientos del universo de discurso y se denomina léxico extendido del lenguaje (LEL).

- ***Vista Básica BMV***

  La vista básica contiene diagramas de entidad-relación con los requerimientos externos propuestos por los clientes.

- ***Vista de escenarios SMV***

  La vista de escenarios es dinámica y define el comportamiento de la aplicación en un momento determinado.

- ***Vista de hipertexto HV***

  La vista de hipertexto integra la vista léxica, la de escenarios y la vista básica, estableciendo vínculos entre los escenarios con sí mismos y con los el resto de los escenarios haciendo referencia a los términos contenidos en el LEL.

- ***Vista de configuración CV***

  La vista de configuración es un sistema de versionado que registra todos los cambios que se van produciendo en el LEL, escenarios y vista básica, mostrando la constante evolución del sistema a lo largo del ciclo de vida hasta el mantenimiento del sistema.

Las vistas básica y de configuración en la actualidad han dejado de tener continuidad por lo tanto nos enfocaremos en dos de las vistas que resultan ser las de mayor importancia: la vista Léxica LEL y la vista de escenarios SMW

3.1.3.1. Vista del modelo léxico LEL

El léxico extendido del lenguaje tiene por objetivo entender el vocabulario manejado por los usuarios del dominio, para que de esta forma se logre una mejor comunicación con éstos, entendiendo los términos que utiliza para expresarse sin preocuparse por entender el problema que se debe solucionar. Las terminologías utilizadas serán entonces representadas en el LEL como símbolos que describen el lenguaje del dominio del problema o universo de discurso, que con frecuencia son aquellas palabras o frases que más se repiten en las entrevistas y otras que se consideren relevantes para el dominio, puede ocurrir en un gran número de posibilidades, que las personas hagan referencia al mismo símbolo, utilizando diferentes terminologías, que llamaremos sinónimos, para representarlos en las entradas del LEL se utiliza como separador el elemento ” / ”.

A continuación se presenta la Fig. 3.2., donde se muestra el modelo del léxico extendido del lenguaje

LEL: representación de los símbolos en el lenguaje del dominio de la aplicación. Sintaxis: $\{Símbolo\}_1^N$

Símbolo: entrada del léxico que tiene un significado especial en el dominio de la aplicación.
Sintaxis:
$\{Nombre\}_1^N + \{Noción\}_1^N + \{Impacto\}_1^N$

Nombre: identificación del símbolo. Más de uno indica la presencia de sinónimos.
Sintaxis:
Palabra | Frase

Noción: denotación del símbolo. Debe ser expresada usando referencias a otros símbolos y usando el vocabulario mínimo.
Sintaxis:
Oración

Impacto: connotación del símbolo. Debe ser expresado usando referencias a otros símbolos y usando el vocabulario mínimo.
Sintaxis:
Oración
Donde:
Oración está compuesta solamente por Símbolos y No Símbolos, éstos pertenecientes al vocabulario mínimo.
+ significa composición, {x} significa cero o mas ocurrencias de x, | representa or

Figura 3.2. El modelo del léxico extendido del lenguaje [Kaplan 2000]

Como se puede ver en la Tabla 3.1., los símbolos deben ser descriptos en dos términos: noción e impacto, la noción representa el significado del símbolo y el impacto las connotaciones o efectos que tendrá el símbolo en el sistema.

Tabla 3.1. Descripción de una entrada de LEL

Entrada de LEL - Sinónimos	Identificador del símbolo
Noción	Denota el significado del símbolo
Impacto	Connotación o repercusión en el sistema

La descripción de los símbolos debe desarrollarse cumpliendo dos reglas [Hadad 1997] [Leite 1993], que se deben cumplir en forma simultánea:

- Principio de circularidad: acotando el lenguaje en función del dominio mediante la maximización de símbolos del lenguaje del LEL, que se logra utilizando en las definiciones de noción e impacto símbolos ya descriptos dentro del LEL.
- Principio del vocabulario mínimo: en donde la tarea es minimizar el uso de símbolos externos al dominio de la aplicación.

Por medio de estos dos principios se logra un vocabulario en forma de red altamente vinculado y auto contenido que permite representar al LEL con formato de hipertexto [Leite 1990].

Los símbolos del LEL pueden definir objetos o entidades pasivas, sujetos o entidades activas, verbos y estados, según sea el tipo de símbolo, sus nociones e impactos tienen una semántica diferente, que se muestra en la Tabla 3.2. [Leonardi 2001].

Tabla 3.2. Heurísticas en la definición de los símbolos del modelo léxico.[Leite1997][Leonardi 2001]

Sujeto	*Nociones:* describen quien es el sujeto.
	Impactos: registran acciones ejecutadas por el sujeto
Objeto	*Nociones:* definen al objeto e identifica a otros términos con los cuales el objeto tiene algún tipo de relación.
	Impactos: describen las acciones que pueden ser aplicadas al objeto.
Verbo	*Nociones:* describen quien ejecuta la acción, cuando ocurre, y cuales son los procedimientos involucrados.
	Impactos: describen las restricciones sobre la acción, cuáles son las acciones desencadenadas en el ambiente y las nuevas situaciones que aparecen como resultado de la acción.
Estado	*Nociones:* describen que significa y que acciones pueden desencadenarse como consecuencia de ese estado.
	Impactos : describen otras situaciones y acciones relacionadas

3.1.3.2. Vista del modelo de escenarios SMV

Los escenarios describen la funcionalidad del sistema en un momento determinado teniendo como objetivo principal comprender el sistema en su totalidad. Los aspectos dinámicos del modelo del contexto son capturados a través de los escenarios.

Las características principales son:

- Un escenario describe situaciones con énfasis en la descripción del comportamiento.
- Un escenario utiliza la descripción textual como representación básica.
- Un escenario está naturalmente ligado al LEL al describirse con el vocabulario definido en el este último.

Los escenarios se vinculan fuertemente al LEL ya que lo adoptan como referencia, y adoptan la siguiente estructura

- Título: identifica al escenario, puede ser uno o varios.
- Objetivo: meta a lograr en el Macrosistema.
- Contexto: describe la ubicación geográfica y temporal del escenario, así como un estado inicial o precondición.
- Recursos: son los medios de soporte, dispositivos que se necesita estén disponibles en el escenario.

- Actores: son las personas o estructuras de organización que tienen un rol en el escenario.
- Episodios: son una serie ordenada de sentencias escritas de manera simple, que posibilitan la descripción de comportamiento. Pueden ser opcionales, condicionales o simples, y estar agrupados según su forma de ocurrencia en grupos secuenciales o no secuenciales. Para cada escenario se prevé la descripción de excepciones: causas y soluciones a situaciones que discontinúan su evolución natural, e impiden el cumplimiento del objetivo. Estas generalmente reflejan la falta o mal funcionamiento de un recurso. El tratamiento de la excepción puede estar dado por un escenario.

Se puede hacer una analogía de los escenarios con la metodología de [Jacobson 1992] de casos de uso, que es utilizada ampliamente en la actualidad, con algunas diferencias. Los escenarios pueden ser vistos como casos de uso donde también encontramos un objetivo como meta a cumplir, un contexto, recursos asociados, actores, etc. Siendo los episodios una suerte de descripción de los cursos normales y alternativos de los casos de uso. Leonardi plantea un conjunto de diferencias [Leonardi 2001]:

- Un escenario describe situaciones del Macrosistema.
- Un escenario evoluciona durante el proceso de desarrollo del software siguiendo la filosofía de la Requirements Baseline a la cual pertenece.
- Los escenarios están naturalmente conectados al LEL.
- Un escenario describe situaciones, poniendo énfasis en el comportamiento del Macrosistema. Usa lenguaje natural para su representación.

En la Figura 3.3 se presenta un modelo de entidad-relación para un escenario [Leite 1997]:

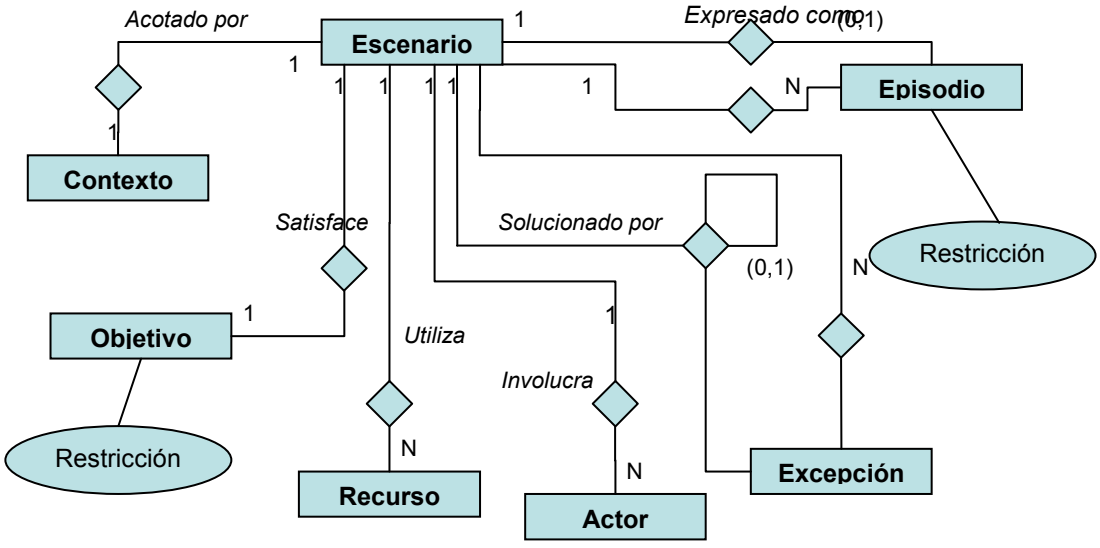


Figura 3.3. Diagrama de entidad-relación para el modelo de escenarios [Leite 1997]

Cada episodio de un escenario puede ser representado como un escenario en sí mismo. Por otro lado también pueden representarse los requerimientos no funcionales del dominio como restricciones.

3.1.4. Tarjetas CRC

Las tarjetas CRC (colaboraciones y responsabilidades de las clases) son obtenidas a partir del LEL y los escenarios; y son utilizadas para identificar las posibles clases que corresponden a la solución orientada a objetos.

Al igual que en un modelo de objetos cada tarjeta CRC representa un objeto del mundo real, estos objetos no son solamente físicos, sino que también se modelan como objetos aquellas entidades abstractas que cumplen un rol definido en el dominio del problema y de la solución. Las tarjetas CRC se obtienen a partir del LEL y los escenarios a través de heurísticas de derivación que nos permite encontrar las responsabilidades y colaboraciones de las clases que pasarán a ser de análisis, pudiendo transformarse en clases de diseño y posteriormente las del modelo preliminar de objetos.

3.2. PROCESO METODOLÓGICO: LEL, ESCENARIOS Y TARJETA CRC

Para el proceso de construcción del LEL y escenarios [Hadad 1997; Doorn 1998; Hadad 1999; Leite 2000] proponen hacerlo en dos etapas:

1. Construcción del LEL.

2. Construcción de los Escenarios a partir del LEL.

Esto se debe a la fuerte vinculación que tienen estos dos elementos.

3. A partir del LEL y escenarios se construyen las fichas CRC

3.2.1. Proceso de construcción del LEL

Como se muestra en la Figura 3.4, el modelo léxico se construye en etapas interdependientes, que a su vez pueden hacerse de forma simultánea:

- Entrevistas
- Generación de la lista de símbolos
- Clasificación de los símbolos
- Descripción de los símbolos
- Validación
- Control del LEL

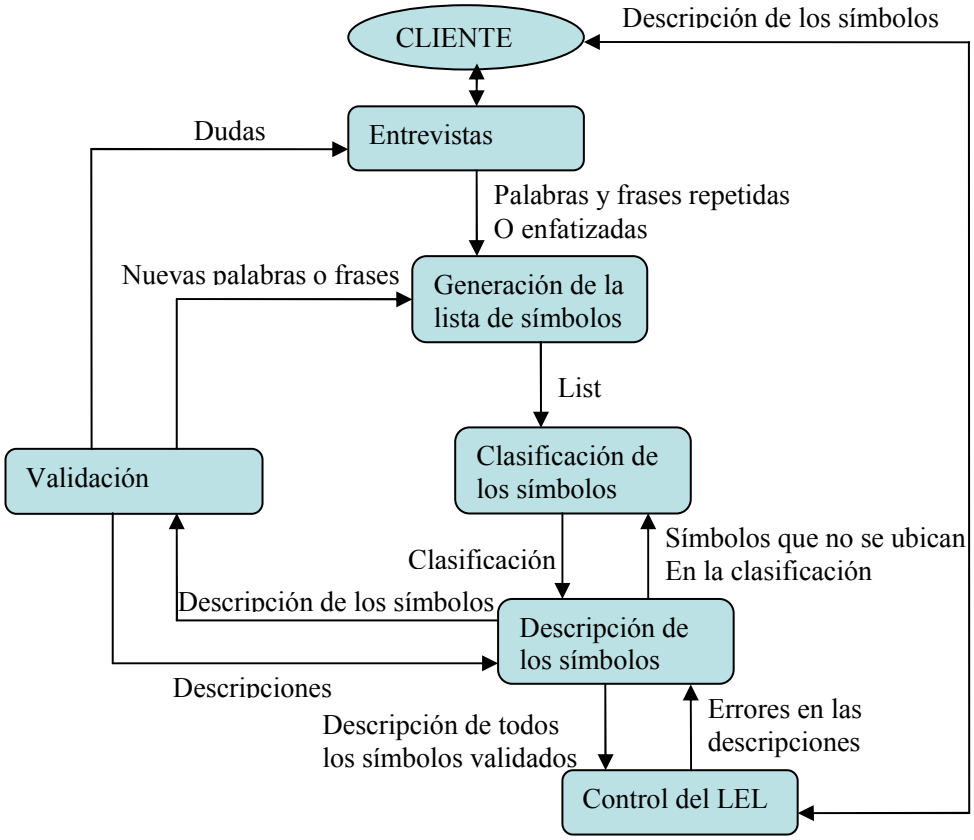


Figura 3.4. Etapas para la construcción del LEL [Hadad 1997]

3.2.1.1. Entrevistas

Las entrevistas tienen como objetivo conocer el lenguaje de los expertos del dominio, en las primeras entrevistas el entrevistador debe dejarlos expresarse libremente para que usen su propio vocabulario, a medida que la entrevista avanza, el entrevistado irá repitiendo algunos términos y a relacionarlos con otros utilizando de forma natural el principio de circularidad, el entrevistador por su parte, comienza a adquirir el lenguaje del dominio de la aplicación.

3.2.1.2. Generación de la lista de símbolos

Luego de las primeras entrevistas se arma una lista de símbolos candidatos, que contiene las palabras y frases que los expertos del dominio han utilizado con mayor frecuencia. Con el transcurso de las entrevistas la lista de símbolos candidatos se modifica validando símbolos, sinónimos, se agrega nuevo conocimiento y se eliminan las entradas erróneas armando la lista definitiva de símbolos. Otro método que puede utilizarse en forma simultánea a las entrevistas es la lectura de los documentos de la organización, ya que puede brindar al analista gran cantidad de información del lenguaje utilizado en el dominio.

3.2.1.3. Clasificación de los símbolos

La clasificación de los símbolos se utiliza para asegurar la integridad y homogeneidad de las descripciones [Hadad 1996], esta clasificación puede hacerse en forma general agrupando los símbolos según sujeto, verbo, objeto, estado, o mediante algún método particular.

3.2.1.4. Descripción de los símbolos

Con el desarrollo de las entrevistas el analista va tomando conocimiento del significado de los símbolos que se incluyen en la lista. La descripción de los símbolos consiste en agregar ese significado a los mismos, en términos de la definición de su noción e impacto.

Este proceso puede soportarse a través de una serie de reglas tal cual están descriptas en [Antonelli 2003]:

- Un símbolo puede tener una o más nociones y cero o más impactos.
- Cada noción e impacto debe ser descrito con oraciones breves y simples.
- Nociones e impacto para un símbolo pueden representar diferentes puntos de vista o pueden ser complementarios.
- Las oraciones breves y simples de las nociones e impactos deben responder a los principios de circularidad y de vocabulario mínimo.
- Para los símbolos que son sujeto de una oración, los impactos deben indicar las acciones que realiza.
- Para los símbolos que cumplen el rol de verbo, las nociones deben decir quién ejecuta la acción, cuándo sucede y el proceso involucrado en la acción.
- Para los impactos se deben identificar las restricciones sobre la realización de la acción, qué es lo que origina esta acción y qué es lo que causa esta acción.
- Para los símbolos que son objetos de una oración, la noción debe identificar otros objetos con los cuales se relaciona y los impactos serán las acciones que se pueden realizar con este signo.

3.2.1.5. Validación

La validación con los expertos del dominio permite corroborar el lenguaje del vocabulario del dominio que se obtuvo en etapa de entrevistas, validando el significado de los símbolos en las primeras entrevistas, mientras que en las siguientes validaciones se chequea

por lo general que el vocabulario obtenido sea el correcto, si falta información se introduce el nuevo conocimiento en el LEL.

3.2.1.6. Control del LEL

El control del LEL tiene como finalidad verificar que el Léxico terminado sea consistente y homogéneo, se controlan los símbolos verificando que pertenezcan a la clasificación correcta y que no existan sinónimos como símbolos diferentes, es fundamental que este proceso comience con las primeras descripciones de los símbolos.

3.2.2. Proceso de construcción de escenarios

La construcción de los escenarios se realiza a partir de la derivación del LEL, los pasos para realizar esta derivación son los siguientes:

- Identificación de los actores de la aplicación.
- Generación de la lista de escenarios candidatos, a partir de los actores Principales.
- Descripción de los escenarios candidatos, provenientes de los actores Principales.
- Ampliación de la lista de escenarios candidatos, a partir de los actores Secundarios.
- Descripción de los escenarios candidatos, provenientes de actores Secundarios.
- Revisión de los escenarios.
- Validación de escenarios.

El Proceso se muestra en la Figura 3.5 a continuación:

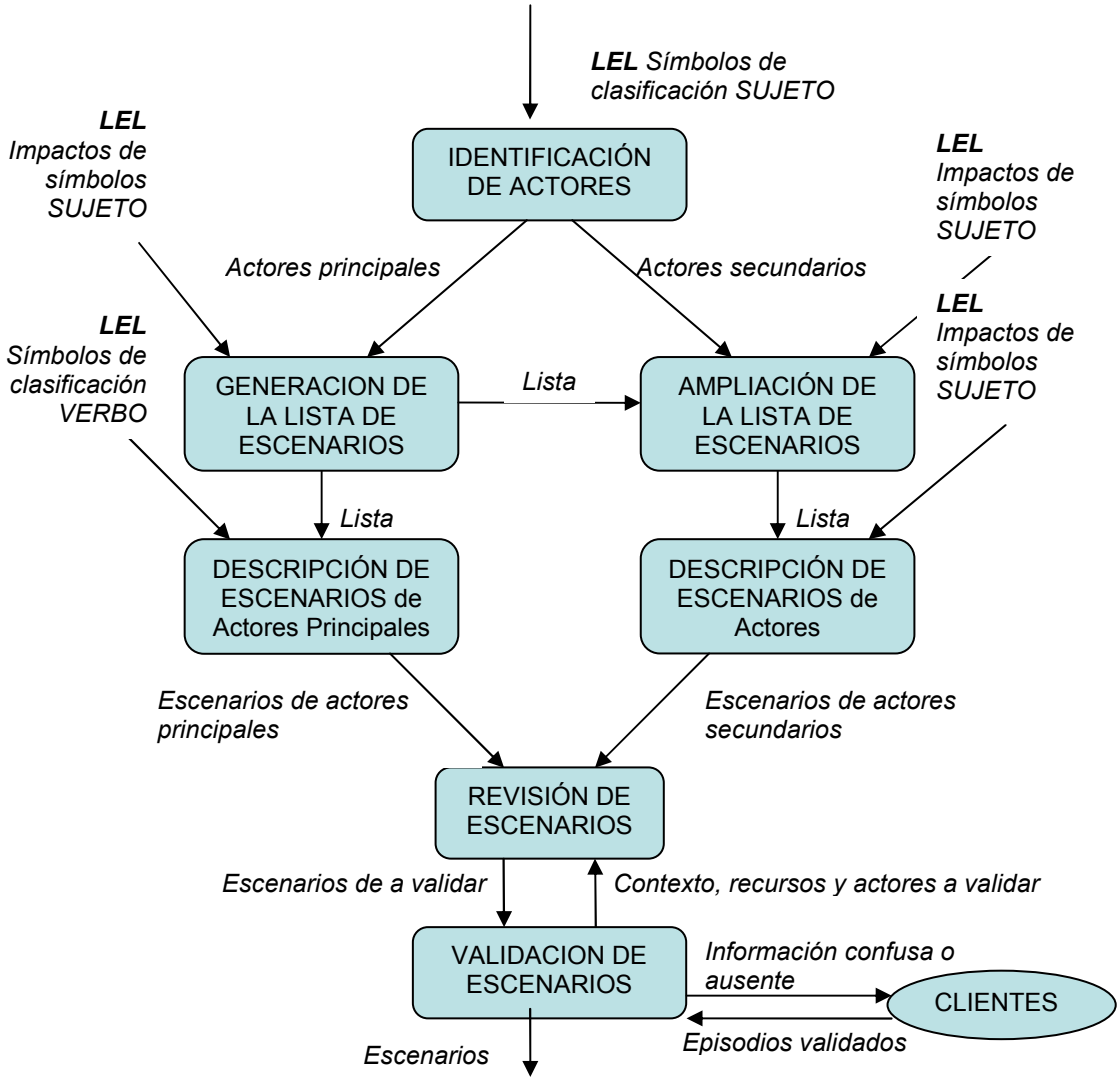


Figura 3.5. Etapas para la construcción de escenarios a partir del LEL [Hadad 1997]

Se define como actores a los usuarios de la aplicación y deben ser englobados bajo la categoría sujeto en el LEL, los actores serán primarios o secundarios según el rol que desempeñe cada usuario, los primarios son los que se encuentran en contacto directo con la aplicación, el secundario solo recibe información. La acción descripta como impacto en el LEL de los actores serán escenarios candidatos y se describen con la misma acción en infinitivo, eliminando aquellos que puedan repetirse. Los escenarios son el desarrollo de la acción que realiza el actor, en donde se especifican las precondiciones que debe cumplir esa acción, actores intervinientes (SUJETOS), recursos involucrados (OBJETOS), descripción de cada episodio y un objetivo que los actores deben cumplir. Luego se describe cada escenario utilizando la información contenida en el LEL. En este proceso se comienzan definiendo escenarios generales que representan toda la funcionalidad del sistema, luego se van refinando en detalle dando lugar a sub-escenarios que también son generados cuando uno o más episodios necesitan llevarse en forma independiente.

Una vez terminada la descripción de los escenarios se hace una revisión de los escenarios para verificar su consistencia y se arma una lista con los escenarios propuestos que serán validados con los expertos del dominio en posteriores entrevistas, en donde también se evacuarán las dudas surgidas durante el proceso para realizar las correcciones necesarias.

Luego de inspeccionar y validar los escenarios se arman los escenarios integradores, que no son otra cosa que una jerarquía de los escenarios y sub-escenarios ya descriptos, conformando grupos en función del contexto y del objetivo a cumplir. En cada grupo se debe establecer un orden de ejecución de escenarios que lo componen, en donde cada uno de ellos pasarán a ser episodios del escenario integrador que también debe ser descrito como un escenario pero teniendo en cuenta el contexto conformado por la combinación de los escenarios que lo componen, uniendo también recursos y actores que pasarán a ser los del escenario integrador.

En el caso de que un escenario no se incluya en ningún escenario integrador se debe analizar su pertenencia al dominio o si es una excepción no detectada.

3.2.3. Proceso de construcción de tarjetas CRC

Las Tarjetas CRC se obtienen a través de la derivación del LEL y escenarios, pudiendo dividirse en tres partes [Leonardi 2001]

3.2.3.1. Encontrar CRC primarias.

Las tarjetas CRC primarias representan objetos con comportamiento relevante dentro del dominio. Los actores de los escenarios son candidatos a convertirse en CRC primarias ya que realizan las tareas del escenario para lograr su objetivo. Los actores son entradas de LEL de la categoría sujeto. Estos sujetos realizan acciones en el dominio, las que son descriptas en los impactos. Entonces, los impactos no son otra cosa más que los servicios que provee, estos servicios son las responsabilidades de la CRC. De esta forma la tarjeta CRC es obtenida de un actor de un escenario y sus responsabilidades son los impactos de la entrada del LEL.

3.2.3.2. Encontrar CRC secundarias

Las CRC secundarias son aquellos colaboradores de las CRC primarias que las ayudan a cumplir con sus responsabilidades. Las CRC secundarias se encuentran en las responsabilidades de las CRC primarias. Aquellos términos de las responsabilidades que también están definidos en el LEL se convierten en CRC secundarias. Las responsabilidades se obtienen de la misma forma que se obtienen las responsabilidades de las CRC primarias.

3.2.3.3. Encontrar colaboraciones

Debido a que las CRC secundarias se obtienen a partir de las responsabilidades de las CRC primarias, es trivial que colaboran en alguna medida. Sin embargo, para completar las

colaboraciones, es necesario analizar los escenarios. En los episodios de los escenarios se deben buscar las entradas de LEL que originan a las CRC tanto primarias como secundarias. Las tarjetas CRC que participan de un mismo escenario colaboran entre ellas.

Como las tarjetas CRC se obtienen a partir de la derivación del LEL y escenarios, cualquier cambio producido en el LEL y los escenarios llevará a producir una nueva derivación de tarjetas CRC, hecho de gran importancia, ya que permite mantener una trazabilidad directa entre los requerimientos y el modelo de objetos que les darán soporte.

3.3. HERRAMIENTAS AUTOMATIZADAS DE SOPORTE

3.3.1. Necesidad de contar con una herramienta automatizada

Para poder identificar dentro de un determinado dominio el vocabulario, su impacto, las relaciones y reflejar los procesos para dar cumplimiento a todos los requerimientos, es necesario contar con una herramienta automatizada. Baseline Mentor Workbench (BMW), una herramienta que nos permite trabajar con LEL, escenarios y tarjetas CRC.

3.3.2. Baseline mentor workbench (BMW)

Es una herramienta que tiene como función asistir al experto del dominio durante la fase de ingeniería de requerimientos utilizando la metodología del Client Oriented Requirements Baseline [Antonelli 1999], gestionando las entradas del LEL, escenarios y tarjetas CRC.

Cada dominio dentro de BMW se denomina proyecto, y su evolución a lo largo del tiempo se registra en un sistema de versionado. Las entradas del LEL deben ser ingresadas en forma manual al igual que los escenarios, luego las tarjetas CRC son obtenidas en forma automática por la herramienta. Lógicamente, cada una de las versiones obtenidas conforme a la evolución del proyecto y sus modificaciones contienen sus propias entradas de LEL, escenarios y tarjetas CRC.

La arquitectura de la aplicación, que se muestra en la Figura 3.6., es una típica Arquitectura de Repositorio [Sommerville 1995], se compone de cuatro subsistemas: editor de entradas de LEL, editor de escenarios, generador de tarjetas CRC y browser de navegación.

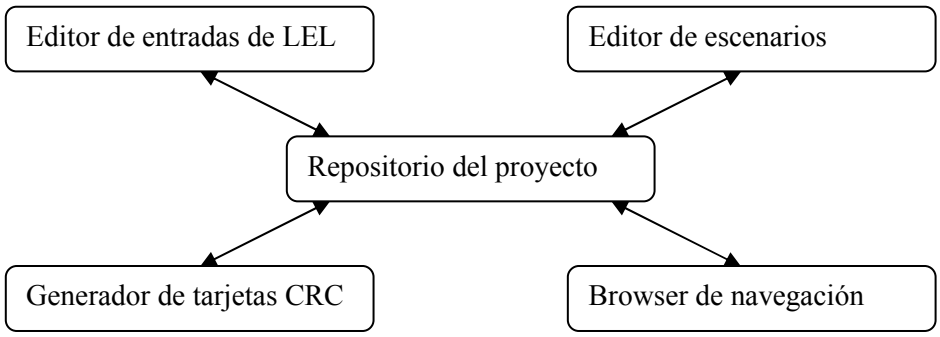


Figura 3.6. Arquitectura de Baseline Mentor Workbench [Antonelli 2003]

La descripción de las entradas del LEL y escenarios se realizan a través de templates que son seleccionados desde el menú contextual, de esta forma el usuario solo debe completar los templates con la información requerida, durante este procedimiento la aplicación verifica si las palabras que se ingresan existen dentro de las entradas del LEL, si es así se señala la palabra para indicar el link correspondiente en forma automática, si la palabra es nueva el usuario selecciona la palabra e indica que debe ser un link, de esta forma se agrega una nueva entrada al conjunto de entradas del LEL.

BMW deriva tarjetas CRC a partir de las entradas del LEL y los escenarios de acuerdo al algoritmo de derivación expuesto en [Leonardi 2001]. Sin embargo, la implementación en el BMW es una simplificación de este algoritmo que de todas formas cumple con las etapas principales [Antonelli 2003]:

- Hallar clases primarias. Son los actores de los escenarios que también están en el conjunto de entradas del LEL.
- Hallar responsabilidades de las clases primarias. Se toman de los impactos de las entradas del LEL.
- Hallar clases secundarias. Son las referencias a entradas del LEL que se encuentran en las responsabilidades de las clases primarias.
- Hallar responsabilidades de las clases secundarias. Se toman de los impactos de las entradas del LEL.
- Hallar colaboraciones. Consiste en determinar con que otras tarjetas CRC participa en la resolución de cada escenario.
- Depurar las tarjetas CRC. Consiste en revisar y eliminar tarjetas CRC y atributos repetidos

Cada entrada de LEL o escenario debe tratar de describirse siempre en términos de otras entradas, que es lo que se propone en el principio de circularidad, esto nos brinda la posibilidad de navegar por toda la información contenida dentro del LEL en un hipertexto.

CAPÍTULO 4

RESUMEN *En este capítulo se presentarán las Ontologías, partiendo desde su descripción, comportamiento, metodologías, lenguajes de representación y herramientas disponibles. Se procede a la selección del conjunto de Metodología, Lenguaje y Herramienta, los que serán utilizados para la comparación de metodologías para el desarrollo de la Ontología de manera de construir un Esquema conceptual*

4.1. ONTOLOGÍAS

4.1.1. Definición

Una ontología es un sistema de representación del conocimiento acerca de un dominio o ámbito específico, con el fin de obtener una representación formal de los conceptos que contiene y de las relaciones que existen entre dichos conceptos. Además, una ontología se construye en relación a un contexto de utilización especificando una conceptualización, por lo que cada ontología incorpora un punto de vista. Todas las conceptualizaciones (definiciones, categorizaciones, jerarquías, propiedades, herencia, etc.) de una ontología pueden ser procesables e interpretadas por una computadora.

Las ontologías tienen como propósito fundamental formular un esquema conceptual exhaustivo y riguroso de un dominio determinado, con la finalidad de facilitar la comunicación, reusar y compartir información entre organizaciones, computadoras y seres humanos. Por lo tanto definirá un vocabulario común el cuál incluirá la interpretación de los conceptos básicos del dominio y sus relaciones.

La definición más popular, extendida y de mayor aceptación de una ontología en informática es la propuesta por Gruber "*La especificación explícita, compartida y formal de una conceptualización*" [Gruber 1993].

Para lograr un mejor entendimiento y el alcance de la definición propuesta en el párrafo anterior, analizamos el contexto de la extensión de la definición propuesta en [Ramos 2007].

- **Conceptualización**: Modelo abstracto de un fenómeno, que puede ser visto como un conjunto de reglas informales que restringen su estructura. Por lo general se expresa como un conjunto de conceptos (entidades, atributos, procesos), sus definiciones e interrelaciones.
- **Formal**: Organización y definición de los términos utilizados, con sus relaciones como herramienta para el análisis de los conceptos de un determinado dominio

- **Compartida**: Captura de conocimiento consensual aceptado por una comunidad.
- **Explícita**: Se refiere a la especificación formal de los conceptos y a las restricciones sobre los mismos.

4.1.2. Orígenes

El término ontología tiene su origen en la filosofía, disciplina que trata de dar una explicación sistemática de la existencia; proviene de la conjunción de los términos griegos *ontos* y *logos* que significan respectivamente existencia y estudio. Fue definido originalmente por Aristóteles en su empeño de clasificar todo lo existente en el universo.

Desde la edad antigua hasta nuestros días, se ha disertado acerca de cuál puede ser la sustancia que concede a las cosas del mundo la propiedad de existir. De este modo una Ontología nos da el marco para entender la realidad, así como una clasificación de la misma de la cual podemos extraer los términos sobre los cuales se puede crear una abstracción de la realidad.

4.1.3. Objetivos de las ontologías

Los objetivos y beneficios logrados con el desarrollo de ontologías, se puede resumir de acuerdo a lo enunciado en [Noy 2005].

- ***Compartir entendimiento común de la estructura del conocimiento, entre personas o agentes de software***.

La ontología pone a disposición de los miembros de una comunidad los términos y conceptos del dominio de interés, lo cual permitirá a las personas o agentes de software extraer y agregar información según sus necesidades.

- ***Permitir reutilizar el dominio de conocimiento***

Es posible que muchos dominios hagan uso de un conocimiento específico, si este conocimiento está constituido en una ontología podrá ser reutilizado por aquellos individuos que la necesiten sin necesidad de desarrollar una ontología propia.

- ***Permitir separar conocimiento de dominio del conocimiento operacional***

Una ontología expresa el conocimiento del dominio de manera general de forma tal que pueda ser utilizado y manipulado por diversas técnicas o algoritmos.

- ***Analizar el conocimiento del dominio***

Específicamente en lo que se refiere al estudio de los términos y relaciones que lo configuran, ya sea formalmente o no.

4.1.4. Aplicación de ontologías en el modelado conceptual

Una Ontología, se basa en la idea de conceptualización, como una visión simplificada del mundo. De esta manera será el resultado por la cual la mente humana forma su representación mental acerca de algún evento o cosa.

De esta forma, las ontologías informáticas son representaciones del conocimiento que se tiene sobre algún dominio. Este conocimiento es descompuesto a través de conceptos, por lo que la representación del conocimiento se transforma en la representación de conceptos que de alguna forma están interrelacionados y generan dicho conocimiento o idea sobre el dominio a Modelar.

Una de las razones por las cuales las ontologías son objeto de estudio en las actividades de modelado, ya han sido vertidas cuando se analizaron los objetivos y ventajas que persiguen, sobre todo cuando tienen por finalidad el de facilitar la interoperatividad. De esta manera, las ontologías pretenden realizar una labor de estandarización de conceptos de tal manera que todos entendamos lo mismo cuando a algún término se refiera y de esta forma reducir ambigüedades y se eviten confusiones y falsas interpretaciones que conduzcan a especificaciones erróneas en el modelo.

Las dimensiones en que una ontología puede participar en el modelo conceptual sobre un dominio, está tratado y documentado en el trabajo realizado en [Sánchez 2005], en donde han analizado trabajos de desarrollos ontológicos, clasificando en dos tendencias bien marcadas a la utilización de ontologías en las actividades de modelado.

- Ontologías como base para la generación de Modelos

  Esta línea toma como punto de partida la concepción de una ontología, la cuál a través de transformaciones es convertida en un modelo. La característica principal es la tendencia a la utilización de ontologías y el principal motivo de la adopción de una ontología como base para el modelado, es que estas representan el conocimiento concentrado sobre un dominio, generalmente dado por una comunidad especializada. De esta manera los modelos logrados a través de esta técnica resultan más completos y en pocas iteraciones. También la ontología es utilizada como punto de referencia de comparación del Modelo final para ver si cumple o no con los requerimientos y la conceptualización inicial.

  La ontología permitirá dar una visión más amplia para la especificación del sistema y reducirá el número de aspectos no contemplados en la construcción del mismo. A través de un proceso de transformación que incluye los procesos de refinamiento, recorte y validación del modelo contra la ontología, se obtiene un modelo conceptual de mejor calidad.

- Ontologías generadas a partir de Modelos

  En este caso se trata de la obtención de una ontología a través de la transformación de uno o varios modelos. En este caso las ontologías cumplen en papel de conciliadoras de conceptos. De esto resulta que varios modelos sobre un mismo dominio deberían compartir una misma metodología

4.1.5. Clasificación de las ontologías

Las ontologías se pueden clasificar tomando en cuenta diferentes criterios, como por ejemplo, la intención de uso o tarea en particular, la formalidad del lenguaje utilizado para su construcción, la generalidad, entre otros. A continuación, se presentan algunas de estas clasificaciones:

4.1.5.1. Por su dependencia del contexto

Ontologías de Nivel Superior: Describen conceptos muy generales que son independientes de un problema particular o dominio. Por ejemplo, ontologías sobre el tiempo, el espacio, la materia, el objeto, el acontecimiento, la acción, entre otros

Ontologías de Dominio: Proporcionan el vocabulario necesario para describir un dominio dado. Incluyen términos relacionados con los objetos del dominio y sus componentes, un conjunto de verbos o frases que dan nombre a actividades y procesos que tienen lugar en ese dominio, y conceptos primitivos que aparecen en teorías, relaciones y fórmulas que regulan o rigen el dominio.

Ontologías de Tareas: Proveen un vocabulario sistemático de los términos usados para resolver problemas asociados con tareas particulares, ya sean dependientes o no del dominio.

Ontologías de Aplicación: Contienen las definiciones necesarias para modelar el conocimiento requerido para una aplicación particular en un dominio dado. Describen conceptos que dependen tanto del dominio particular como de las tareas. Estas ontologías son especializaciones de las ontologías de dominio y de tareas.

4.1.5.2. Por la granularidad de la conceptualización (cantidad y tipo de conceptualización)

Ontologías Terminológicas: Especifican los términos que son usados para representar conocimiento en el universo de discurso. Suelen ser usadas para unificar vocabulario en un dominio determinado (contenido léxico y no semántico).

Ontologías de Información: Especifican la estructura de almacenamiento de bases de datos. Ofrecen un marco para el almacenamiento estandarizado de información (estructura de los registros de una base de datos).

Ontologías de representación de conocimiento o Meta-ontologías: Especifican la conceptualización que subyace en un paradigma o formalismo de representación de conocimiento, es decir, proporcionan el vocabulario necesario para modelar otras ontologías utilizando un determinado paradigma de representación de conocimiento.

4.1.5.3. Por su propósito de uso

Ontologías para la comunicación entre personas: Proporcionan los términos necesarios para describir y representar un área de conocimiento. Una ontología informal (no ambigua) puede ser suficiente.

Ontologías para la interoperabilidad entre sistemas: Permiten realizar traducciones entre diferentes métodos, paradigmas, lenguajes y herramientas de software. La ontología se usa como un formato de intercambio de conocimiento.

Ontologías para beneficiar la ingeniería de sistemas: Favorecen la reutilización de componentes, facilitan la adquisición de conocimiento e identificación de requerimientos, y aumentan la fiabilidad de los sistemas al proporcionar consistencia en el conocimiento utilizado.

4.1.5.4. Por su nivel de formalidad

Se refiere al grado de formalismo del lenguaje usado para expresar la conceptualización.

Ontología altamente informal: Expresada en lenguaje natural (Glosario de términos).

Ontología informal estructurada: Expresada en una forma restringida y estructurada de lenguaje natural, que permite incrementar la claridad y reducir la ambigüedad.

Ontología semi-formal: Usa un lenguaje de definición formal.

Ontología rigurosamente formal: La definición de términos se lleva a cabo de manera meticulosa usando semántica formal y teoremas.

4.1.6. Elementos de las ontologías

Las ontologías proporcionan un vocabulario común de un área y definen, a diferentes niveles de formalismo, el significado de los términos y relaciones entre ellos.

El conocimiento en ontologías se formaliza principalmente usando cinco tipos de componentes: conceptos, relaciones, funciones, axiomas e instancias.

- Los conceptos, entidades o clases en la ontología se suelen organizar en taxonomías. Se suele usar tanto el término clases como conceptos. Un concepto puede ser algo sobre lo que se dice algo y, por lo tanto, también podría ser la descripción de una tarea, función, acción, estrategia, proceso de razonamiento, etc.

- Las relaciones representan un tipo de interacción entre los conceptos del dominio. Como ejemplos clásicos de relaciones binarias podemos mencionar: “subclase de” y “conectado a”.
- Las funciones son un tipo especial de relaciones en las que el n-ésimo elemento de la relación es único para los n-1 precedentes.
- Los axiomas son expresiones que son siempre ciertas. Pueden ser incluidas en una ontología con muchos propósitos, tales como definir el significado de los componentes ontológicos, definir restricciones complejas sobre los valores de los atributos, argumentos de relaciones, etc. verificando la corrección de la información especificada en la ontología o deduciendo nueva información.
- Las instancias se usan para representar elementos específicos de la ontología.

4.1.7. Aspectos a tener en cuenta al diseñar ontologías

De acuerdo a las recomendaciones efectuadas por diferentes autores, y al trabajo realizado en [Ramos 2007], se pueden enunciar un conjunto de características deseables en la construcción de ontologías, las cuales se pueden resumir en las siguientes:

- **Coherencia**: una ontología debe permitir hacer inferencias que sean consistentes con las definiciones (axiomas).
- **Completitud**: Todos los conceptos deben estar expresados en términos necesarios y suficientes.
- **Precisión**: debe hacerse la menor cantidad de "suposiciones" acerca del mundo modelado.
- **Claridad y objetividad**: una ontología debe poder comunicar de manera efectiva el significado de sus términos. Las definiciones serán lo más objetivas posibles y deben explicarse también en lenguaje natural.
- **Estandarización:** Siempre que sea posible, los nombres asignados a los términos deberán seguir un estándar, definiendo y respetando reglas para la formación de los mismos.
- **Máxima extensibilidad monótona:** Deberá ser posible incluir en la ontología especializaciones o generalizaciones, sin requerir una revisión de las definiciones existentes.
- **Principio de distinción ontológica:** Las clases de la ontología con diferente criterio de identidad, deberán ser disjuntas.

- **Diversificación de las jerarquías:** Para que la ontología se vea favorecida con los mecanismos de herencia múltiple, es conveniente usar tantos criterios de clasificación como sea posible, de manera de representar la mayor cantidad de conocimiento.
- **Minimización de la distancia semántica:** Conceptos similares deberán ser agrupados y representados utilizando las mismas primitivas.
- **Mínimo compromiso ontológico:** Una ontología debería imponer las menores exigencias posibles sobre el dominio que modela, es decir, se deben construir sólo los axiomas necesarios para representar el mundo a ser modelado.
- **Modularidad:** Al especificar una ontología se hacen definiciones de diferentes elementos como clases, relaciones y axiomas; tales definiciones se pueden agrupar en teorías que reúnen los objetos de una ontología más relacionados entre sí. Se puede lograr una organización altamente modular con máxima cohesión en cada módulo y mínima interacción, considerando que cada teoría es un módulo en la organización de la ontología. La modularidad permite flexibilidad y posibilidad de reuso de algunos módulos de la ontología.
- **Mínima dependencia con respecto a la codificación:** Una ontología debería permitir que los agentes que compartan los conocimientos, puedan ser implementados en diferentes sistemas y estilos de representación.

4.2. METODOLOGÍAS PARA DESARROLLAR ONTOLOGÍAS

A continuación se presentan diferentes metodologías para la construcción de ontologías, las cuales se presentan en orden cronológico. Esta sección no pretende describir en su totalidad a las mismas, sino más bien, brindar un panorama de las diferentes metodologías existentes, sus características esenciales y su evolución.

Comenzamos describiendo un conjunto de pasos propuestos para la metodología CYC. [Lenat 1990]. Posteriormente, en [Uschold 1995] se difunde la metodología empleada para la creación de la ontología TOVE (Toronto Virtual Enterprise) para modelar organizaciones. Al año siguiente, estos autores propusieron unas reseñas metodológicas para construir ontologías [Uschold 1996]. En [Bernaras 1996] se presentó un método para construir ontologías para redes eléctricas como parte del proyecto KACTUS. Methontology [Fernández 1997] apareció simultáneamente y fue extendido posteriormente [Fernández 1999], [Fernández 2000]. En 1997, se propuso otra metodología basada en la ontología SENSUS [Swartout 1997].

4.2.1. Metodología CYC

La metodología CYC consiste en varios pasos, los cuales fueron publicados en conjunto con las características más relevantes. [Lenat 1990]

En primer lugar hay que extraer manualmente el conocimiento común que está implícito en diferentes fuentes. A continuación, una vez que tengamos suficiente conocimiento en nuestra ontología, podemos adquirir nuevo conocimiento común usando herramientas de procesamiento de lenguaje natural o aprendizaje computacional. Esta metodología recomienda los siguientes pasos:

- Codificación manual de conocimiento implícito y explícito extraído de diferentes fuentes.
- Codificación de conocimiento usando herramientas software
- Delegación de la mayor parte de la codificación en las herramientas

4.2.2. Metodología de construcción de ontologías de Uschold y King

Esta metodología fue presentada en [Uschold 1995] y propone algunos pasos generales para desarrollar ontologías, a saber: (1) identificar el propósito; (2) capturar los conceptos y relaciones entre estos conceptos y los términos utilizados para referirse a estos conceptos y relaciones; (3) codificar la ontología. La ontología debe ser documentada y evaluada, y se pueden usar otras ontologías para crear la nueva. De esta forma se creó la Enterprise Ontology. Esta metodología recomienda los siguientes pasos:

- Identificar propósito
- Capturar la ontología
- Codificación
- Integrar ontologías existentes
- Evaluación
- Documentación

4.2.3. Metodología de construcción de ontologías de Grüninger y Fox

En esta metodología, presentada en [Grüninger 1995], el primer paso es identificar intuitivamente las aplicaciones posibles en las que se usará la ontología. Posteriormente, se usa un conjunto de preguntas en lenguaje natural, llamadas cuestiones de competencia, para determinar el ámbito de la ontología. Se usan estas preguntas para extraer los conceptos principales, sus propiedades, relaciones y axiomas, los cuales se definen formalmente en Prolog.

Por consiguiente, ésta es una metodología muy formal que se aprovecha de la robustez de la lógica clásica y que puede ser usada como guía para transformar escenarios informales en modelos computables. Esta metodología, que se usó para construir la ontología TOVE, recomienda los siguientes pasos:

- Escenarios motivantes
- Cuestiones informales de competencia
- Terminología formal
- Cuestiones formales de competencia
- Axiomas formales
- Teoremas de completitud

4.2.4. Metodología KACTUS

En esta metodología se construye la ontología sobre una base de conocimiento por medio de un proceso de abstracción. Cuantas más aplicaciones se construyen, las ontologías se convierten en más generales y se alejan más de una base de conocimiento. En otras palabras, se propone comenzar por construir una base de conocimiento para una aplicación específica. A continuación, cuando se necesita una nueva base de conocimiento en un dominio parecido, se generaliza la primera base de conocimiento en una ontología y se adapta para las dos aplicaciones, y así sucesivamente. De esta forma, la ontología representaría el conocimiento consensuado necesario para todas las aplicaciones. Esta metodología ha sido utilizada para construir una ontología para diagnosticar fallos, y recomienda seguir los siguientes pasos:

- Especificación de la aplicación
- Diseño preliminar basado en categorías ontológicas top-level relevantes
- Refinamiento y estructuración de la ontología

4.2.5. Metodología METHONTOLOGY

Methontology es una metodología para construir ontologías tanto partiendo desde cero como reusando otras ontologías, o a través de un proceso de reingeniería. Este entorno permite la construcción de ontologías a nivel de conocimiento, e incluye: (1) identificación del proceso de desarrollo de la ontología donde se incluyen las principales actividades (evaluación, gestión de configuración, conceptualización, integración, implementación, etc.); (2) un ciclo de vida basado en prototipos evolucionados; y (3) la metodología propiamente dicha, que especifica los pasos a ejecutar en cada actividad, las técnicas usadas, los productos a obtener y cómo deben ser evaluados. Esta metodología está parcialmente soportada por el entorno de desarrollo ontológico WebODE. Esta metodología ha sido usada en la construcción de múltiples ontologías, como una ontología química, ontologías hardware y software, etc. Se proponen los siguientes pasos:

- Especificación

- Conceptualización
- Formalización
- Implementación
- Mantenimiento

4.2.6. Metodología SENSUS

La metodología basada en SENSUS es un enfoque top-down para derivar ontologías específicas del dominio a partir de grandes ontologías. Los autores proponen identificar un conjunto de términos semilla que son relevantes en un dominio particular. Tales términos se enlazan manualmente a una ontología de amplia cobertura. Los usuarios seleccionan automáticamente los términos relevantes para describir el dominio y acotar la ontología SENSUS.

Consecuentemente, el algoritmo devuelve el conjunto de términos estructurados jerárquicamente para describir un dominio, que puede ser usado como esqueleto para la base de conocimiento. Esta metodología sirvió para construir la ontología Sensus y recomienda los siguientes pasos:

- Tomar una serie de términos como semillas.
- Enlazarlos manualmente.
- Incluir todos los conceptos en el camino que va de la raíz de Sensus a los conceptos semilla.
- Añadir nuevos términos relevantes del dominio.
- Opcionalmente, añadir, para aquellos nodos por los que pasan más caminos, su subárbol inferior.

4.2.7. Metodología On-To-Knowledge

El proyecto OTK aplica ontologías a la información disponible electrónicamente para mejorar la calidad de la gestión de conocimiento en organizaciones grandes y distribuidas. La metodología proporciona guías para introducir conceptos y herramientas de gestión de conocimiento en empresas, ayudando a los proveedores y buscadores de conocimiento a presentar éste de forma eficiente y efectiva. Esta metodología incluye la identificación de metas que deberían ser conseguidas por herramientas de gestión de conocimiento y está basada en el análisis de escenarios de uso y en los diferentes papeles desempeñados por trabajadores de conocimiento y accionistas en las organizaciones. Cada una de las herramientas de la arquitectura de OTK se centra en el desarrollo de aplicaciones dirigidas por ontologías y, finalmente, describe el uso y la evaluación de la metodología mediante casos de

estudio como, por ejemplo, la ontología Proper o AIFB. Esta metodología recomienda los siguientes pasos:

- Estudio de viabilidad
- Comienzo
- Refinamiento
- Evaluación

4.2.8. Selección de la metodología

En esta instancia, se toma las recomendaciones efectuadas en [Noy 2005] la cuál provee un simple guía de pasos para la construcción de ontologías. Estos autores, expresan que no existe ni una sola forma o ni una sola metodología "correcta" para desarrollar ontologías, proponiendo un conjunto de puntos generales que deben ser tomados en consideración y ofrecen uno de los procedimientos posibles para desarrollar una ontología.

Se basa en un proceso iterativo en donde se evolucionará y afinará completándola con detalles, en donde necesariamente el diseñador de la ontología tendrá que tomar decisiones con los pros y los contras de las diferentes alternativas de diseño.

Esta guía enfatiza algunas reglas fundamentales en el diseño de ontologías, las cuales pueden ayudarnos a la toma de decisiones de diseños en muchos casos

1. No hay una forma correcta de modelar un dominio - siempre hay alternativas viables. La mejor solución casi siempre depende de la aplicación que se tiene en mente y las futuras extensiones que se anticipan.
2. El desarrollo de ontologías es un proceso necesariamente iterativo.
3. Los conceptos en la ontología deben ser cercanos a los objetos (físicos o lógicos) y relaciones en tu dominio de interés. Esos son muy probablemente sustantivos (objetos) o verbos (relaciones) en oraciones que describen su dominio. Es decir, decidir para que vamos a usar la ontología y cuán detallada o general será la ontología guiará a muchas de las decisiones de modelado a lo largo del camino. Entre las varias alternativas viables, necesitaremos determinar cuál funcionará mejor para la tarea proyectada, cuál será más intuitiva, más extensible y más mantenible. Necesitamos también recordar que una ontología es un modelo de la realidad del mundo y los conceptos en la ontología deben reflejar esta realidad. Después de que hayamos definido una versión inicial de la ontología, podemos evaluarla y depurarla usándola en aplicaciones o métodos que resuelvan problemas o discutiéndola con expertos en el área. En consecuencia, casi seguramente necesitaremos revisar la ontología inicial.

Este proceso de diseño iterativo probablemente continuara a través del ciclo de vida completo de la ontología.

Para el desarrollo de la ontología dominio del presente trabajo utilizaremos esta guía de pasos definidas en [Noy 2005], ya que cuenta con una ejemplificación descriptiva que facilita el entendimiento de la misma.

4.3. LENGUAJES PARA EL DESARROLLO DE ONTOLOGÍAS

4.3.1. Fundamentos y evolución

El lenguaje XML proporciona una forma de escribir datos que es independiente de lenguajes, plataformas y herramientas, y proporciona una estructura sintáctica para que los mismos puedan ser interpretados por computadoras. Así mismo, XMLS (el lenguaje de esquemas de XML) permite la definición de gramáticas y etiquetas significativas para los documentos a través de namespaces o espacios de nombre. Sin embargo, XML o XMLS no son suficientes ya que aportan una estructura, pero no una semántica (la semántica estudia cómo los símbolos se refieren a los objetos).

A raíz de ello, surgió el lenguaje RDF, como un lenguaje para modelar los datos. El lenguaje RDF mediante recursos, propiedades (atributos y relaciones para describir recursos) y sentencias (combinación de recursos y propiedades) permite una representación explícita de la semántica de los datos. Sin embargo, RDF carece de poder expresivo (negación, implicación, cardinalidad, etc). Para lograr una mayor expresividad para el procesamiento semántico, se han desarrollado nuevos estándares para la representación de ontologías que se basan en RDF y RDF Schemas (RDFS). Las ontologías que utilicen estos lenguajes permitirán, entre otras cuestiones, distribuir definiciones autorizadas de vocabularios que soporten referencias cruzadas como los tesauros.

Un tesauro es una lista que contiene los "términos" empleados para representar los conceptos, temas o contenidos de los documentos, con miras a efectuar una normalización terminológica que permita mejorar el canal de acceso y comunicación entre los usuarios y las Unidades de Información

Por lo tanto, las ontologías de representación están pensadas para que tomen el papel que hasta ahora ocupaban los tesauros normalizados, pero es preciso un lenguaje estándar que especifique dichas ontologías con mayor precisión que los estándares ISO sobre tesauros. Como extensión de RDF se ha creado el lenguaje OWL, una especificación del W3C para especificar ontologías, basado en RDF.

Las ontologías requieren de un lenguaje lógico y formal para ser expresadas. En la inteligencia artificial se han desarrollo numerosos lenguajes para este fin, algunos basados en la lógica de predicados y otros basados en frames (taxonomías de clases y atributos), que

tienen un mayor poder expresivo, pero menor poder de inferencia; e incluso existen lenguajes orientados al razonamiento. Todos estos lenguajes han servido para desarrollar otros lenguajes aplicables a la Web. En un lenguaje de ontologías se pretenderá un alto grado de expresividad y uso.

De entro los principales lenguajes de ontologías podemos destacar los siguientes:

SHOE: Simple HTML Ontology Extensions. Fue el primer lenguaje de etiquetado para diseñar ontologías en la Web. Este lenguaje nació antes de que se ideara la Web Semántica. Las ontologías y las etiquetas se incrustaban en archivos HTML. Este lenguaje permite definir clases y reglas de inferencia, pero no negaciones o disyunciones.

OIL: Ontology Inference Layer. Este lenguaje, derivado en parte de SHOE, fue impulsado también por el proyecto de la Unión Europea On-To-Knowledge. Utiliza ya la sintaxis del lenguaje XML y está definido como una extensión de RDFS. Se basa tanto en la lógica descriptiva (declaración de axiomas) y en los sistemas basados en frames (taxonomías de clases y atributos). OIL posee varias capas de sub-lenguajes, entre ellas destaca la capa base que es RDFS, a la que cada una de las capas subsiguientes añade alguna funcionalidad y mayor complejidad. La principal carencia de este lenguaje es la falta de expresividad para declarar axiomas.

DAML y OIL: Este lenguaje nació fruto de la cooperación entre OIL y DARPA y unifica los lenguajes DAML (DARPA's Agent Markup Language) y OIL (Ontology Inference Layer). Se basa ya en estándares del W3C. El lenguaje DAML se desarrolló como una extensión del lenguaje XML y de Resource Description Framework (RDF) y para extender el nivel de expresividad de RDFS. DAML- OIL hereda muchas de las características de OIL, pero se aleja del modelo basado en clases (frames) y potencia la lógica descriptiva. Es más potente que RDFS para expresar ontologías. En la última revisión del lenguaje (DAML+OIL) ofrece ya un rico conjunto de elementos con los cuales se pueden crear ontologías y marcar la información para que sea legible y comprensible por máquina. También funciona como formato de intercambio. Sin embargo, este lenguaje presenta algunas carencias debido a su complejidad conceptual y de uso, complejidad que se intentó solventar con el desarrollo de OWL. No obstante, se desarrollaron muchas aplicaciones que utilizan DAML-OIL y también existen herramientas para convertir DAML a OWL.

OWL: OWL Web Ontology Language o Lenguaje de Ontologías para la Web es un lenguaje de etiquetado semántico para publicar y compartir ontologías en la Web. Se trata de una recomendación del W3C, y puede usarse para representar ontologías de forma explícita, es decir, permite definir el significado de términos en vocabularios y las relaciones entre aquellos términos (ontologías). En realidad, OWL es una extensión del lenguaje RDF y emplea las tripletas de RDF, aunque es un lenguaje con más poder expresivo que éste. Se trata

de un lenguaje diseñado para usarse cuando la información contenida en los documentos necesita ser procesada por programas o aplicaciones, en oposición a situaciones donde el contenido solamente necesita ser presentado a los seres humanos. OWL surge como una revisión al lenguaje DAML-OIL y es mucho más potente que éste. Al igual que OIL, OWL se estructura en capas que difieren en la complejidad y puede ser adaptado a las necesidades de cada usuario, al nivel de expresividad que se precise y a los distintos tipos de aplicaciones existentes (motores de búsqueda, agentes, etc.).

KIF: Knowledge Interchange Format es un lenguaje para representar ontologías basado en la lógica de primer orden. KIF está basado en la lógica de predicados con extensiones para definir términos, metaconocimiento, conjuntos, razonamientos no monotónicos, etc.; y pretende ser un lenguaje capaz de representar la mayoría de los conceptos y distinciones actuales de los lenguajes más recientes de representación del conocimiento. Se trata de un lenguaje diseñado para intercambiar conocimiento entre sistemas de computación distintos, diferentes lenguas, etc.; y no para la interacción entre seres humanos.

FOAF: aunque no es exactamente un lenguaje de ontologías ya que se trata de un vocabulario con definiciones que usa el lenguaje RDFS/OWL, FOAF hace más fácil que el software procese los términos del vocabulario FOAF para describir documentos. FOAF permite crear una base de datos unificada de información al normalizar una forma de definir categorías, tipos de relaciones, etc.

4.3.2. Selección del lenguaje

En el desarrollo del presente trabajo se utilizará como lenguaje de modelado de ontologías RDF, ya que es el lenguaje que cumple con todas las características deseadas, en lo que se refiere a poder expresivo y de representación para construir una ontología de dominio y resulta con mayor claridad dada su simplicidad y facilidad de implementación en todas las herramientas de desarrollo para modelar ontologías.

4.4. HERRAMIENTAS PARA EL DESARROLLO DE ONTOLOGÍAS

Los editores de ontologías son herramientas especializadas que apoyan la construcción de las mismas. Las facilidades que proporcionan van desde la definición y modificación de conceptos, propiedades, relaciones, axiomas y restricciones, hasta la inspección y navegación.

Una de las herramientas más utilizadas por los desarrolladores de ontologías es Protégé, la cual se describe a continuación:

4.4.1. Protégé

Protégé es un software libre de código abierto implementado en Java, desarrollado en la Universidad de Stanford, que permite la construcción de ontologías de dominio. Es capaz de operar como una plataforma para acceder a otros sistemas basados en conocimiento o aplicaciones integradas, o como una librería que puede ser usada por otras aplicaciones para acceder y visualizar bases de conocimiento. La herramienta ofrece una interfaz gráfica que permite al desarrollador de ontologías enfocarse en el modelado conceptual sin que requiera de conocimientos de la sintaxis de los lenguajes de salida.

El modelo de conocimiento de Protégé está basado en marcos (frames). Las primitivas de representación internas en Protégé pueden ser redefinidas declarativamente, permitiendo tener representaciones apropiadas para una variedad de lenguajes de ontologías. Las primitivas de representación (elementos de su modelo de conocimiento) proporcionan clases, instancias de esas clases, propiedades que representan los atributos de las clases y sus instancias, y restricciones que expresan información adicional sobre las propiedades. Protégé comprueba la entrada de datos nuevos, y no permite dos clases o atributos con el mismo nombre.

Protégé puede correr como una aplicación local o a través de un cliente en una comunicación con un servidor remoto. El navegador Web de Protege permite a los usuarios compartir, navegar y editar sus ontologías utilizando un navegador Web estándar, lo que proporciona un ambiente de colaboración que puede ayudar a las comunidades en el desarrollo de ontologías. Protégé ha sido utilizado como el ambiente de desarrollo primario para muchas ontologías, y se ha convertido en la herramienta más utilizada en el mundo para trabajar con OWL. La comunidad de usuarios de Protégé regularmente contribuye a mejorar la calidad del software y participa en grupos de discusión en línea dedicados a formular preguntas, realizar peticiones de nuevas características y cuestiones de soporte técnico.

Protégé presenta una gran capacidad de extensión, debido al soporte de conectores (plug-ins), que son aditivos que se adquieren de manera individual y se acoplan al entorno de trabajo de Protégé para añadirle funcionalidad. Existen varios conectores disponibles para importar ontologías en diferentes formatos, incluyendo DAG-EDIT, XML, RDF y OWL. Las herramientas PROMPT, son conectores para Protégé que permiten a los desarrolladores integrar ontologías, trazar los cambios en las ontologías a través del tiempo y crear vistas de las mismas.

Protégé está disponible de manera gratuita, junto con los conectores y algunas ontologías, en su sitio Web (http://protege.stanford.edu/).

4.4.2. Ontolingua

Ontolingua es una herramienta de desarrollo para navegar, crear, editar, modificar, verificar, evaluar y usar ontologías. Contiene una librería de ontologías cuyas definiciones, axiomas y términos no-lógicos, pueden ser reutilizadas en la construcción de nuevas ontologías.

Ontolingua basa la construcción de ontologías en el principio de diseño modular. Esto permite que las ontologías de las librerías puedan ser reutilizadas de cuatro diferentes maneras:

• **Inclusión**: Una ontología A es explícitamente incluida en una ontología B.
• **Polimorfismo**: Una definición de una ontología es incluida en otra y refinada.
• **Restricción**: Una versión restringida de una ontología es incluida en otra
• **Inclusión de Ciclos**: Situaciones como la siguiente se pueden dar, más no son recomendables: la ontología A se incluye en la B, la ontología B se incluye en la C y la ontología C se incluye en la A.

4.4.3. Chimaera

Chimaera es una herramienta que permite crear y mantener ontologías en la web, proporciona un ambiente distribuido para navegar, crear, editar, modificar y usar ontologías. Entre las facilidades que ofrece la herramienta se tienen: cargar bases de conocimiento en diferentes formatos, reorganizar taxonomías, resolver conflictos de nombres y editar términos. Destaca la capacidad para cargar datos de entrada en 15 diferentes formatos, tales como, KIF, Ontolingua, OKBC, Protegé, etcétera.

4.4.4. Selección del editor de ontologías

El editor de ontologías seleccionado para el desarrollo del presente trabajo es Protégé ya que es una herramienta que resulta acorde con el planteo metodológico. Permite una comparación de la aplicación durante todo el proceso, ya que existen una gran cantidad de ejemplos de desarrollos de ontologías en otros dominios lo que facilita corroborar la correctitud de nuestra aplicación. Dentro de las ventajas que colaboraron con el criterio de selección es que dispone de un ambiente visual de diseño y registro de ontologías, orientado a frames y slots desarrollado en Java y que puede correr en forma independiente en un PC

4.5. METODOLOGÍA, LENGUAJE Y HERRAMIENTA SELECCIONADA

En virtud a lo expresado en párrafos anteriores, cuando se analizaron los lenguajes para representar y describir ontologías, las herramientas para su implementación y el proceso metodológico a seguir, es que se define para la realización de este trabajo, la utilización de Protégé como editor de ontologías utilizando al lenguaje RDF, y siguiendo los pasos

metodológicos propuestos por la guía de pasos “Ontology Development 101” propuesta en [Noy 2005].

CAPÍTULO 5

RESUMEN *Este capítulo, parte de una revisión histórica de la evolución de la metodología seleccionada RUP/UML (Rational Unified Process), se describen sus características principales y estructura del proceso describiendo a sus componentes. Se presenta y se selecciona como herramienta de modelado a Rational, ya que es la que incluye la descripción del contexto a través de la definición de las reglas del negocio, describiendo su funcionamiento.*

5.1. RATIONAL UNIFIED PROCESS (RUP)

5.1.1. Historia y evolución

La Figura 5.1. ilustra la historia de RUP. El antecedente más importante se ubica en 1967 con la Metodología Ericsson (Ericsson Approach) elaborada por Ivar Jacobson, una aproximación de desarrollo basada en componentes, que introdujo el concepto de Caso de Uso. Entre los años de 1987 a 1995 Jacobson fundó la compañía Objectory AB y lanza el proceso de desarrollo Objectory (abreviación de Object Factory).

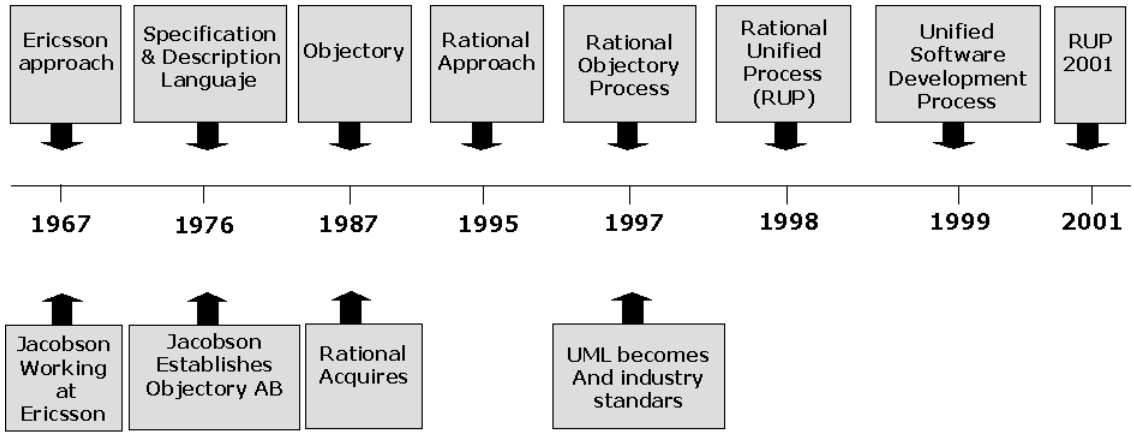


Figura 5.1. Historia de RUP

Posteriormente en 1995 Rational Software Corporation adquiere Objectory AB y entre 1995 y 1997 se desarrolla Rational Objectory Process (ROP) a partir de Objectory 3.8 y del Enfoque Rational (Rational Approach) adoptando UML como lenguaje de modelado.

Desde ese entonces y al comando de Grady Booch, Ivar Jacobson y James Rumbaugh, Rational Software desarrolló e incorporó diversos elementos para expandir ROP, destacándose especialmente el flujo de trabajo conocido como modelado del negocio. En junio del 1998 se lanza Rational Unified Process.

5.1.2. Características esenciales del proceso

El Proceso Unificado es un proceso de desarrollo de software, el cual se describe a través de un conjunto de actividades necesarias para transformar los requisitos de un usuario en un sistema de software. Se constituye de esta manera en un marco de trabajo genérico que puede especializarse para una gran variedad de sistemas de software, para diferentes áreas de aplicación, diferentes tipos de organizaciones, diferentes niveles de aptitud y diferentes tamaños de proyecto.

El Proceso Unificado utiliza el Lenguaje Unificado de Modelado UML (Unifield Modeling Languaje), para preparar todos los esquemas de representación conceptual de un sistema de software. De hecho, UML es una parte esencial del Proceso Unificado, siendo sus desarrollos realizados en forma paralela.

Los aspectos definitorios del Proceso Unificado es que es dirigido por casos de uso, centrado en la arquitectura, e iterativo e incremental.

5.1.2.1. Proceso dirigido por casos de uso

Partiendo de las definiciones de Kruchten, en donde presenta a los Casos de Uso como una técnica de captura de requisitos que fuerza a pensar en términos de importancia para el usuario y no sólo en términos de funciones que sería bueno contemplar [Kruchten 2000]. Se define un Caso de Uso como un fragmento de funcionalidad del sistema que proporciona al usuario un valor añadido. Los Casos de Uso representan los requisitos funcionales del sistema.

En RUP los Casos de Uso no son sólo una herramienta para especificar los requisitos del sistema. También guían su diseño, implementación y prueba. Los Casos de Uso constituyen un elemento integrador y una guía del trabajo.

En esta línea, los Casos de Uso no sólo inician el proceso de desarrollo sino que proporcionan un hilo conductor, permitiendo establecer trazabilidad entre los artefactos que son generados en las diferentes actividades del proceso de desarrollo (o también llamados workflows) [Booch 2006]. Dirigido por casos de uso quiere decir que el proceso de desarrollo sigue un hilo: avanza a través de una serie de flujos de trabajo que parten de los casos de uso.

Como se muestra en la Figura 5.2, basándose en los Casos de Uso se crean los modelos de análisis y diseño, luego la implementación que los lleva a cabo, y se verifica que efectivamente el producto implemente adecuadamente cada Caso de Uso. Todos los modelos deben estar sincronizados con el modelo de Casos de Uso.

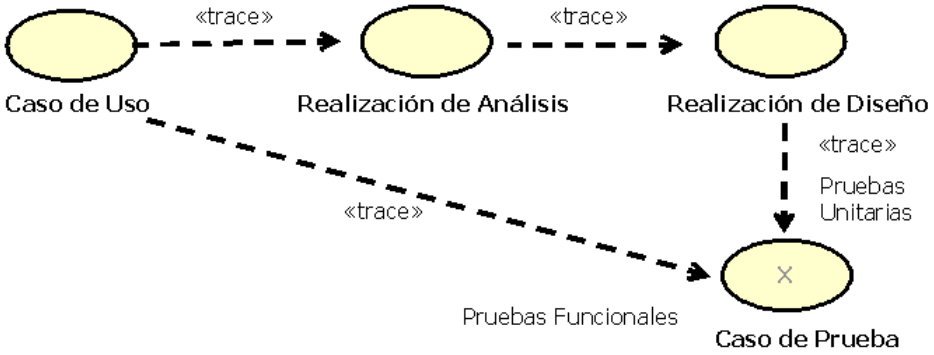


Figura 5.2.Trazabilidad a partir de los Casos de Uso

5.1.2.2. Proceso centrado en la arquitectura

La arquitectura de un sistema es la organización o estructura de sus partes más relevantes, lo que permite tener una visión común entre todos los involucrados (desarrolladores y usuarios) y una perspectiva clara del sistema completo, necesaria para controlar el desarrollo [Kruchten 2000].

El concepto de arquitectura de software incluye los aspectos estáticos y dinámicos más significativos del sistema. La arquitectura surge de las necesidades de la empresa, como las perciben los usuarios y se refleja en los casos de uso. Es influenciada por otros factores: plataforma en la que tiene que funcionar el software, los bloques de construcción reutilizables de que se dispone, consideraciones de implantación, sistemas heredados, y requisitos no funcionales.

Los arquitectos deben trabajar sobre la comprensión general de las funciones claves: los casos de uso claves del sistema.

Debe crearse un esquema en borrador de la arquitectura, comenzando por la parte que no es especifica de los casos de uso.

Es un proceso eminentemente evolutivo en donde básicamente se debe realizar:

- Trabajar con un subconjunto de los casos de uso especificados, con aquellos que representen las funciones claves del sistema en desarrollo. Cada caso de uso seleccionado se especifica en detalle y se realiza en términos de subsistemas, clases y componentes.
- A medida que los casos de uso se especifican y maduran, se descubre más de la arquitectura. Esto, a su vez, lleva a la maduración de más casos de uso.

Este proceso continúa hasta que se considere que la arquitectura es estable.

5.1.2.3. Diferentes vistas en el modelado

La arquitectura de un sistema puede ser descripta por diferentes vistas, como podemos ver en la Figura 5.3. [Booch 2006]

Figura 5.3. Modelado de la arquitectura de un sistema

La ***vista de casos de uso*** de un sistema comprende los casos de uso que describen el comportamiento del sistema tal y como es percibido por los usuarios finales, analistas, y encargados de prueba. Con UML, los aspectos estáticos de esta vista se capturan en los diagramas de casos de uso; los aspectos dinámicos de esta vista se capturan en los diagramas de interacción, de transición de estados y actividades.

La vista de ***vista de diseño*** de un sistema comprende las clases, interfaces y colaboraciones que forman el vocabulario del problema y su solución. Esta vista soporta principalmente los requisitos funcionales del sistema, entendiendo por ello los servicios que el sistema debería proporcionar a sus usuarios finales. Con UML, los aspectos estáticos de esta vista se capturan en los diagramas de clases y de objetos; y los aspectos dinámicos se capturan en los diagramas de interacción, diagramas de transición de estados y de actividades.

La ***vista de procesos*** de un sistema comprende los hilos y procesos que forman los mecanismos de sincronización y concurrencia del sistema. Esta vista cubre principalmente el funcionamiento, capacidad de crecimiento y rendimiento del sistema. Con UML, los aspectos estáticos y dinámicos de esta vista se capturan en el mismo tipo de diagramas que la vista de diseño, pero con énfasis en las clases activas que representan estos hilos y procesos.

La ***vista de implementación*** de un sistema comprende los componentes y archivos que se utilizan para ensamblar y hacer disponible.

5.1.2.4. Proceso iterativo e incremental

El equilibrio correcto entre los Casos de Uso y la arquitectura es algo muy parecido al equilibrio de la forma y la función en el desarrollo del producto, lo cual se consigue con el tiempo. [Booch 2006]. Para esto, la estrategia que se propone en RUP es tener un proceso iterativo e incremental en donde el trabajo se divide en partes más pequeñas o mini proyectos. Permitiendo que el equilibrio entre Casos de Uso y arquitectura se vaya logrando durante cada mini proyecto, así durante todo el proceso de desarrollo. Cada mini proyecto se puede ver

como una iteración (un recorrido más o menos completo a lo largo de todos los flujos de trabajo fundamentales) del cual se obtiene un incremento que produce un crecimiento en el producto.

Una iteración puede realizarse por medio de una cascada como se muestra en la Figura 5.4. Se pasa por los flujos fundamentales (Requisitos, Análisis, Diseño, Implementación y Pruebas), también existe una planificación de la iteración, un análisis de la iteración y algunas actividades específicas de la iteración. Al finalizar se realiza una integración de los resultados con lo obtenido de las iteraciones anteriores.

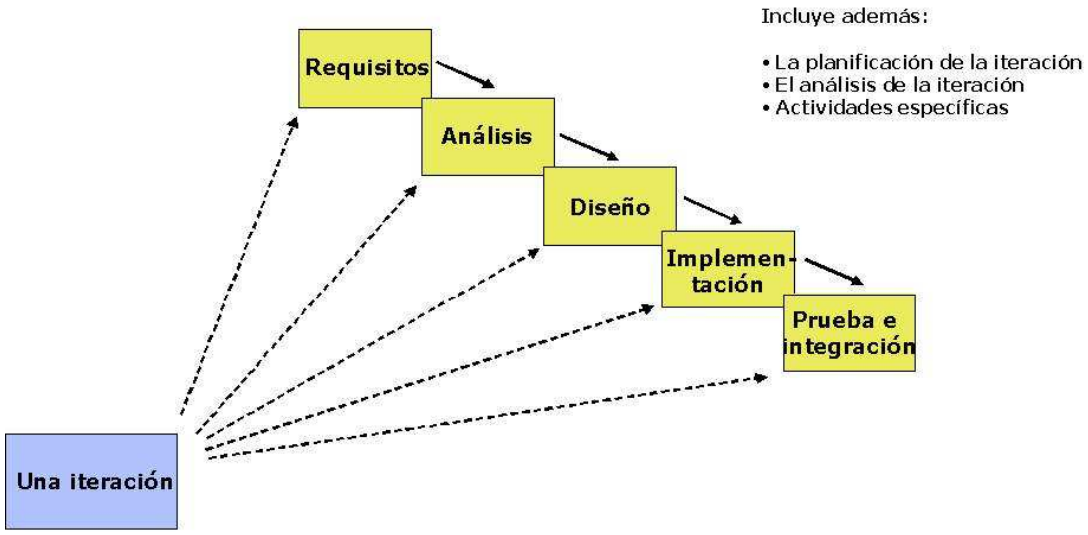


Figura 5.4.Una iteración en RUP

El proceso iterativo e incremental consta de una secuencia de iteraciones. Cada iteración aborda una parte de la funcionalidad total, pasando por todos los flujos de trabajo relevantes y refinando la arquitectura. En cada iteración, los desarrolladores identifican y especifican los casos de uso más relevantes, crean un diseño utilizando la arquitectura seleccionada como guía, implementan el diseño mediante componentes, y verifican que los componentes satisfagan los casos de uso. Si una iteración cumple con sus objetivos, el desarrollo continúa con la siguiente iteración. Cuando una iteración no cumple con sus objetivos, los desarrolladores deben revisar sus decisiones previas y probar con un nuevo enfoque.

La arquitectura proporciona la estructura sobre la cual guiar las iteraciones, mientras que los casos de uso definen los objetivos y dirigen el trabajo de cada iteración. La eliminación de una de las tres ideas reduciría drásticamente el valor del Proceso Unificado.

5.1.3. Fases dentro de un ciclo de vida

Cada ciclo se desarrolla a lo largo del tiempo. Cada fase termina con un hito, que determina la disponibilidad de un conjunto de artefactos; es decir, ciertos modelos o documentos han sido desarrollados hasta alcanzar un estado determinado.

Los hitos también permiten a la dirección, y a los mismos desarrolladores, controlar el progreso del trabajo según pasa por esos cuatro puntos clave.

La figura 5.5 muestra en la columna izquierda los flujos de trabajo (requisitos, análisis, diseño, implementación y prueba). Las curvas son una aproximación (no deben considerarse muy literalmente) de hasta dónde se llevan a cabo los flujos de trabajo en cada fase. Una iteración típica pasa por los cinco flujos de trabajo (o workflow), como se muestra en la siguiente figura:

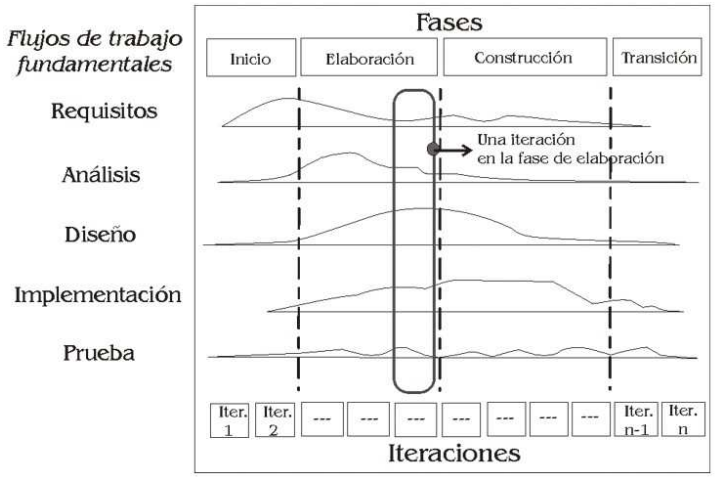


Figura 5.5. Esfuerzo en actividades según fase del proyecto

A continuación se procede a describir cada una de las fases: [Booch 2006]

- Durante la ***fase de inicio***, se desarrolla una descripción del producto final a partir de una buena idea y se presenta el análisis del negocio para el producto.
- En la ***fase de elaboración***, se especifica en detalle la mayoría de los casos de uso del producto y se diseña la arquitectura del sistema. La relación entre la arquitectura del sistema y el propio sistema es primordial. La arquitectura se expresa en forma de vistas de todos los modelos del sistema, los cuales juntos representan al sistema entero. Esto implica que hay varias vistas arquitectónicas del modelo de casos de uso, del modelo de análisis, del modelo de diseño, del modelo de implementación y modelo de despliegue. El resultado de esta fase es una línea base de la arquitectura.
- Durante la ***fase de construcción*** se crea el producto. Al final de esta fase, el producto contiene todos los casos de uso que la dirección y el cliente han acordado

para el desarrollo de esta versión. Sin embargo, puede que no esté completamente libre de defectos; muchos de estos se descubrirán y corregirán en la fase de transición.

- La ***fase de transición*** cubre el periodo durante el cual el producto se convierte en versión beta. En la versión beta un número reducido de usuarios con experiencia prueba el producto e informa defectos y deficiencias. La fase de transición conlleva actividades como la fabricación, formación del cliente, el proporcionar una línea de ayuda y asistencia, y la corrección de los defectos que se encuentren tras la entrega.

5.1.4. Pilares del proceso unificado – 4P

El resultado final del proyecto software es un producto que toma la forma durante su desarrollo gracias a la intervención de muchos tipos de personas, como se muestra en la Figura 5.6.

Los términos personas, proyecto, producto, proceso y herramientas serán definidos a continuación:

- **Personas:** los principales autores de un proyecto software son los arquitectos, desarrolladores, ingenieros de prueba, y el personal de gestión que les da soporte, además de los usuarios, clientes, etc. Las personas son realmente seres humanos a diferencia del término abstracto trabajadores (que mas adelante será definido).

- **Proyecto:** elemento organizativo a través del cual se gestiona el desarrollo de software. El resultado de un proyecto es una versión del producto.

- **Producto:** artefactos que se crean durante la vida del proyecto, como los modelos, código fuente, ejecutables, y documentación.

- **Proceso:** es una definición del conjunto completo de actividades necesarias para transformar los requisitos del usuario en un producto. Un proceso es una plantilla para crear proyectos.

- **Herramientas:** software que se utiliza para automatizar las actividades definidas en el proceso.

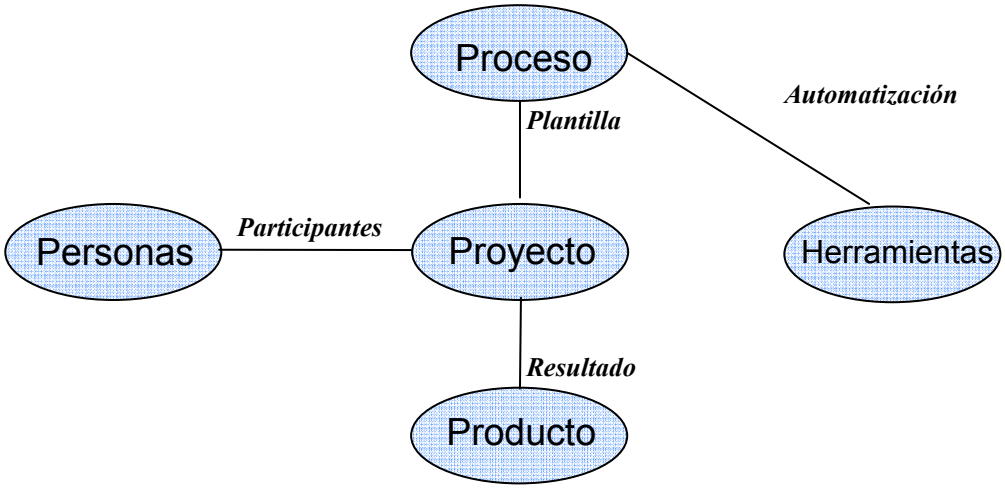


Figura 5.6. Las cuarto "P" en el desarrollo de software

5.1.4.1. Los procesos de desarrollo influyen sobre las personas

El modo en que se organiza y gestiona un proyecto software afecta a las personas implicadas en él. Conceptos como viabilidad, la gestión de riesgo, la organización de los equipos, la planificación del proyecto y la facilidad de comprensión del proyecto tienen un papel importante:

- Viabilidad del proyecto: la mayoría de la gente no disfruta trabajar en un proyecto que parece imposible de realizar. Los proyectos que no son viables suelen terminar en una fase temprana, aliviando así problemas de moral.
- Gestión del riesgo: de igual forma, cuando la gente siente que los riesgos no han sido analizados y reducidos, se sienten incómodos. La exploración en las primeras etapas atenúa este problema.
- Estructura de los equipos: es conveniente que los grupos de trabajo sean pequeños, ya que se trabaja de una forma más cómoda.
- Planificación del proyecto: las técnicas que se utilizan en las fases de inicio y de elaboración permiten a los desarrolladores tener una buena noción de cuál debería ser el resultado del proyecto.
- Facilidad de comprensión del proyecto: a la gente le gusta saber qué es lo que se está haciendo; quieren tener una comprensión global. La descripción de la arquitectura proporciona una visión general para cualquiera que se encuentre implicado en el proyecto.

Debido a que las personas son las que ejecutan las actividades clave del desarrollo de software, es necesario un proceso de desarrollo que esté soportado por herramientas y un Lenguaje Unificado de Modelado, de esta manera sus acciones resultan más eficaces. Este proceso permitirá a los desarrolladores construir un mejor software en términos de tiempo de

salida al mercado, calidad y costes. Les permitirá especificar los requisitos que mejor se ajusten a las necesidades de los usuarios; como así también elegir una arquitectura que permita construir los sistemas de forma económica y puntual.

La gente llega a ocupar muchos puestos diferentes en una organización de desarrollo de software. Se llama trabajador a los puestos a los cuales pueden asignar a personas, y los cuales esas personas aceptan; es definitiva es un papel que un individuo puede desempeñar en el desarrollo de software. También es importante dar la definición de rol. Un rol es un papel que cumple un trabajador. Un trabajador es responsable de un conjunto completo de actividades. Al mismo tiempo, si tiene que hacer su trabajo, las herramientas que emplean deben ser las adecuadas. Las herramientas no sólo deben ayudar a los trabajadores a llevar a cabo sus propias actividades, sino también deben aislarles de la información que no le es relevante. Para lograr estos objetivos, el Proceso Unificado describe formalmente los puestos (es decir, los trabajadores), que las personas pueden asumir en el proceso. Un trabajador también puede representar a un conjunto de personas que trabajan juntas. Cada trabajador tiene un conjunto de responsabilidades y lleva a cabo un conjunto de actividades en el desarrollo de software.

5.1.4.2. El producto es más que código

Un proyecto de desarrollo da como resultado una nueva versión de un producto. En el contexto del Proceso Unificado, el producto que se obtiene es un sistema de software.

Un sistema es todos los artefactos que se necesitan para representarlos en una forma comprensible por máquinas u hombres, para las máquinas (herramientas, compiladores, etc.), los trabajadores (directores, arquitectos, desarrolladores) y los interesados (inversores, usuarios, jefes de proyecto).

Un artefacto es un término general para cualquier tipo de información creada, producida, cambiada o utilizada por los trabajadores en el desarrollo del sistema.

El tipo de artefacto más utilizado en el Proceso Unificado es el modelo. La construcción de un sistema es un proceso de construcción de modelos, utilizando distintos modelos para describir todas las perspectivas diferentes del sistema.

Un modelo es una abstracción del sistema, especificando el sistema modelado desde cierto punto de vista y en un determinado nivel de abstracción. Un punto de vista es, por ejemplo, una vista de especificación o una vista de diseño del sistema.

5.1.4.3. El proceso: una plantilla

En el Proceso Unificado, proceso hace referencia a un contexto que sirve como plantilla que puede reutilizarse para crear instancias de ella. Es comparable a una clase, que

puede utilizarse para crear objetos en el paradigma orientado a objetos. Instancia del proceso es un sinónimo de proyecto.

Un proceso de desarrollo de software es un conjunto completo de actividades necesarias para convertir los requisitos de usuario en un conjunto consistente de artefactos que conforman un producto de software, y para convertir los cambios sobre esos requisitos en un nuevo conjunto consistente de artefactos. Un proceso es una definición de un conjunto de actividades, no su ejecución.

5.1.4.4. Las herramientas son esenciales en el proceso

Las herramientas soportan los procesos de desarrollo de software modernos.

El proceso se ve fuertemente influido por las herramientas. Las herramientas son buenas para automatizar procesos repetitivos, mantener las cosas estructuradas, gestionar grandes cantidades de información y para guiarnos a lo largo de un camino de desarrollo concreto.

Las herramientas se desarrollan para automatizar actividades, de manera completa o parcial, para incrementar la productividad y la calidad, y para reducir el tiempo de desarrollo.

5.1.5. Bloques de construcción de UML

El Proceso Unificado utiliza el Lenguaje Unificado de Modelado (UML) para preparar todos los esquemas de un sistema de software. De hecho, UML es una parte esencial del Proceso Unificado (sus desarrollos fueron paralelos). En la figura 5.7 se muestra el bloque de construcción de UML:

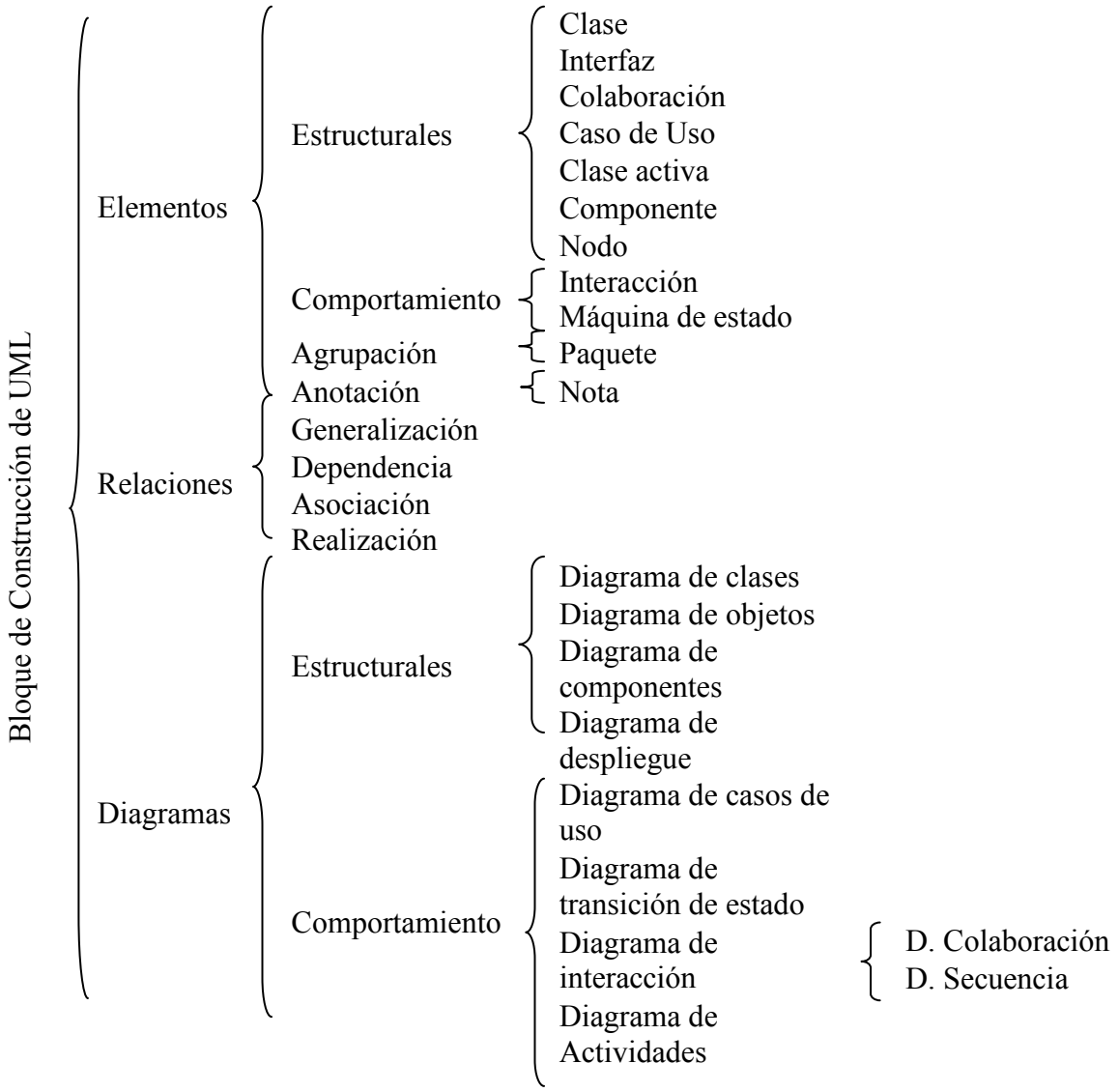


Figura 5.7. Bloques de Construcción de UML

5.1.5.1. Elementos estructurales

Son los nombres de los modelos de UML. En su mayoría son las partes estáticas de un modelo, y representan cosas que son conceptuales o materiales.

- **Clase**: Es una descripción de un conjunto de objetos que comparten los mismos atributos, operaciones, relaciones y semántica. Una clase implementa una o más interfaces.
- **Interface**: Es una colección de operaciones que especifican un servicio de una clase o componente. Describe un comportamiento visible externamente de ese elemento. Puede representar el comportamiento completo de una clase o componente o sólo una parte de ese comportamiento.

- **Colaboración**: Define una interacción y es una sociedad de roles y otros elementos que colaboran para proporcionar un comportamiento cooperativo mayor que la suma de los comportamientos de sus elementos. Por lo tanto, las colaboraciones tienen una dimensión tanto estructural como de comportamiento. Una clase dada puede participar en varias colaboraciones. Estas colaboraciones representan la implementación de patrones que forman un sistema.
- **Clase activa**: Es una clase cuyos objetos tienen uno o más procesos o hilos de ejecución y por lo tanto pueden dar origen a actividades de control. Una clase activa es igual que una clase, excepto en que sus objetos representan elementos cuyo comportamiento es concurrente con otros elementos.
- **Componente**: Es una parte física y reemplazable de un sistema que conforma un conjunto de interfaces y proporciona la implementación de dicho conjunto. En un sistema, se podrán encontrar diferentes tipos de componentes de despliegue (tales como componentes de Java), así como componentes que sean artefactos del proceso de desarrollo (tales como archivos de código fuente). Un componente representa típicamente el empaquetamiento físico de diferentes elementos lógicos, como clases, interfaces y colaboraciones.
- **Nodo**: Es un elemento físico que existe en tiempo de ejecución y representa un recurso computacional, que por lo general dispone de algo de memoria y con frecuencia, capacidad de procesamiento. Un conjunto de componentes puede residir en un nodo, y puede migrar de nodo a otro.

5.1.5.2. Elementos de comportamiento

Son las partes dinámicas de los modelos de UML. Éstos son los verbos de un modelo, y representan comportamiento en el tiempo y espacio.

- **Interacción**: Es un comportamiento que comprende un conjunto de mensajes intercambiados entre un conjunto de objetos, dentro de un contexto particular, para alcanzar un propósito específico. El comportamiento de una sociedad de objetos o una operación individual puede especificarse con una interacción. Una interacción involucra muchos otros elementos. Mensajes, secuencias de acción (el comportamiento involucrado por un mensaje) y enlaces (conexiones entre objetos).
- **Máquina de estados**: Es un comportamiento que especifica las secuencias de estados por las que pasa un objeto o una interacción durante su vida en respuesta a eventos, junto con sus reacciones a estos eventos. El comportamiento de una clase individual o una colaboración de clases pueden especificarse con una máquina de estados. Una máquina de estados involucra a elementos: estados, transiciones (el

flujo de un estado a otro), eventos (que disparan una transición) y actividades (la repuesta a una transición).

5.1.5.3. Elementos de agrupación

Los elementos de agrupación son las partes organizativas de los modelos de UML. Estos son las cajas en las que puede descomponerse un modelo.

- **Paquete**: Es un mecanismo de propósito general para organizar elementos en grupos. Los elementos estructurales, e incluso otros elementos de agrupación pueden incluirse en un paquete.

5.1.5.4. Elementos de anotación

Los elementos de anotación son las partes explicativas de los modelos UML. Son comentarios que se pueden aplicar para describir, clarificar y hacer observaciones sobre cualquier elemento de un modelo.

- **Nota**: Es simplemente un símbolo para mostrar restricciones y comentarios junto a un elemento o una colección de elementos.

5.1.5.5. Relaciones

- **Dependencia:** Es una relación semántica entre dos elementos, en la cual un cambio a un elemento (el elemento independiente) puede afectar a la semántica del otro elemento (elemento dependiente).
- **Asociación:** Es una relación estructural que describe un conjunto de enlaces, los cuales son conexiones entre objetos. La agregación es un tipo especial de asociación, que representa una relación estructural entre un todo y sus partes.
- **Generalización:** Es una relación de especialización/generalización en la cual los objetos del elemento especializado (el hijo) pueden sustituir a los objetos del elemento general (el padre). De esta forma, el hijo comparte la estructura y el comportamiento del padre.
- **Realización:** Es una relación semántica entre clasificadores, en donde un clasificador especifica un contrato que otro clasificador garantiza que cumplirá. Se pueden encontrar relaciones de realización en dos sitios: entre interfaces y clases y componentes que las realizan, y entre los casos de uso y las colaboraciones que los realizan.

5.1.5.6. Diagramas

Un diagrama es una representación gráfica de un conjunto de elementos, visualizado la mayoría de las veces como un grafo conexo de nodos (elementos) y arcos (relaciones).

- **Diagrama de clases**: Muestra un conjunto de clases, interfaces y colaboraciones, así como sus relaciones. Estos diagramas son los diagramas más comunes en el modelado de sistemas orientados a objetos. Los diagramas de clases cubren la vista estática de un sistema. Incluyen clases activas que cubren la vista de procesos estática de un sistema.
- **Diagrama de objetos**: Muestran un conjunto de objetos y sus relaciones. Representan instantáneas de instancias de los elementos encontrados en los diagramas de clases. Cubren la vista de diseño estática, pero desde la perspectiva de casos reales.
- **Diagrama de casos de uso**: Muestra un conjunto de casos de uso y actores (un tipo especial de clases) y sus relaciones. Cubren la vista estática de un sistema. Son especialmente importantes en el modelado y organización del comportamiento del sistema.
- **Diagrama de interacción**: Muestra una interacción, que consta de un conjunto de objetos y sus relaciones, incluyendo los mensajes que pueden ser enviados entre ellos. Cubren la vista dinámica de un sistema, pueden ser de dos tipos:
    - **Diagrama de secuencia**: Es un diagrama de interacción que resalta la ordenación temporal de los mensajes.
    - **Diagrama de colaboración**: Es un diagrama de interacción que resalta la organización estructural de los objetos que envían y reciben mensajes.

  Entonces, los diagramas de secuencia y los de colaboración podemos decir que son isomorfos, porque se pueden tomar uno y transformarlo en otro.
- **Diagrama de transición de estados**: Muestra una máquina de estados, que consta de estados, transiciones, eventos y actividades Cubren la vista dinámica de un sistema. Son importantes en el modelado del comportamiento de una interfaz, una clase o una colaboración.
- **Diagrama de actividades**: Es un tipo especial de diagrama de estados que muestra el flujo de actividades dentro de un sistema. Cubren la vista dinámica de un sistema. Sirven para modelar el funcionamiento de un sistema y resaltan el flujo de control entre objetos.
- **Diagrama de componentes**: Muestra la organización y dependencias entre un conjunto de componentes. Cubren la vista de implementación estática de un sistema. Se relacionan con los diagramas de clases en que un componente se corresponde, por lo general, con una o más clases, interfaces o colaboraciones.

- **Diagrama de despliegue**: Muestra la configuración de nodos de procesamiento en tiempo de ejecución y los componentes que residen en ellos. Cubren la vista de despliegue estática de una arquitectura. Se relacionan con los diagramas de componentes en que un nodo incluye, por lo general, uno o más componentes.

5.2. HERRAMIENTA SOPORTE DE MODELADO: RATIONAL ROSE

La presente sección ha sido construida a través de consultas en el sitio de soporte de la herramienta [Rational 2009] y presenta un resumen de las características esenciales.

5.2.1. Descripción

Rational Rose es una herramienta de modelado de software, que propone la utilización de los diferentes tipos de modelos para realizar un diseño de sistemas, utilizando una vista estática y otra dinámica de los modelos del sistema, uno lógico y otro físico. Permite crear y refinar estas vistas creando de esta manera un modelo completo que representa el dominio del problema y el sistema de software.

Soporta el enfoque y perspectivas de UML, posibilitando la especificación de requisitos mediante casos de uso, análisis y diseño orientado a objetos pudiendo utilizar los diagramas para visualizar el sistema desde diferentes perspectivas.

Estos diagramas pueden tener cualquier combinación de elementos y relaciones. Hay que tener en cuenta que en la práctica solo surge un pequeño número de combinaciones, las cuales son consistentes con las vistas más útiles que comprenden la arquitectura de un sistema con gran cantidad de software.

Es una herramienta visual que permite realizar análisis orientado a objetos, modelización, diseño y construcción de aplicaciones. Permitiendo además que el equipo de desarrollo pueda comunicar efectivamente la arquitectura del software por medio de una representación gráfica, documentar y generar el código de la aplicación.

Rational Rose incluye, entre otros, los siguientes diagramas:

- Diagrama de Clases
- Diagrama de Objetos
- Diagrama de Casos de Usos
- Diagrama de Secuencia
- Diagrama de Colaboración
- Diagrama de Estados
- Diagrama de Actividades

- Diagrama de Componentes
- Diagrama de Despliegue

5.2.2. Beneficios esperados de la herramienta

Al permitir realizar un modelado visual, se espera una ventaja competitiva cuyos beneficios se pueden resumir en los siguientes:

- Ciclos de desarrollos cortos por medio de un desarrollo interactivo controlado.
- Incremento de la productividad por medio del desarrollo "conducido por el modelo" y eliminación del riesgo.
- Soporte para proyectos de gran tamaño y organizaciones con múltiples proyectos por medio de un ambiente escalable.
- Significante reutilización del software a través de software basado en arquitectura y componentes.
- Mejora la comunicación del equipo de trabajo por medio del lenguaje de para modelar.
- Integración con sistemas heredados por medio de las capacidades de ingeniería reversa y extensibilidad de interfaces.

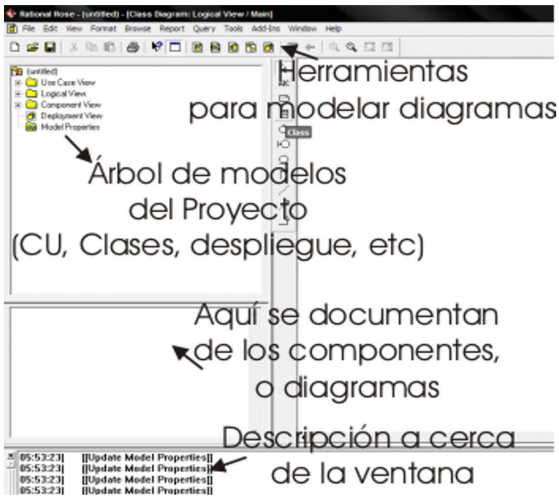
5.2.3. Facilidad de aprendizaje y forma de utilización

Describiremos ahora las características principales de la herramienta utilizada. En este proyecto se ha utilizado el Rational Enterprise Edition 2003.

Es una herramienta de modelado resulta bastante sencilla de utilizar, pero el requerimiento básico es que el usuario tenga nociones sobre Lenguaje Unificado de Modelado UML, ya que todos las vistas, diagramas y modelos que se pueden desarrollar utilizan esta notación.

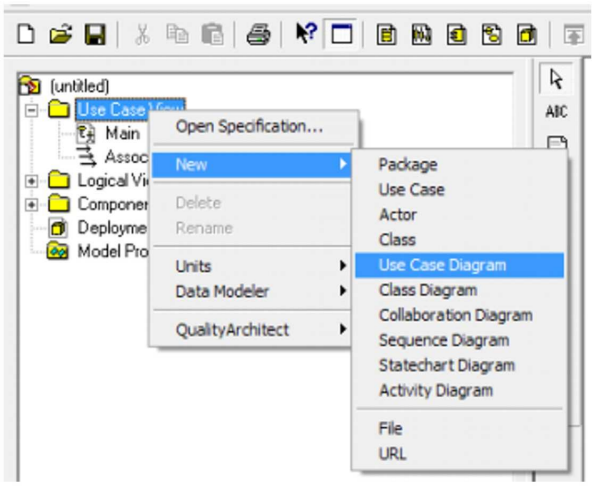
En la primera pantalla (Pantalla 5.1) se detallan las diferentes partes a las que luego haremos referencia:

Para crear por ejemplo un diagrama de casos de uso, tenemos que hacer click en el árbol de nuestro proyecto, en la carpeta "Use Case View", hacer click izquierdo ir a New→Use Case Diagram:



Pantalla 5.1. Pantalla inicial de Rational Rose

Ahora, para agregar actores al diagrama hay que arrastrarlos a la pantalla principal como se ve en la Pantalla 5.2; se procede de la misma manera para agregar casos de uso a nuestro modelo.



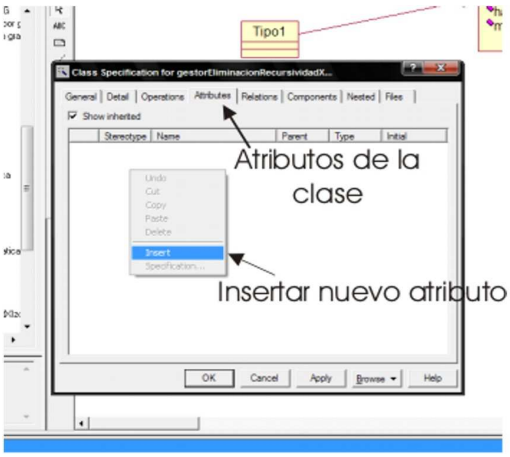
Pantalla 5.2. Agregado de actores en Rational Rose

Dentro del modelo de Casos de uso se pueden realizar varios diagramas de casos de uso, normalmente se los divide en diagrama de casos de uso esenciales, y Diagrama de casos de uso de soporte, para tener una visión más clara.

Para crear un diagrama de clases se procede prácticamente de la misma manera que en el caso anterior: nos posicionamos con el cursor sobre el árbol del proyecto, en la carpeta "Logical View", hacemos click en New→Class Diagram.

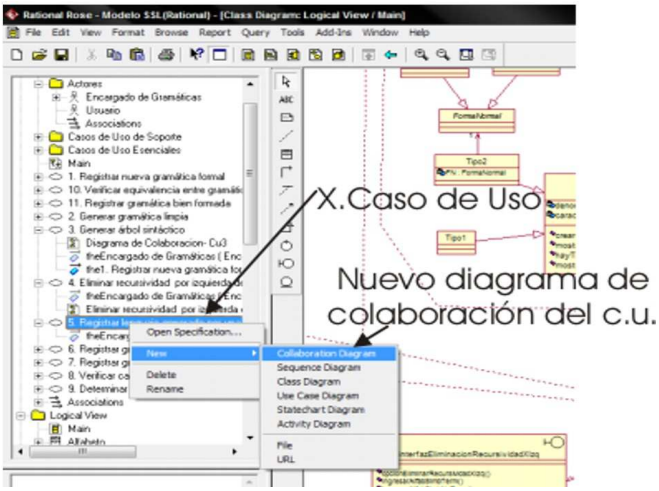
Para agregar clases simplemente se las arrastra a la pantalla principal, se pueden también especificar las relaciones entre estas clases, seleccionando previamente el tipo de relación y luego indicando de qué clase a otra está relacionada.

Para agregar un atributo a una clase, como se muestra en la pantalla 5.3, hay que hacer click derecho sobre la misma y abrir las especificaciones (Open Specification), ir a la pestaña de atributo, hacer click derecho e insertar el nuevo atributo.



Pantalla 5.3. Agregado de atributo en Rational Rose

Para crear un diagrama de colaboración, nos posicionamos con el cursor sobre el árbol del proyecto, en el Modelo de casos de uso, sobre el caso de uso al cual deseamos realizar el diagrama de colaboración. Hacemos click derecho New→Collaboration Diagram como se muestra en la Pantalla 5.4:



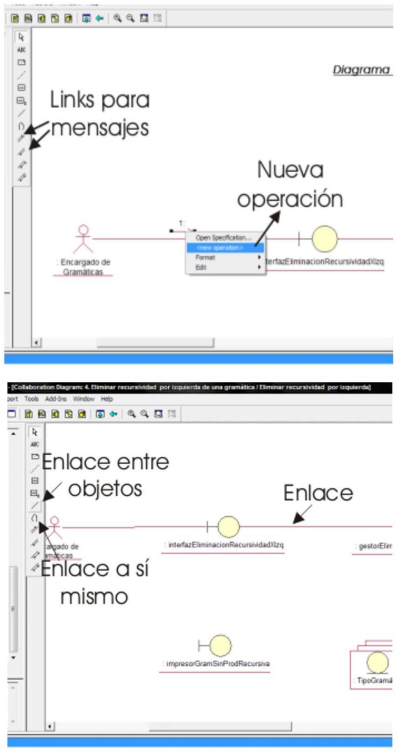
Pantalla 5.4. Agregado de componentes en Rational Rose

Para agregarle componentes al diagrama de colaboración, previamente tienen que estar creadas las clases intervinientes (en el Modelo de Clases) y los actores (definidos en el Modelo de Casos de Uso), e irlos arrastrándolos a la pantalla principal. Para relacionar los diferentes objetos utilizaremos los "enlaces". Para poderlos incluir en nuestro diagrama de colaboración hay que hacer click en la barra de herramientas del diagrama sobre la línea continua, y arrastrar con el mouse desde qué objeto se relaciona con otro. A continuación se muestran los enlaces que se pueden realizar:

Ahora bien, para insertar los mensajes que se van a enviar los objetos entre sí, hay que ir a la barra de herramientas del diagrama de colaboración y hacer click en alguno de los tipos de links (ver Pantalla 5.5), y luego hacer click sobre el enlace. Luego, hacemos click derecho, y seleccionamos "new operation":

Es posible seleccionar alguna otra operación (método de la clase), que hayamos creado anteriormente. Cabe destacar que Rational Rose crea automáticamente cada nueva operación de este diagrama en el diagrama de clases, para que haya una consistencia entre los diferentes modelos, además de ahorrarnos trabajo.

Si deseamos crear un diagrama de secuencia teniendo ya un diagrama de colaboración, es muy sencillo lo que hay que hacer. Nos posicionamos en el árbol de nuestro proyecto, en el Modelo de Casos de Uso, sobre el caso de uso al cual le queremos crear el diagrama de secuencia, abrimos el diagrama de colaboración y presionamos F5.



Pantalla 5.5. Mensajes entre objetos en Rational Rose

CAPÍTULO 6

RESUMEN *En este capítulo se presentan los aspectos metodológicos seguidos en el proceso de comparación de las diferentes metodologías/herramientas bajo estudio en el presenta trabajo. Se enuncian y analizan las particularidades de los dominios seleccionados. Se describe la metodología seleccionada para establecer la comparación de las diferentes metodologías/herramientas es el proceso de análisis Jerárquico (AHP). Se visualizan los aspectos de los criterios de evaluación seleccionados y la operatoria seguida para la asignación de los mismos.*

6.1. DOMINIOS A MODELAR BAJO ESTUDIO

De manera de construir un marco de referencia para efectuar comparativas de las diferentes Herramientas/Metodologías objetos de estudio del presente trabajo, se han seleccionado dos dominios bastantes diferente entre si, los que permitirán apreciar debido a las particularidades de los mismos el grado de respuesta de cada Metodología y/o Herramienta

Los dominios seleccionados son: por un lado, un acotado y simple dominio que no presenta condicionantes en el proceso de modelado como ser un Sistemas de Docentes a cursos y por otro, un dominio de Gramáticas Formales y Máquinas Abstractas el cuál es fuertemente abstracto y cercano a las matemáticas.

6.1.1. Sistema a modelar: Docentes a cursos

Un Docente además de los datos que identifican a la persona (Nombre, DNI, Sexo, Edad) cuenta con un legajo docente, año de ingreso a la docencia y una categoría la cuál puede ser de Profesor o Auxiliar docente. Dentro de la Categoría profesor existen tres categorías Titular, Asociado o Adjunto. Dentro de la categoría de Auxiliares, puede ser JTP, Ayte Graduado o Ayudante Alumno. En el caso de que un alumno se desempeñe como Ayte alumno deberá formar parte del plantel con legajo docente, y deberá identificarse el legajo de alumno en el mismo.

Los cursos tienen un código de Identificación, nombre del curso, año de dictado, docente del curso y auxiliar del curso, y opcionalmente pueden tener hasta dos Aytes Alumnos. En el caso de auxiliares docentes del curso, puede ser un profesor o Ayudante Docente de las Categorías JTP o Ayte de 1ra. Los cursos pueden tener inscripto hasta 30 alumnos.

Un alumno además de los datos de identificación de la persona, debe poseer un Legajo Alumno, año de ingreso y cantidad de cursos aprobados.

Para la asignación de docentes y auxiliares a cursos, ningún docente de la categoría de profesor puede estar en más de tres cursos y los Ayudantes Alumnos pueden estar en solo un dentro de un mismo año

Ayte 1ra = Ayte Graduado

El sistema deberá permitir:

- Dar alta y permitir consultas y modificaciones sobre docentes, alumnos y cursos
- Listar docente por categorías y cantidad de cursos asignados en un determinado año.
- Listar cursos dictados en un año
- Listar detalles de los cursos (Código, Nombre, Docente, Auxiliar /Aytes Alumnos y cantidad de Alumnos).
- Listar alumnos inscriptos a cursos.

6.1.2. Sistema a modelar: Gramáticas Formales y Máquinas Abstractas

Se presenta a continuación el segundo dominio seleccionado para efectuar las comparaciones de las diferentes metodologías/herramientas en estudio, tal cual consta en la Figura 6.1. El dominio seleccionado son las Gramáticas Formales y Máquinas Abstractas y su particularidad radica en el isomorfismo que presenta entre sí y concon los Lenguajes Formales.

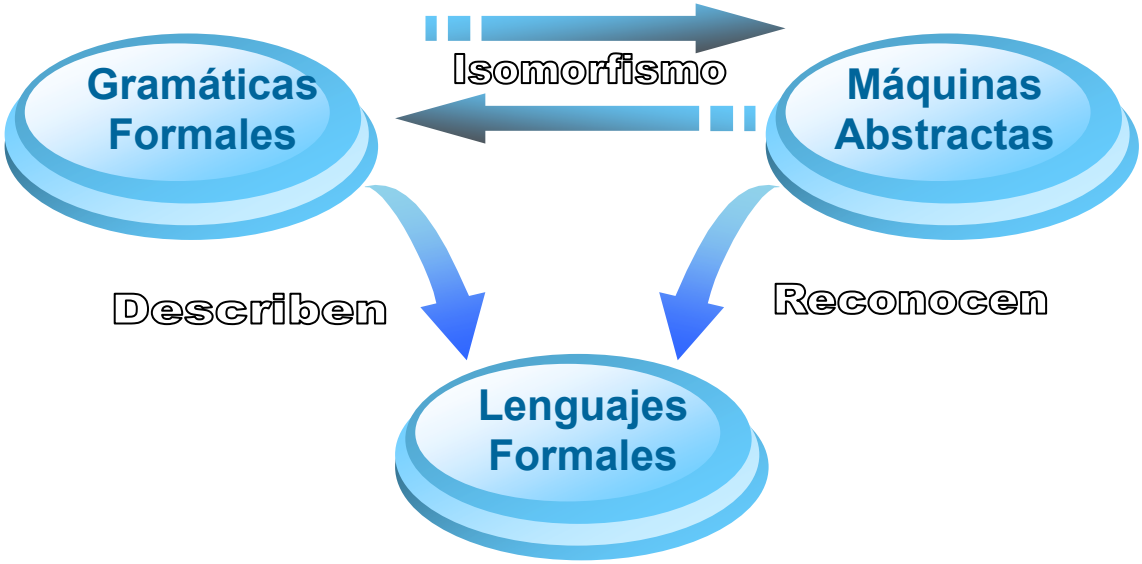


Figura 6.1. Isomorfismo entre gramáticas formales y máquinas abstractas

La elección de este dominio de aplicación tiene un doble propósito: por un lado nos permitirá comparar entre si las diferentes metodologías en donde se analizarán, ventajas y desventajas sobre las dimensiones de análisis determinadas. Por otro lado, es que dentro de cada uno de los Modelos adoptados en la comparación, se establecerá el grado de correspondencia entre la conceptualización de las Máquinas Abstractas y Gramáticas Formales, ya que al existir un isomorfismo entre ambos dominios, éste debería continuar en los esquemas conceptuales resultantes de la aplicación de cada una de las metodologías comparadas.

6.1.3. Características de los Sistemas a modelar

El Dominio de Gramáticas Formales y Máquinas Abstractas, por ser fuertemente abstracto y estar más cercano a las matemáticas, en una primera instancia, no permitiría apreciar y evaluar a las distintas cualidades de las metodologías/herramientas sin estar condicionada la evaluación a las particularidades del dominio de aplicación. Por este motivo, surge la necesidad de contar con otro dominio que no presente condicionantes en el proceso de modelado y que luego estas valoraciones obtenidas sean comparadas y contrastadas con las que se obtengan las mismas sobre el dominio de GF y MA.

6.2. CRITERIOS A EVALUAR

Los criterios seleccionados para efectuar la comparación de las diferentes metodologías/herramientas, han sido obtenidos a través de la investigación bibliográfica sobre propuestas de trabajos realizados en determinación de criterios para la comparación de Metodologías y Herramientas en la especificación de un modelo conceptual, y se encuentran descriptas en el Anexo I. Esta caracterización de criterios para la comparación de modelos conceptuales, se encuentra desarrollada en el Capítulo II del presente trabajo de tesis y provee una valoración de criterios que será aplicada a cada una de las Metodologías/Herramientas en estudio.

Cada uno de los criterios son valorados con una escala de (Malo – Regular – Bueno – Muy Bueno – Excelente), utilizando anotaciones marginales para establecer de acuerdo al criterio seleccionado cuales son las ponderaciones que influenciado en la calificación otorgada.

6.2.1. Metodologías

A continuación, se detallarán las diferentes dimensiones de análisis, junto con la agrupación de criterios de manera de facilitar el proceso de análisis jerárquico para la comparación y valoración de las metodologías objetos de estudio.

- **Facilidad de entendimiento, comprensión y validación de los modelos por parte del experto del dominio (usuario de la Aplicación).**
  Criterios incluidos:
  - Claridad Conceptual.
  - Potencialidad para abstraer esencia del dominio.
  - Identificación de la fuente de Requerimientos.
  - Facilidad de entendimiento con el usuario del Dominio.
  - Validación del modelo resultante.
  - Facilidad de trazabilidad de requerimientos.
- **Facilidad de aplicación de la Metodología y grado de representación del dominio a través del modelo obtenido.**
  Criterios incluidos:
  - Reducción de ambigüedades sobre conceptos y manejo de sinónimos
  - Facilidad de aplicación y flexibilidad para adopción de criterios de diseño
  - Mantenibilidad del Modelo
  - Reutilización del Modelo
- **Facilidad de comunicación entre analistas y desarrolladores aportados por la Metodología y soporte documental para todas las etapas en el proceso de desarrollo.**
  Criterios incluidos:
  - Documentación del modelo.
  - Jerarquización de los conceptos del modelo
  - Versionado en proceso Iterativo.
  - Producto como insumo para la construcción del sistema modelado.

6.2.2. Herramientas

A continuación, se detallarán las diferentes dimensiones de análisis, junto con los criterios agrupados para la comparación y valoración de las Herramientas de soporte objeto de estudio.

- **Facilidad de uso y aprendizaje.**
  Criterios incluidos:
  - Facilidad de instalación y configuración.
  - Curva de Aprendizaje de la Herramienta
  - Portabilidad
- **Prestaciones en la actividad de modelado**

Criterios incluidos:

- Capturar en forma fiel y precisa la abstracción del modelo.
- Facilidad de introducir cambios en el diseño.
- Soporte a proceso iterativo y manejo de versionado.
- Visualización a través de diferentes vistas del modelo.
- Uso de notaciones y simbología que faciliten el entendimiento del usuario.

- **Facilidad de comunicación con etapas posteriores de desarrollo**

  Criterios incluidos:

  - Uso de notaciones y simbología que faciliten el entendimiento del equipo de desarrollo en la construcción del modelo.
  - Facilidad de mapeo directo del modelo para la construcción del sistema

6.2.3. Metodología para la asignación de valores a Criterios de Evaluación

Las valoraciones asignadas a cada atributo de los criterios seleccionados sobre cada metodología y cada herramienta, han sido en todo momento, validadas y contrastadas dentro del equipo de trabajo, ya que es necesario hacer notar que el presente desarrollo de esta tesis está inscripto dentro y es parte integrante de un proyecto de investigación consolidado. Los integrantes del proyecto de investigación se han involucrado activamente en las actividades de modelado de cada dominio, sobre todas las metodologías y herramientas y han participado en las apreciaciones y valoraciones realizadas.

6.3. HERRAMIENTA MULTICRITERIO DE COMPARACIÓN

En esta sección se describirá el proceso metodológico seleccionado para efectuar la comparación entre las diferentes metodologías/herramientas.

6.3.1. Proceso de análisis jerárquico (AHP)

Este proceso fue desarrollado por Thomas L. Saaty [Saaty 1980], cuya metodología está descripta en [Hurtado 2007]. Este proceso está desarrollado para resolver problemas complejos con criterios múltiples, los cuales pueden incluir valoración de criterios cualitativos ponderados a través de una escala numérica. El AHP se basa en la construcción de un modelo jerárquico el cual permite de manera eficiente y gráfica organizar la información del

problema, de manera de obtener como resultado un ranking de prioridad de selección de las alternativas analizadas.

6.3.1.1 Ventajas de la aplicación del proceso de análisis jerárquico (AHP)

- Estructuración de un modelo jerárquico
- Priorización de los criterios
- Evaluación mediante la asignación de pesos
- Obtención de ranking de alternativas
- Desglose del problema por partes
- Permite medir criterios cualitativos
- Fácil aplicación, con sustento matemático en su definición

6.3.1.2. Pasos involucrados

- Construir el modelo jerárquico: básicamente está estructurado en tres niveles: Metas u objetivos, criterios y alternativas.
- Establecimiento de prioridades de preferencia de cada criterio en relación a la meta u objetivo: Se asigna un valor numérico, el cuál representará esta preferencia.
- Establecimiento de prioridades de preferencia de cada subcriterio en relación a cada criterio: Se asigna un valor numérico, el cuál representará esta preferencia.
- Construcción de las matrices para comparación pareadas para criterios en relación de la meta y subcriterios en relación a criterios.
- Síntesis: la misma provee el orden de prioridad de cada una de las alternativas, en relación a los criterios y subcriterios analizados. Este proceso se calcula a través de las matrices de prioridades, el cuál es un proceso matemático preciso que implica cálculo de valores y vectores característicos. Para efectuar estos cálculos nos apoyaremos en el software Expert Choice 11.5 diseñado para evaluación del proceso de análisis jerárquico (AHP)

6.3.2. Construcción del Modelo de Proceso de Análisis Jerárquico (AHP)

Para aplicar el proceso de Análisis jerárquico (AHP), es necesario determinar un orden de prevalencia entre los criterios involucrados

6.3.2.1. Modelo Jerárquico para Metodologías

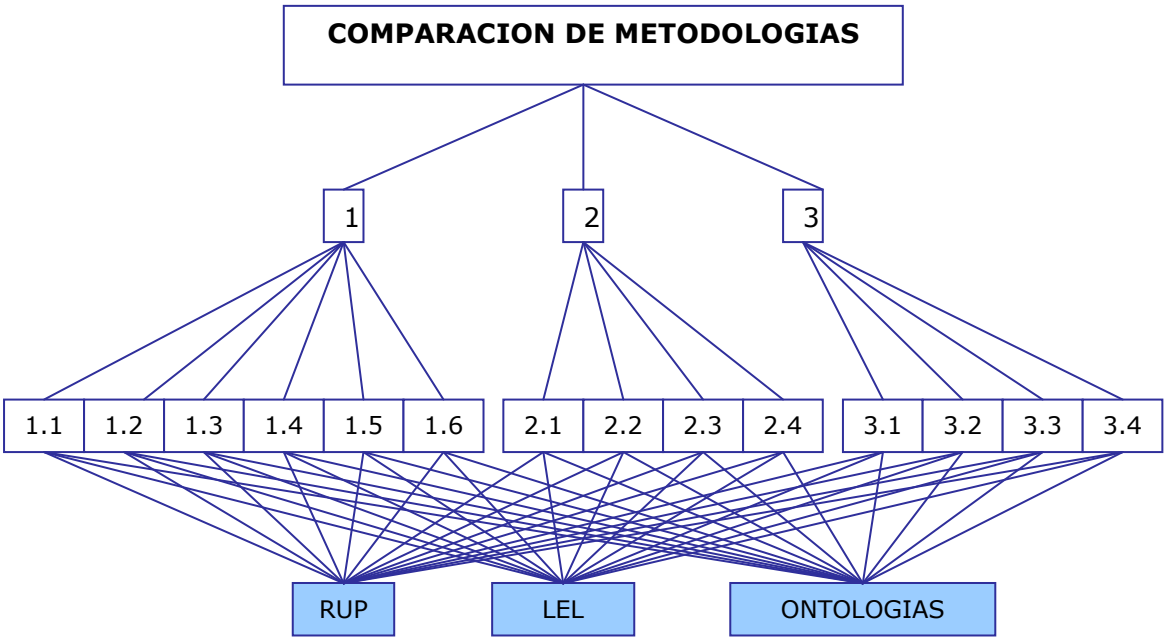


Figura 6.2. Modelo jerárquico AHP – comparación metodologías

Referencias:

Tabla 6.1. Descripción modelo jerárquico AHP en comparación metodologías

ID	DESCRIPCION
1	Facilidad de entendimiento, comprensión y validación de los modelos por parte del experto del dominio
2	Facilidad de aplicación de la Metodología y grado de representación del dominio a través del modelo obtenido
3	Facilidad de comunicación entre analistas y desarrolladores aportados por la Metodología y soporte documental para todas las etapas en el proceso de desarrollo
1.1	Claridad Conceptual
1.2	Potencialidad para abstraer esencia del dominio
1.3	Identificación de la fuente de Requerimientos
1.4	Facilidad de entendimiento con el usuario del Dominio
1.5	Validación del modelo resultante
1.6	Facilidad de trazabilidad de requerimientos
2.1	Reducción de ambigüedades sobre conceptos y manejo de sinónimos
2.2	Facilidad de aplicación y flexibilidad para adopción de criterios de diseño
2.3	Mantenibilidad del Modelo
2.4	Reutilización del Modelo
3.1	Documentación del modelo
3.2	Jerarquización de los conceptos del modelo
3.3	Versionado en proceso Iterativo
3.4	Producto como insumo para la construcción del sistema modelado
RUP	Alternativa metodología RUP
LEL	Alternativa metodología LEL
ONTOLOGIAS	Alternativa metodología ONTOLOGIAS

Ponderación de Criterios

Tabla 6.2. Ponderación de criterios en metodologías para AHP

ID	Descripcion	Puntaje
1	Facilidad de entendimiento, comprensión y validación de los modelos por parte del experto del dominio	3
2	Facilidad de aplicación de la Metodología y grado de representación del dominio a través del modelo obtenido	1
3	Facilidad de comunicación entre analistas y desarrolladores aportados por la Metodología y soporte documental para todas las etapas en el proceso de desarrollo	2

Nota: Mientras más alta es la asignación del puntaje, resulta de menor prioridad

Ponderación de Subcriterios

Tabla 6.3. Ponderación de subcriterios en metodologías para AHP

Facilidad de entendimiento, comprensión y validación de los modelos por parte del experto del dominio

ID	Descripcion	Puntaje
1.1	Claridad Conceptual	5
1.2	Potencialidad para abstraer esencia del dominio	3
1.3	Identificación de la fuente de Requerimientos	6
1.4	Facilidad de entendimiento con el usuario del Dominio	1
1.5	Validación del modelo resultante	2
1.6	Facilidad de trazabilidad de requerimientos	4

Facilidad de aplicación de la Metodología y grado de representación del dominio a través del modelo obtenido

ID	Descripcion	Puntaje
2.1	Reducción de ambigüedades sobre conceptos y manejo de sinónimos	1
2.2	Facilidad de aplicación y flexibilidad para adopción de criterios de diseño	2
2.3	Mantenibilidad del Modelo	3
2.4	Reutilización del Modelo	4

Facilidad de comunicación entre analistas y desarrolladores aportados por la Metodología y soporte documental para todas las etapas en el proceso de desarrollo

ID	Descripcion	Puntaje
3.1	Documentación del modelo	2
3.2	Jerarquización de los conceptos del modelo	3
3.3	Versionado en proceso Iterativo	4
3.4	Producto como insumo para la construcción del sistema modelado	1

6.3.2.2. Modelo Jerárquico para Herramientas

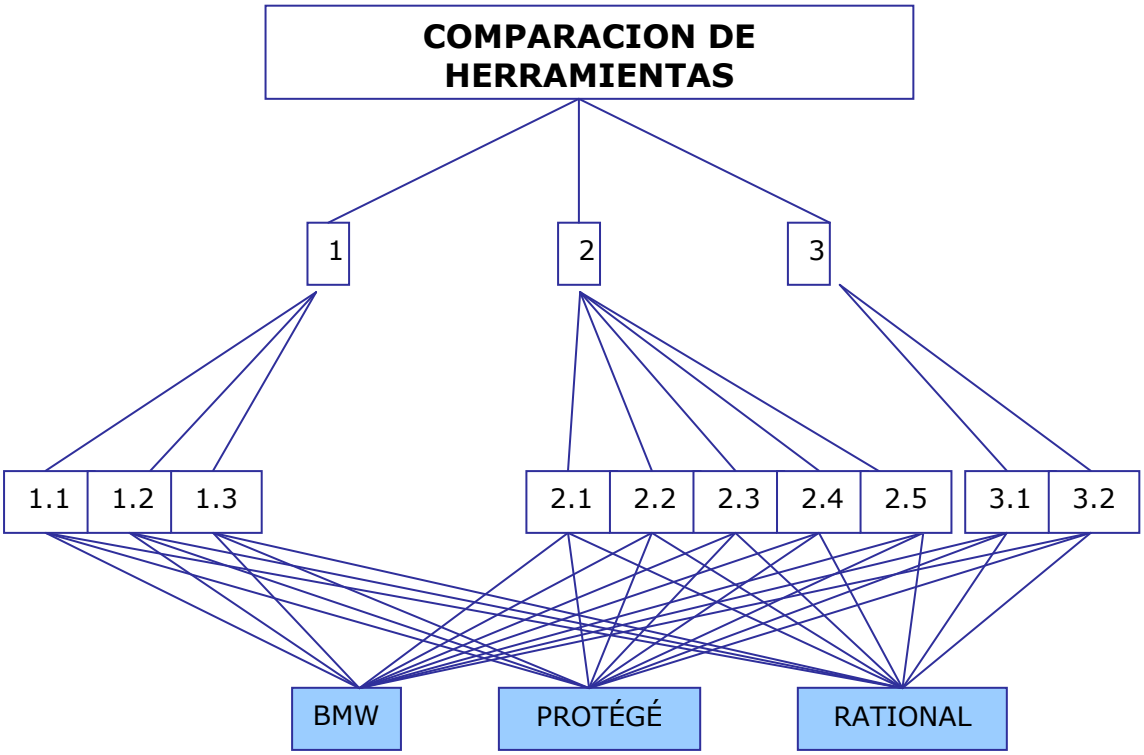


Figura 6.3. Modelo jerárquico AHP – comparación herramientas

Referencias:

Tabla 6.4. Descripción modelo jerárquico AHP en comparación herramientas

ID	DESCRIPCION
1	Facilidad de uso y aprendizaje
2	Prestaciones en la actividad de modelado
3	Facilidad de comunicación con etapas posteriores de desarrollo
1.1	Facilidad de instalación y configuración
1.2	Curva de Aprendizaje de la Herramienta
1.3	Portabilidad
2.1	Capturar en forma fiel y precisa la abstracción del modelo
2.2	Facilidad de introducir cambios en el diseño
2.3	Soporte a proceso iterativo y manejo de versionado
2.4	Visualización a través de diferentes vistas del modelo
2.5	Uso de notaciones y simbología que faciliten el entendimiento del usuario
3.1	Uso de notaciones y simbología que faciliten el entendimiento del equipo de desarrollo en la construcción del modelo
3.2	Facilidad de mapeo directo del modelo para la construcción del sistema
BMW	Alternativa herramienta BMW
PROTÉGÉ	Alternativa herramienta Protégé
RATIONAL	Alternativa herramienta Rational

Ponderación de Criterios

Tabla 6.5. Ponderación de criterios en herramientas para AHP

ID	Descripcion	Puntaje
1	Facilidad de uso y aprendizaje	2
2	Prestaciones en la actividad de modelado	1
3	Facilidad de comunicación con etapas posteriores de desarrollo	3

Nota: Mientras más alta es la asignación del puntaje, resulta de menor prioridad

Ponderación de Subcriterios

Tabla 6.6. Ponderación de subcriterios en herramientas para AHP

Facilidad de uso y aprendizaje

ID	Descripcion	Puntaje
1.1	Facilidad de instalación y configuración	2
1.2	Curva de Aprendizaje de la Herramienta	1
1.3	Portabilidad	3

Prestaciones en la actividad de modelado

ID	Descripcion	Puntaje
2.1	Capturar en forma fiel y precisa la abstracción del modelo	1
2.2	Facilidad de introducir cambios en el diseño	4
2.3	Soporte a proceso iterativo y manejo de versionado	5
2.4	Visualización a través de diferentes vistas del modelo	3
2.5	Uso de notaciones y simbología que faciliten el entendimiento del usuario	2

Facilidad de comunicación con etapas posteriores de desarrollo

ID	Descripcion	Puntaje
3.1	Uso de notaciones y simbología que faciliten el entendimiento del equipo de desarrollo en la construcción del modelo	1
3.2	Facilidad de mapeo directo del modelo para la construcción del sistema	2

CAPÍTULO 7

RESUMEN *En este capítulo se efectúa la valoración de acuerdo a los criterios metodológicos propuestos en el capítulo anterior sobre ambos dominios objetos bajo estudio. Se construyen los esquemas conceptuales para ambos dominios a través de la aplicación de las diferentes metodologías/herramientas los cuales quedan plasmados en diferentes anexos.*
Se presentan los resultados de la aplicación del método de comparación multicriterio AHP sobre las diferentes metodologías/herramientas de acuerdo a los criterios y dimensiones de análisis descriptas para la evaluación de las mismas en ambos dominios.

7.1 INTRODUCCIÓN

En este capítulo, se realiza la valoración sobre los criterios seleccionados para cada uno de los dominios seleccionados, en el proceso de construcción de los esquemas conceptuales, que servirán posteriormente, de insumo en la comparación final de las diferentes metodologías/herramientas objeto de estudio del presente trabajo de tesis.

Para reflejar de mejor manera, en el caso de diferir la valoración de una determinada característica, sobre una misma metodología/herramienta al cambiar únicamente de dominio de aplicación, es que se presentará en el tratamiento del segundo ejemplo "Gramáticas Formales y Máquinas Abstractas", sólo las diferencias y/o nuevas apreciaciones encontradas que puedan llevar o no a un cambio de la valoración del criterio evaluado.

7.2. VALORACIÓN DE CRITERIOS: DOCENTES A CURSOS

En esta sección se valorará los distintos criterios sobre el simple dominio de aplicación "Docentes a Cursos" para cada una de las Metodologías/Herramientas objetos de comparación. A cada uno de los criterios se le otorgará una valuación de acuerdo a una determinada escala, y se reflejará sobre esta característica evaluada el o los aspectos más sobresalientes.

7.2.1. Léxico extendido del lenguaje / BMW

El desarrollo de la Metodología Client Oriented Requirements Baseline, junto con la aplicación de la Herramienta Baseline Mentor Workbench (BMW) objeto de estudio del presente trabajo, aplicadas sobre el Sistema a Modelar: Docentes a cursos, se encuentra disponible en el Anexo II. En este anexo se puede ver el desarrollo paso a paso de la aplicación de la metodología y el resultado del modelo obtenido a través de la aplicación de la herramienta BMW. Estas aplicaciones, por un lado la aplicación de la metodología y por otro, el desarrollo de la herramienta, han servido para validar el modelo resultante, ya que fueron desarrolladas íntegramente obteniendo resultados idénticos en cuanto a las tarjetas CRC obtenidas.

7.2.1.1. Valoración de criterios.

Valoración Metodologías/Herramientas	
Metodología	Client Oriented Requirements Baseline
Herramienta	Baseline Mentor Workbench (BMW)

Tabla 7.1. Valoración metodología LEL en "Docentes a cursos"

Metodología		
Criterio evaluado	Valoración	Observaciones
Claridad Conceptual.	E	Se puede determinar claramente la representación y vinculación de los conceptos del modelo.
Potencialidad para abstraer esencia del dominio.	MB	La agrupación de conceptos representados en futuras clases abstractas no se evidencian en las tarjetas CRC obtenidas, solo toman relevancia las clases activas dentro del dominio, aunque sus atributos pueden no estar presentes
Identificación de la fuente de Requerimientos.	B	La fuente no es identificada, salvo en las anotaciones que surjan de las entrevistas, no existen formalidades acerca de su documentación.
Reducción de ambigüedades sobre conceptos y manejo de sinónimos	MB	Se pueden representar conceptos sinónimos de manera explícita. Hay que poner especial atención a los criterios de circularidad y vocabulario mínimo, ya que el uso del lenguaje natural tiende a provocar ambigüedades.
Facilidad de aplicación y flexibilidad para adopción de criterios de diseño	B	La metodología de modelado resulta de fácil aplicación pero requiere de mucho cuidado y precisión al catalogar conceptos. Hay que realizar varias iteraciones para dejar cabalmente precisado estos conceptos. Hay que tener especial cuidado con la clasificación de las entradas del LEL, ya que de no seguir un criterio de uniforme en la catalogación de las mismas producen inconsistencia en las nociones e impacto, por lo tanto en el modelo resultante. Los criterios de diseño que influyen en las decisiones de modelado, sobre todo en la agrupación de conceptos en clases Abstractas.
Facilidad de entendimiento con el usuario del Dominio.	E	El modelo es entendible por el usuario del Dominio, ya que usa el lenguaje natural por él propuesto. Es posible validar el modelo con el usuario en varias etapas (LEL / Escenarios / Tarjetas CRC). La simplificación conceptual del modelo vendrá dada por el grado de granularidad propuesto por el analista.
Mantenibilidad del Modelo	MB	El modelo es mantenible a través del tiempo sin demasiada complejidad.
Reutilización del Modelo	B	El modelo es reutilizable solo en la medida que el nuevo dominio sea una extensión al mismo y no presente diferencias conceptuales
Documentación del modelo.	MB	Es posible contar con una documentación detallada del modelo en base a la metodología como así también en la herramienta.
Jerarquización de los conceptos del modelo	R	Si bien permite una taxonomía de conceptos que facilita el entendimiento y su modelado los mismos no son plasmados en las tarjetas CRC.
Validación del modelo	MB	Es validado en varios puntos, por medio de dos

resultante.		caminos. Por el usuario ya que en todo momento es capaz de comprender el proceso y lenguaje utilizado y el segundo es la validación de las interacciones y relaciones de conceptos.
Versionado en proceso Iterativo.	B	No provee las facilidades para realizar un versionado, solo esto se logra con la ejecución de la herramienta
Facilidad de trazabilidad de requerimientos.	MB	La trazabilidad entre entradas al LEL y tarjetas CRC es propuesta por definición metodológica.
Producto como insumo para la construcción del sistema modelado.	R	Las Clases obtenidas a través de las fichas CRC resultan incompletas en lo que se refiere a sus atributos y no completas la jerarquía (Ausencia de Clases abstractas).

Tabla 7.2. Valoración herramienta BMW en "Docentes a Cursos"

Herramienta		
Criterio evaluado	Valoración	Observaciones
Facilidad de instalación y configuración.	E	Es fácil de instalar y configurar, muy intuitivo.
Curva de Aprendizaje de la Herramienta	MB	De muy fácil aprendizaje
Capturar en forma fiel y precisa la abstracción del modelo.	MB	Es posible plasmar todas las particularidades del modelo en lo que respecta a sus símbolos y definiciones, no así en lo referente a las responsabilidades cuando existen jerarquías de conceptos
Facilidad de introducir cambios en el diseño.	B	No resulta fácil la incorporación de nuevos conceptos, sobre todo cuando se altera alguna jerarquía de conceptos en su definición ya que deben ser revidados todas las nociones e impactos relacionados
Soporte a proceso iterativo y manejo de versionado.	MB	Soporta proceso iterativo y control de versionado, es posible actualizar las tarjetas CRC en forma automática posterior a la introducción de cambios en el LEL
Visualización a través de diferentes vistas del modelo.	B	El LEL podemos considerarlo como una vista estática del contexto, mientras es posible considerar como vista dinámica a los escenarios que relacionan estos conceptos.
Uso de notaciones y simbología que faciliten el entendimiento del usuario.	E	La notación es mínima y fácil de entender por el usuario.
Uso de notaciones y simbología que faciliten el entendimiento del equipo de desarrollo en la construcción del modelo.	E	Las notaciones y manejo de simbología es mínimo y especificado dentro de los escenarios, y el resto es uso del lenguaje natural lo que resulta fácil de entender por el equipo de desarrollo en la construcción del modelo.
Facilidad de mapeo directo del modelo para la construcción del sistema	B	No existe mapeo directo del modelo en la construcción del sistema, debe ser construido a partir de las CRC resultantes ampliando las mismas.
Portabilidad	B	No es posible instalar la herramienta en diversas plataformas, aunque el modelo resultante puede ser generado y navegado en formato HTML.

7.2.1.2. Conclusiones sobre la aplicación de la metodología/herramienta.

A modo de conclusión sobre la valoración obtenida en la tabla anterior serán expuestas las consideraciones más relevantes, clasificadas comos aspectos positivos y negativos.

Aspectos Positivos

- Lenguaje natural entendido por el usuario, con uso preciso de las definiciones de los conceptos, lo cual facilita la validación del modelo.
- Resulta un modelo, bien documentado de fácil acceso a sus componentes y sus definiciones a través de la navegación de los mismos.
- De fácil aprendizaje, con poco uso de notaciones y convecciones sintácticas lo que lo hace fácil de entender por parte de los usuarios y equipos de desarrollo en futuras etapas de desarrollo.
- Es posible armar a partir de las tarjetas CRC un diagrama de clases consistente con el modelo, solo es necesario extender a las clases desde las tarjetas CRC agregando jerarquías y atributos de clase.
- Es posible contar con una documentación detallada del modelo en base a la metodología como así también en la herramienta.
- Buen manejo de control de versionado entre iteraciones

Aspectos Negativos:

- Es posible no ser homogéneo en la clasificación de entradas al LEL lo que acarrea problemas en la definición de conceptos.
- No resulta ser un modelo completo en lo que respecta a las clases a implementar, faltan detalles de atributos y clases abstractas en las jerarquías de clases.
- No hay representación visual de los conceptos del modelo, lo cuál no permite una rápida ubicación por parte del usuario.
- Ante cambios sustanciales en la definición de conceptos que impactan en la jerarquía de éstos, hay que revisar, definir y valorar todas las nociones e impactos de las entradas al LEL involucradas.
- Al introducir modificaciones al diseño de clases para las futuras etapas de desarrollo se pierde la trazabilidad y potenciabilidad de mantenimiento del modelo

En resumen es una metodología que resulta sumamente útil en lo que respecta a las actividades de exploración y elicitación de requerimientos, forzando a ser preciso en la definición de conceptos, utilizando un vocabulario mínimo con utilización de sinónimos en la

definición de los mismos, los cuales están a la vista en el proceso de definición y documentación. Permite al usar poca notación sintáctica y lenguaje natural en la definición de conceptos que el usuario sea capaz de comprender cabalmente el modelo. Permite un proceso iterativo controlado con manejo de versiones. La deficiencia más marcada es que al no poder hacer un mapeo directo con el diseño de clases a utilizar en futuras etapas de desarrollo exige un esfuerzo adicional en el mantenimiento del modelo.

7.2.2. Ontologías / Protégé-2000

El desarrollo de la aplicación de la ontología de diseño del sistema de ejemplo de aplicación "Sistemas Docentes" se encuentra detallada en el Anexo III del presente trabajo, en donde se refleja cada uno de los pasos de la metodología, el modelo resultante de la aplicación de la herramienta, y la ejecución de las preguntas de competencia para la validación del modelo.

7.2.2.1. Valoración de criterios.

Valoración Metodologías/Herramientas	
Metodología	*Ontology Development 101*
Herramienta	Protégé-2000 Ver. 1.7

Tabla 7.3. Valoración metodología ontology en "Docentes a Cursos"

Metodología		
Criterio evaluado	Valoración	Observaciones
Claridad Conceptual.	MB	Se puede determinar claramente la representación y vinculación de los conceptos del modelo.
Potencialidad para abstraer esencia del dominio.	B	Sería necesario poder incorporar información referida a la etapa de diseño e implementación. No es posible implementar conceptos relacionados con cálculo de valores de Slots (autoincrementar valores)
Identificación de la fuente de Requerimientos.	B	La fuente es identificada pero no existen formalidades acerca de su documentación.
Reducción de ambigüedades sobre conceptos y manejo de sinónimos	B	No se pueden representar conceptos sinónimos de manera explícita.
Facilidad de aplicación y flexibilidad para adopción de criterios de diseño	B	La metodología de modelado es fácil de aplicar pero se presentan inconvenientes en el momento de definir criterios de diseño que influyen en las decisiones de modelado.
Facilidad de entendimiento con el usuario del Dominio.	E	El modelo es entendible por el usuario del Dominio pues es una simplificación conceptual del mismo.
Mantenibilidad del Modelo	MB	El modelo es mantenible a través del tiempo sin demasiada complejidad.
Reutilización del Modelo	MB	El modelo es reutilizable pero es necesario adaptarlo al dominio de aplicación.
Documentación del modelo.	MB	Es posible contar con una documentación detallada del modelo en base a la metodología

		como así también en la herramienta.
Jerarquización de los conceptos del modelo	E	Permite una taxonomía de conceptos que facilita el entendimiento y su modelado.
Validación del modelo resultante.	E	Es posible validar el modelo resultante con respecto a las Preguntas de Competencia a través de Queries que permitan determinar la relación entre ambos.
Versionado en proceso Iterativo.	R	No provee las facilidades para realizar un versionado y gestión de la configuración del modelo.
Facilidad de trazabilidad de requerimientos.	MB	Se puede determinar la trazabilidad de los requerimientos a través de las preguntas de competencia.
Producto como insumo para la construcción del sistema modelado.	R	Los frames o clases obtenidas es frecuente que difieran del diseño de clases en una futura implementación Orientada a Objetos.

Tabla 7.4. Valoración herramienta Protégé en "Docentes a Cursos"

Herramienta		
Criterio evaluado	Valoración	Observaciones
Facilidad de instalación y configuración.	E	Es fácil de instalar y configurar, muy intuitivo.
Curva de Aprendizaje de la Herramienta	MB	De muy fácil aprendizaje
Capturar en forma fiel y precisa la abstracción del modelo.	B	Existen ciertas consideraciones de implementación que no se pueden plasmar en el modelo.
Facilidad de introducir cambios en el diseño.	MB	Es fácil la incorporación de nuevos conceptos, como así también modificar sus relaciones.
Soporte a proceso iterativo y manejo de versionado.	R	No brinda prestaciones compatibles con el manejo de versionado, pero soporta el proceso iterativo a partir de las sucesivas mejoras que es posible incorporarle al modelo.
Visualización a través de diferentes vistas del modelo.	R	No existen vistas en el modelo.
Uso de notaciones y simbología que faciliten el entendimiento del usuario.	MB	La notación es fácil de entender por el usuario.
Uso de notaciones y simbología que faciliten el entendimiento del equipo de desarrollo en la construcción del modelo.	E	La notación es fácil de entender por el equipo de desarrollo en la construcción del modelo.
Facilidad de mapeo directo del modelo para la construcción del sistema	R	No existe mapeo directo del modelo en la construcción del sistema.
Portabilidad	MB	Es posible instalar la herramienta en diversas plataformas.

7.2.2.2. Conclusiones sobre la aplicación de la metodología/herramienta.

A modo de conclusión sobre la valoración de la Metodología/herramienta desarrollada sobre este simple ejemplo, se puede concluir que la Metodología *"Ontology Development 101"* que se basa fuertemente en poder dar respuestas a preguntas de

competencia de la ontología sobre el dominio a modelar y el uso de la herramienta *"Protégé-2000 Ver. 1.7"*, pueden ser clasificados en aspectos positivos y negativos.

Aspectos Positivos

- El modo de inspección que ofrece la herramienta por medio de la funcionalidad de Queries permite evaluar el diseño de la ontología posibilitando la verificación de las Preguntas de Competencia formuladas a través del desarrollo de la metodología, sin embargo es acotado a una serie de relaciones entre los conceptos según la funcionalidad de consultas provista por la herramienta. Cabe destacar que es posible utilizar otras herramientas más específicas para la generación de consultas sobre la ontología, pero en este caso, al evaluar Protégé, hemos optado por utilizar el módulo de consultas que el mismo ofrece para evaluar su funcionalidad.
- El entorno visual ofrece una fácil, rápida e intuitiva interacción con el usuario.
- Permite determinar claramente la representación y vinculación de los conceptos del modelo.
- En cuanto a la metodología de modelado, es fácil su aplicación pero se presentan inconvenientes en el momento de definir criterios de diseño que influyen en las decisiones de modelado.
- El modelo ontológico puede ser reutilizado por otras ontologías.
- Es posible contar con una documentación detallada del modelo en base a la metodología como así también en la herramienta.
- Permite una taxonomía de conceptos que facilita el entendimiento y su modelado.
- La representación visual de los conceptos del modelo permite su fácil entendimiento del usuario.

Aspectos Negativos:

- La herramienta no soporta la representación de sinónimos de conceptos de manera explícita.
- Ciertas consideraciones de implementación como autoincrementar el valor de un slot o valores calculables, no pueden ser representados más que con solo un comentario en el campo del slot.
- No permite el borrado en cadena, es decir, si se borra un slot de un concepto, el mismo seguirá vigente en el entorno del proyecto, por lo cual hay que borrarlo también del proyecto.
- No permite representar las instancias a través de alias que permitan su identificación univoca.

- No proporciona utilidades adaptadas al dominio de elicitación de requerimientos, tales como aspectos a tener en cuenta para el análisis, diseño e implementación del software.
- No se interesa desde la perspectiva de punto de vista del cliente/usuario, aunque puede ser subsanado con las respuestas o preguntas de competencias que debe ser capaz de responder la ontología.
- No brinda prestaciones compatibles con el manejo de versionado y gestión de la configuración, pero soporta el proceso iterativo a partir de las sucesivas mejoras que es posible incorporarle al modelo.

En resumen, el conjunto de metodología/herramienta seleccionada, es útil para la exploración del dominio en la etapa de elicitación, aunque sería importante poder contar con diferentes vistas (estática-Dinámica), aunque en parte esta deficiencia, en referencia a la validación del modelo es salvada con las preguntas de competencia.

La metodología se basa fuertemente en la representación de conceptos, los cuales pueden o no coincidir con los objetos a construir en la implementación del sistema.

7.2.3. RUP/rational (rational unified process) / rational rose

El desarrollo del modelo conceptual del sistema de ejemplo de aplicación "Sistemas Docentes" se encuentra detallada en el Anexo IV del presente trabajo, en donde se refleja cada una de las vistas en el modelado mediante la aplicación de la metodología seleccionada.

7.2.3.1. Valoración de criterios.

Valoración Metodologías/Herramientas	
Metodología	RUP (Rational Unified Process)
Herramienta	Rational Rose (UML 1.0)

Tabla 7.5. Valoración metodología RUP/UML en "Docentes a Cursos"

Metodología		
Criterio evaluado	Valoración	Observaciones
Claridad Conceptual.	E	Se puede determinar claramente la representación y vinculación de los conceptos del modelo.
Potencialidad para abstraer esencia del dominio.	E	Es posible lograr un alto nivel de abstracción por medio de conceptos del paradigma orientado a objetos como Encapsulamiento, Polimorfismo, lo que permite un alto grado de reutilizacion de los artefactos obtenidos mediante la aplicación metodológica.
Identificación de la fuente de Requerimientos.	E	La metodología permite detallar en la documentación la fuente que provee los requerimientos por medio de la ERS (Especificación de Requerimientos del Software).
Reducción de ambigüedades sobre conceptos y manejo de sinónimos	B	Existen ciertas ambigüedades en la descripción de los casos de uso lo que puede llegar a dificultar su entendimiento e interpretación ya que está expresada en lenguaje natural.

Facilidad de aplicación y flexibilidad para adopción de criterios de diseño	B	La metodología de modelado es compleja de aplicar en casos de dominio restringido muy pequeño y sencillo, se adapta mejor para dominios más complejos y de mayor tamaño, sin embargo, se presentan inconvenientes en el momento de definir criterios de diseño que influyen en las decisiones de modelado.
Facilidad de entendimiento con el usuario del Dominio.	MB	El usuario deberá comprender la simbología para luego ver si el modelo es entendible. El usuario deberá comprenderlo por medio de los diagramas UML.
Mantenibilidad del Modelo	E	El modelo es mantenible a través del tiempo ya que RUP posibilita la Administración de Configuración y Gestión del Cambio.
Reutilización del Modelo	MB	El modelo es reutilizable, en la medida que sea adaptarlo y modificado al nuevo dominio de aplicación.
Documentación del modelo.	E	Es posible contar con una documentación detallada del modelo en base a la metodología como así también en la herramienta.
Jerarquización de los conceptos del modelo	E	Permite una taxonomía de clases que facilita el entendimiento y su modelado.
Validación del modelo resultante.	MB	El modelo es validado iterativamente durante todo el proceso de desarrollo de software y la metodología brinda casos de Prueba para la validación de los resultados obtenidos. Solo será efectiva la validación, en virtud al grado de comprensión de los diagramas por parte del usuario.
Versionado en proceso Iterativo.	E	El versionado de los artefactos del modelo se logra mediante la Administración de Configuración y Gestión del Cambio que soporta RUP.
Facilidad de trazabilidad de requerimientos.	E	A partir del modelo de Casos de uso, se puede lograr la trazabilidad en el proceso de desarrollo de software a través de sus fases.
Producto como insumo para la construcción del sistema modelado.	MB	Las clases obtenidas en el modelo de diseño, coinciden con la implementación Orientada a Objetos.

Tabla 7.6. Valoración herramienta rational en "Docentes a Cursos"

Herramienta		
Criterio evaluado	Valoración	Observaciones
Facilidad de instalación y configuración.	E	Es fácil de instalar y configurar, muy intuitivo.
Curva de Aprendizaje de la Herramienta	B	De fácil aprendizaje, pero se necesitan conocimientos previos de UML para poder utilizarla eficientemente y una extensa lista de símbolos.
Capturar en forma fiel y precisa la abstracción del modelo.	MB	Es posible plasmar todas las particularidades del dominio representado en el modelo.
Facilidad de introducir cambios en el diseño.	B	Introducir cambios en fases avanzadas del proceso conlleva a costos que pueden variar de acuerdo a la magnitud y la importancia del cambio introducido.
Soporte a proceso iterativo y	MB	Soporta proceso iterativo y control de versionado.

manejo de versionado.		
Visualización a través de diferentes vistas del modelo.	E	En base a los diferentes diagramas de UML se puede representar el modelo desde diferentes vistas.
Uso de notaciones y simbología que faciliten el entendimiento del usuario.	MB	La notación es fácil de entender por el usuario pero requiere una explicación introductoria acerca de UML para mejorar su entendimiento.
Uso de notaciones y simbología que faciliten el entendimiento del equipo de desarrollo en la construcción del modelo.	MB	La notación es fácil comprensión pero requiere una explicación introductoria acerca de UML para mejorar su entendimiento.
Facilidad de mapeo directo del modelo para la construcción del sistema	E	Existe un mapeo directo del modelo en la construcción del sistema con su implementación.
Portabilidad	E	Es posible instalar la herramienta en diversas plataformas.

7.2.3.2. Conclusiones sobre la aplicación de la metodología/herramienta.

Aspectos Positivos

- Permite definir criterios de diseño que influyen en las decisiones de modelado.
- Es posible reutilizar el modelo.
- El modelo es mantenible a través del tiempo.
- Es posible contar con una documentación detallada del modelo en base a la metodología.
- Es posible crear una taxonomía de clases que facilita el entendimiento y su modelado.

Aspectos Negativos:

- De fácil aprendizaje, con poco uso de notaciones y convecciones sintácticas lo que lo hace fácil de entender por parte de los usuarios y equipos de desarrollo en futuras etapas de desarrollo.
- Debido a que la descripción de los casos de uso es en lenguaje natural existen ambigüedades en la descripción de los casos de uso lo que puede llegar a dificultar su entendimiento e interpretación ya sea por parte del usuario final como el usuario técnico.
- La metodología de modelado es compleja de aplicar en casos de dominio restringido muy pequeño y sencillo, se adapta mejor para dominios más complejos y de mayor tamaño.
- Desde el punto más purista de la concepción del modelo objetos, los casos de uso no conducen a los mismos ya que estos son de naturaleza secuencial

7.2.4. Resumen sobre valoración de Criterios Obtenidos

A continuación se presentan a modo de resumen, un cuadro comparativo sobre los criterios evaluados para las metodologías y herramientas sobre el dominio de “Docentes a Curso”

7.2.4.1. Metodologías Evaluadas

Tabla 7.7. Resumen valoración de Criterios – Metodologías "Docentes a Cursos"

Metodología			
Criterio evaluado	LEL	Ontologías	RUP
Claridad Conceptual.	E	MB	E
Potencialidad para abstraer esencia del dominio.	MB	MB	E
Identificación de la fuente de Requerimientos.	B	B	E
Reducción de ambigüedades sobre conceptos y manejo de sinónimos	MB	B	B
Facilidad de aplicación y flexibilidad para adopción de criterios de diseño	B	B	B
Facilidad de entendimiento con el usuario del Dominio.	E	MB	MB
Mantenibilidad del Modelo	MB	MB	E
Reutilización del Modelo	B	MB	MB
Documentación del modelo.	MB	MB	E
Jerarquización de los conceptos del modelo	R	E	E
Validación del modelo resultante.	MB	E	MB
Versionado en proceso Iterativo.	B	R	E
Facilidad de trazabilidad de requerimientos.	MB	MB	E
Producto como insumo para la construcción del sistema modelado.	R	R	MB

7.2.4.2. Herramientas evaluadas

Tabla 7.8. Resumen valoración de Criterios – Metodologías "Docentes a Cursos"

Herramienta			
Criterio evaluado	BMW	Protégé	Rational
Facilidad de instalación y configuración.	E	E	E
Curva de Aprendizaje de la Herramienta	MB	MB	B
Capturar en forma fiel y precisa la abstracción del modelo.	MB	B	MB
Facilidad de introducir cambios en el diseño.	B	MB	B
Soporte a proceso iterativo y manejo de versionado.	MB	B	MB
Visualización a través de diferentes vistas del modelo.	B	R	E
Uso de notaciones y simbología que faciliten el entendimiento del usuario.	E	E	MB
Uso de notaciones y simbología que faciliten el entendimiento	E	E	MB

del equipo de desarrollo en la construcción del modelo.			
Facilidad de mapeo directo del modelo para la construcción del sistema	B	R	E
Portabilidad	B	MB	E

7.3. VALORACIÓN DE CRITERIOS: GRAMATICAS FORMALES Y MAQUINAS ABSTRACTAS

En esta sección se valorará los distintos criterios sobre el dominio seleccionado de Máquinas Abstractas y Gramáticas Formales para cada una de las Metodologías/Herramientas objetos de comparación. Se tomará como base las apreciaciones observadas en el Sistemas Docentes añadiendo comentarios sobre las nuevas apreciaciones.

7.3.1. Léxico extendido del lenguaje / BMW

El desarrollo de la Metodología Client Oriented Requirements Baseline, junto con la aplicación de la Herramienta Baseline Mentor Workbench (BMW) objeto de estudio del presente trabajo, aplicadas sobre el Sistema a Modelar: Máquinas Abstractas y Gramáticas Formales, se encuentra disponible en el Anexo V.

7.3.1.1. Valoración de criterios.

Tabla 7.9 Valoración metodología LEL en Gramáticas Formales y Máquinas Abstractas

METODOLOGIA		
Criterio evaluado	**Valoración**	**Nuevas Apreciaciones**
Claridad Conceptual.	MB	Al crecer el dominio en tamaño y complejidad se observan dificultades en la implantación del principio de circularidad y de vocabulario mínimo
Potencialidad para abstraer esencia del dominio.	B	Al crecer el dominio en tamaño y complejidad se observan dificultades en representar taxonomías de conceptos
Identificación de la fuente de Requerimientos.	B	
Reducción de ambigüedades sobre conceptos y manejo de sinónimos	MB	
Facilidad de aplicación y flexibilidad para adopción de criterios de diseño	B	
Facilidad de entendimiento con el usuario del Dominio.	MB	Al Crecer el Dominio las relaciones y circularidad complican el seguimiento

Mantenibilidad del Modelo	MB	El modelo es mantenible a través del tiempo sin demasiada complejidad.
Reutilización del Modelo	B	
Documentación del modelo.	MB	
Jerarquización de los conceptos del modelo	R	Incrementa las dificultades en las taxonomías de conceptos.
Validación del modelo resultante.	MB	
Versionado en proceso Iterativo.	B	
Facilidad de trazabilidad de requerimientos.	MB	
Producto como insumo para la construcción del sistema modelado.	R	

Tabla 7.10 Valoración metodología LEL en Gramáticas Formales y Máquinas Abstractas

HERRAMIENTA		
Criterio evaluado	**Valoración**	**Nuevas Apreciaciones**
Facilidad de instalación y configuración.	E	
Curva de Aprendizaje de la Herramienta	MB	
Capturar en forma fiel y precisa la abstracción del modelo.	MB	
Facilidad de introducir cambios en el diseño.	B	
Soporte a proceso iterativo y manejo de versionado.	MB	
Visualización a través de diferentes vistas del modelo.	R	Al incrementarse el dominio se hace más necesario contar con una interfaz gráfica
Uso de notaciones y simbología que faciliten el entendimiento del usuario.	B	El conjunto de Símbolos admitidos en la descripción no presenta riqueza para describir fórmulas matemáticas
Uso de notaciones y simbología que faciliten el entendimiento del equipo de desarrollo en la construcción del modelo.	MB	No presenta una interfaz visual. La descripción es narrativa y la lo cual dificulta el proceso de construcción a partir del modelo
Facilidad de mapeo directo del modelo para la construcción del sistema	B	
Portabilidad	B	

7.3.1.1. Nuevas Apreciaciones.

En la construcción del Modelo de las Gramáticas Formales y las Máquinas Abstractas, podemos adicionar a las observaciones identificadas anteriormente las siguientes apreciaciones: Se incrementa la dificultad de implementar una taxonomía de conceptos. El

poder expresivo del vocabulario disponible para utilizar no es el adecuado cuando el dominio a modelar es abstracto y cercano a las matemáticas. Para la introducción a cambios en el modelo las relaciones entre conceptos dificultan la tarea de modelado

En referencia a la herramienta, las apreciaciones realizadas sobre el primer dominio, no se ven afectadas en gran medida con la aplicación del nuevo dominio, salvo en lo referente a que la herramienta BMW no presenta posibilidad de escribir símbolos necesarios para describir formulas matemáticas, y la necesidad de contar con una interfaz gráfica para facilitar la visualización de conceptos.

7.3.2. Ontologías / Protégé-2000

El desarrollo de la aplicación de la ontología de diseño del sistema de ejemplo de aplicación "Gramáticas Formales y Máquinas Abstractas" se encuentra detallada en el Anexo VI del presente trabajo, en donde se refleja cada uno de los pasos de la metodología, el modelo resultante de la aplicación de la herramienta, y la ejecución de las preguntas de competencia para la validación del modelo.

7.3.2.1. Valoración de criterios.

Tabla 7.11 Valoración Ontology Development 101: Gramáticas Formales y Máquinas Abstractas

METODOLOGIA		
Criterio evaluado	**Valoración**	**Nuevas Apreciaciones**
Claridad Conceptual.	B	A medida que el dominio crece es más difícil visualizar la conceptualización del mismo.
Potencialidad para abstraer esencia del dominio.	B	
Identificación de la fuente de Requerimientos.	B	
Reducción de ambigüedades sobre conceptos y manejo de sinónimos	B	
Facilidad de aplicación y flexibilidad para adopción de criterios de diseño	B	
Facilidad de entendimiento con el usuario del Dominio.	E	
Mantenibilidad del Modelo	MB	
Reutilización del Modelo	MB	
Documentación del modelo.	MB	
Jerarquización de los conceptos del modelo	E	
Validación del modelo resultante.	MB	La herramienta básica no permite realizar inferencias complejas sobre el modelo, para dar respuestas a las preguntas de competencia, sin embargo existen razonadores adicionales.

Versionado en proceso Iterativo.	R	
Facilidad de trazabilidad de requerimientos.	MB	
Producto como insumo para la construcción del sistema modelado.	R	

Tabla 7.12 Valoración comparativa herramienta Protégé-2000 Ver. 1.7 En GF y MA

HERRAMIENTA		
Criterio evaluado	**Valoración**	**Nuevas Apreciaciones**
Facilidad de instalación y configuración.	E	
Curva de Aprendizaje de la Herramienta	MB	
Capturar en forma fiel y precisa la abstracción del modelo.	B	
Facilidad de introducir cambios en el diseño.	MB	
Soporte a proceso iterativo y manejo de versionado.	R	
Visualización a través de diferentes vistas del modelo.	R	
Uso de notaciones y simbología que faciliten el entendimiento del usuario.	MB	
Uso de notaciones y simbología que faciliten el entendimiento del equipo de desarrollo en la construcción del modelo.	E	
Facilidad de mapeo directo del modelo para la construcción del sistema	R	
Portabilidad	MB	

7.3.2.1. Nuevas apreciaciones.

En la construcción del Modelo de las Gramáticas Formales y las Máquinas Abstractas, sólo podemos adicionar a las observaciones identificadas anteriormente la siguiente apreciación: La gran ventaja que presenta el conjunto Metodología/herramienta que es la posibilidad de aplicar al modelo las preguntas de competencias desaparece, ya que La herramienta básica no permite realizar inferencias complejas sobre el modelo, para dar respuestas a las preguntas de competencia, sin embargo existen razonadores adicionales, para poder evaluar las connotaciones dinámicas que presenta el modelo en el reconocimiento y la generación de cadenas.

7.3.3. RUP/rational (rational unified process) / rational rose

El desarrollo del modelo conceptual del sistema de ejemplo de aplicación "Gramáticas Formales y Máquinas Abstractas" se encuentra detallada en el Anexo VII del presente trabajo, en donde se refleja cada una de las vistas en el modelado mediante la aplicación de la metodología seleccionada.

7.3.3.1. Valoración de criterios.

Tabla 7.13 Valoración metodología RUP en Gramáticas Formales y Máquinas Abstractas

METODOLOGIAS		
Criterio evaluado	**Valoración**	**Nuevas Apreciaciones**
Claridad Conceptual.	MB	En dominios poco triviales o complejos, es más difícil ver claramente la representación conceptual de los elementos del dominio.
Potencialidad para abstraer esencia del dominio.	MB	La abstracción se vuelve complicada al momento de que el dominio se convierte en dominios no triviales. Ejemplo CU: Tipos de Gramática.
Identificación de la fuente de Requerimientos.	E	
Reducción de ambigüedades sobre conceptos y manejo de sinónimos	B	
Facilidad de aplicación y flexibilidad para adopción de criterios de diseño	MB	Al tratarse de un dominio complejo la aplicación de criterios de diseño se vuelve muy útil.
Facilidad de entendimiento con el usuario del Dominio.	MB	
Mantenibilidad del Modelo	E	
Reutilización del Modelo	MB	
Documentación del modelo.	MB	Posibilita una documentación detallada, pero se debe conocer sobre la simbología específica del proceso.
Jerarquización de los conceptos del modelo	E	
Validación del modelo resultante.	MB	
Versionado en proceso Iterativo.	E	
Facilidad de trazabilidad de requerimientos.	E	
Producto como insumo para la construcción del sistema modelado.	MB	

Tabla 7.14 Valoración herramienta Rational Rose Gramáticas Formales y Máquinas Abstractas

HERRAMIENTA		
Criterio evaluado	**Valoración**	**Nuevas Apreciaciones**
Facilidad de instalación y configuración.	E	
Curva de Aprendizaje de la Herramienta	B	
Capturar en forma fiel y precisa la abstracción del modelo.	MB	
Facilidad de introducir cambios en el diseño.	B	
Soporte a proceso iterativo y manejo de versionado.	MB	
Visualización a través de diferentes vistas del modelo.	E	
Uso de notaciones y simbología que faciliten el entendimiento del usuario.	B	En dominios poco triviales, abstractos o cercanos a las Matemáticas es difícil representarlos
Uso de notaciones y simbología que faciliten el entendimiento del equipo de desarrollo en la construcción del modelo.	MB	
Facilidad de mapeo directo del modelo para la construcción del sistema	E	
Portabilidad	E	

7.3.3.2. Nuevas apreciaciones.

En la construcción del Modelo de las Gramáticas Formales y las Máquinas Abstractas, podemos adicionar a las observaciones identificadas sobre el modelo de docentes las siguientes apreciaciones. Con respecto a la Metodología RUP al encontrarnos en un dominio poco trivial o complejo, es más difícil ver claramente la representación conceptual de los elementos del dominio y el proceso de abstracción se vuelve aún más dificultoso. Por otro lado la posibilidad de adoptar diferentes criterios de diseño se vuelve sumamente útil, en lo que respecta a la representación de la realidad atendiendo al mismo tiempo a necesidades de implementación.

Con respecto a la herramienta, no presenta mayores diferencias, sólo en lo que respecta a la representación de conceptos sumamente abstractos o cercanos a las Matemáticas.

7.3.4. Resumen sobre valoración de Criterios Obtenidos

A continuación se presentan a modo de resumen, un cuadro comparativo sobre los criterios evaluados para las metodologías y herramientas sobre el dominio de “Gramáticas Formales y Máquinas Abstractas”

7.3.4.1. Metodologías Evaluadas

Tabla 7.15 Resumen criterios sobre metodologías Gramáticas Formales y Máquinas Abstractas

Metodología			
Criterio evaluado	LEL	Ontologías	RUP
Claridad Conceptual.	MB	B	MB
Potencialidad para abstraer esencia del dominio.	B	B	MB
Identificación de la fuente de Requerimientos.	B	B	E
Reducción de ambigüedades sobre conceptos y manejo de sinónimos	MB	B	B
Facilidad de aplicación y flexibilidad para adopción de criterios de diseño	B	B	MB
Facilidad de entendimiento con el usuario del Dominio.	MB	E	MB
Mantenibilidad del Modelo	MB	MB	E
Reutilización del Modelo	B	MB	MB
Documentación del modelo.	MB	MB	MB
Jerarquización de los conceptos del modelo	R	E	E
Validación del modelo resultante.	MB	MB	MB
Versionado en proceso Iterativo.	B	R	E
Facilidad de trazabilidad de requerimientos.	MB	MB	E
Producto como insumo para la construcción del sistema modelado.	R	R	MB

7.3.4.2. Herramientass Evaluadas

Tabla 7.16 Resumen criterios sobre Herramientas Gramáticas Formales y Máquinas Abstractas

Herramienta			
Criterio evaluado	BMW	Protégé	Rational Rose
Facilidad de instalación y configuración.	E	E	E
Curva de Aprendizaje de la Herramienta	MB	MB	B
Capturar en forma fiel y precisa la abstracción del modelo.	MB	B	MB
Facilidad de introducir cambios en el diseño.	B	MB	B
Soporte a proceso iterativo y manejo de versionado.	MB	R	MB
Visualización a través de diferentes vistas del modelo.	R	R	E
Uso de notaciones y simbología que faciliten el entendimiento del usuario.	B	MB	B
Uso de notaciones y simbología que faciliten el entendimiento	MB	E	MB

del equipo de desarrollo en la construcción del modelo.			
Facilidad de mapeo directo del modelo para la construcción del sistema	B	R	E
Portabilidad	B	MB	E

7.4. APLICACIÓN MÉTODO AHP EN "DOCENTES A CURSOS"

En esta sección se procede a aplicar el método de Análisis Jerárquico de procesos (AHP) descripto en el capítulo VI con la ponderación de los criterios y subcriterios, aplicados sobre las valoraciones efectuadas de los criterios seleccionados tanto para las metodologías y herramientas sobre el dominio "Docentes a Cursos" objeto de estudio.

7.4.1. Modelo Jerárquico de Metodologías

Se aplicará la ponderación de criterios y Modelo jerárquico detallado para la evaluación de metodologías descripto en el capítulo VI

7.4.1.1. Matrices de comparación de pares de criterios

A partir de la ponderación de criterios y subcriterios son creadas las relaciones entre los mismos en matrices de comparación las cuales son el insumo básico para la evaluación mediante el software Expert Choice. Los valores de estas matrices de cálculo son las definidas a continuación

Comparación de Criterios

	1	**2**	**3**
1		-3	-2
2			2
3			

Comparación de Subcriterios

- Facilidad de entendimiento, comprensión y validación de los modelos por parte del experto del dominio

	1.1	**1.2**	**1.3**	**1.4**	**1.5**	**1.6**
1.1		-3	2	-5	-4	-2
1.2			4	3	-2	2
1.3				-6	-5	-3
1.4					2	4
1.5						3
1.6						

- Facilidad de aplicación de la Metodología y grado de representación del dominio a través del modelo obtenido

	2.1	**2.2**	**2.3**	**2.4**
2.1		2	3	4
2.2			2	3
2.3				2
2.4				

- Facilidad de comunicación entre analistas y desarrolladores aportados por la Metodología y soporte documental para todas las etapas en el proceso de desarrollo

	3.1	**3.2**	**3.3**	**3.4**
3.1		2	3	-2
3.2			2	-3
3.3				-4
3.4				

Comparación de las alternativas con respecto a cada Subcriterio

1.1. Claridad Conceptual

	RUP	LEL	ONTOLOGIAS
RUP		1	2
LEL			2
ONTOLOGIAS			

1.2. Potencialidad para abstraer esencia del dominio

	RUP	LEL	ONTOLOGIAS
RUP		2	2
LEL			1
ONTOLOGIAS			

1.3. Identificación de la fuente de Requerimientos

	RUP	LEL	ONTOLOGIAS
RUP		4	4
LEL			1
ONTOLOGIAS			

1.4. Facilidad de entendimiento con el usuario del Dominio

	RUP	LEL	ONTOLOGIAS
RUP		-2	1
LEL			2
ONTOLOGIAS			

1.5. Validación del modelo resultante

	RUP	LEL	ONTOLOGIAS
RUP		1	-2
LEL			-2
ONTOLOGIAS			

1.6. Facilidad de trazabilidad de requerimientos

	RUP	LEL	ONTOLOGIAS
RUP		2	2
LEL			1
ONTOLOGIAS			

2.1. Reducción de ambigüedades sobre conceptos y manejo de sinónimos

	RUP	LEL	ONTOLOGIAS
RUP		-2	1
LEL			2
ONTOLOGIAS			

2.2. Facilidad de aplicación y flexibilidad para adopción de criterios de diseño

	RUP	LEL	ONTOLOGIAS
RUP		1	1
LEL			1
ONTOLOGIAS			

2.3. Mantenibilidad del Modelo

	RUP	LEL	ONTOLOGIAS
RUP		2	2
LEL			1
ONTOLOGIAS			

2.4. Reutilización del Modelo

	RUP	LEL	ONTOLOGIAS
RUP		2	1
LEL			-2
ONTOLOGIAS			

3.1. Documentación del modelo

	RUP	LEL	ONTOLOGIAS
RUP		2	2
LEL			1
ONTOLOGIAS			

3.2. Jerarquización de los conceptos del modelo

	RUP	LEL	ONTOLOGIAS
RUP		6	1
LEL			-6
ONTOLOGIAS			

3.3. Versionado en proceso Iterativo

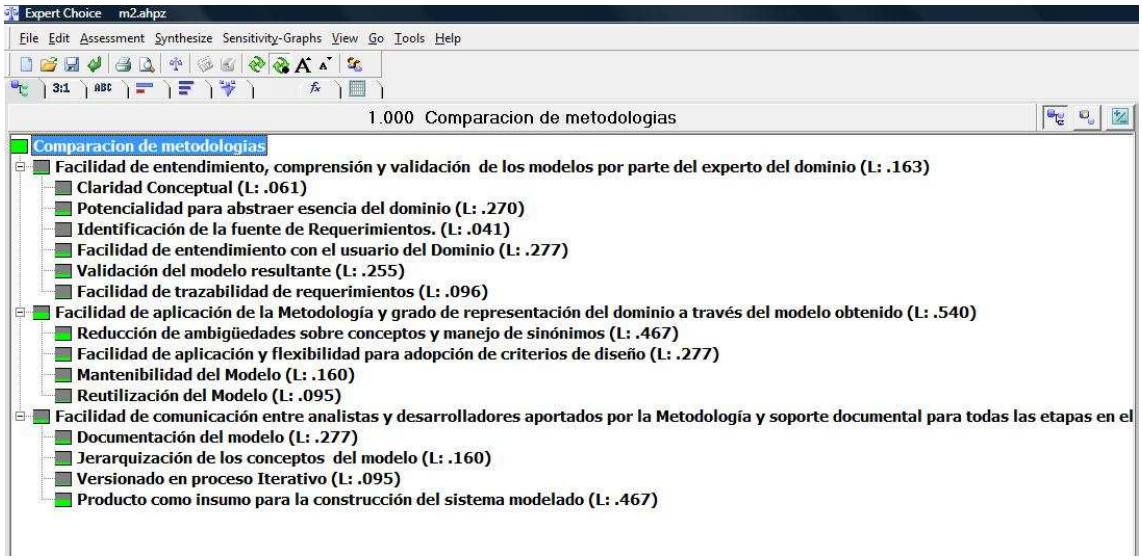
	RUP	LEL	ONTOLOGIAS
RUP		4	6
LEL			2
ONTOLOGIAS			

3.4. Producto como insumo para la construcción del sistema modelado

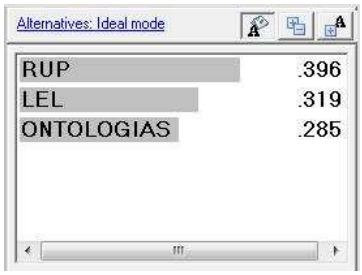
	RUP	LEL	ONTOLOGIAS
RUP		4	4
LEL			1
ONTOLOGIAS			

7.4.1.2. Resultados obtenido del proceso de síntesis

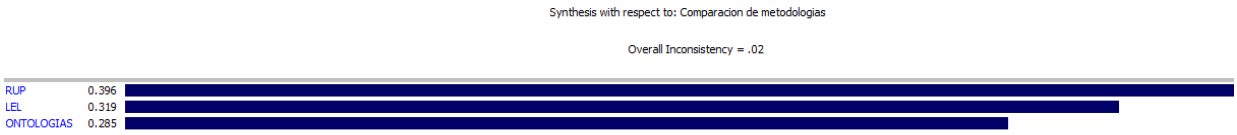
Una vez determinadas la estructura jerárquica y las matrices de comparaciones pareadas se procedió a la carga de dichos datos en el software Expert Choice 11.5, como se muestra en la pantalla 6.1, 6.2 y 6.3, para la resolución por medio del método AHP, con el objetivo de calcular las prioridades de las alternativas con respecto a la meta global.



Pantalla 7.1. Expert Choice - comparación de metodologías para AHP



Pantalla 7.2. Expert Choice - resultado numérico de la comparación de metodologías



Pantalla 7.3. Expert Choice - resultado gráfico de la comparación de metodologías

7.4.1.3. Metodología como mejor alternativa

Con la aplicación del método AHP sobre los criterios y jerarquías planteadas para la evaluación, se puede observar que la alternativa con ponderación prevalente es la RUP, lo que determina que es la mejor alternativa resultante a partir del proceso de selección.

7.4.2. Modelo Jerárquico de Herramientas

Se aplicará la ponderación de criterios y Modelo jerárquico para la evaluación de Herramientas detallado en el capítulo VI

7.4.2.1. Matrices de comparación de pares de criterios

A partir de la ponderación de criterios y subcriterios son creadas las relaciones entre los mismos en matrices de comparación las cuales son el insumo básico para la evaluación mediante el software Expert Choice. Los valores de estas matrices de cálculo son las definidas a continuación

Comparación de Criterios

	1	2	3
1		-2	2
2			3
3			

Comparación de Subcriterios

- Facilidad de uso y aprendizaje

	1.1	1.2	1.3
1.1		-2	2
1.2			3
1.3			

- Prestaciones en la actividad de modelado

	2.1	2.2	2.3	2.4	2.5
2.1		4	5	3	2
2.2			2	-2	-3
2.3				-3	-4
2.4					-2
2.5					

- Facilidad de comunicación con etapas posteriores de desarrollo

	3.1	3.2
3.1		2
3.2		

Comparación de las alternativas con respecto a cada Subcriterio

1.1. Facilidad de instalación y configuración

	BMW	Protégé	Rational
BMW		1	1
Protégé			1
Rational			

1.2. Curva de Aprendizaje de la Herramienta

	BMW	Protégé	Rational
BMW		1	2
Protégé			2
Rational			

1.3. Portabilidad

	BMW	Protégé	Rational
BMW		-2	-4
Protégé			-2
Rational			

2.1. Capturar en forma fiel y precisa la abstracción del modelo

	BMW	Protégé	Rational
BMW		2	1
Protégé			-2
Rational			

2.2. Facilidad de introducir cambios en el diseño

	BMW	Protégé	Rational
BMW		-2	1
Protégé			2
Rational			

2.3. Soporte a proceso iterativo y manejo de versionado

	BMW	Protégé	Rational
BMW		2	1
Protégé			-2
Rational			

2.4. Visualización a través de diferentes vistas del modelo

	BMW	Protégé	Rational
BMW		2	-4
Protégé			-6
Rational			

2.5. Uso de notaciones y simbología que faciliten el entendimiento del usuario

	BMW	Protégé	Rational
BMW		1	2
Protégé			2
Rational			

3.1. Uso de notaciones y simbología que faciliten el entendimiento del equipo de desarrollo en la construcción del modelo

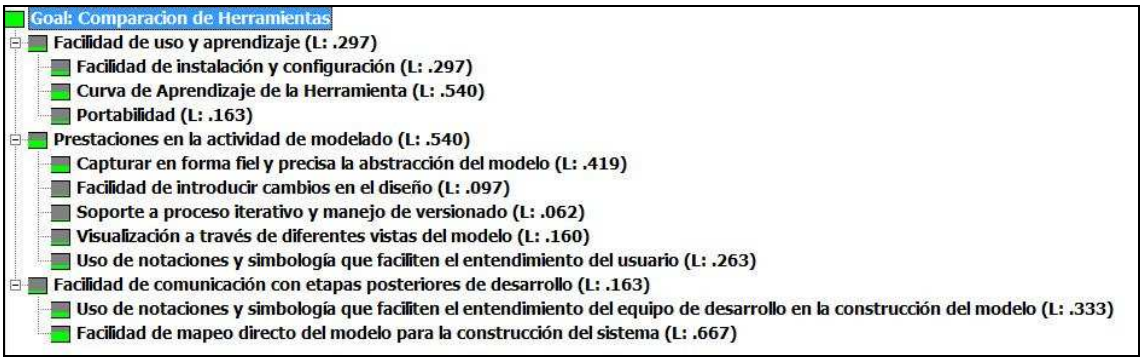
	BMW	Protégé	Rational
BMW		1	2
Protégé			2
Rational			

3.2. Facilidad de mapeo directo del modelo para la construcción del sistema

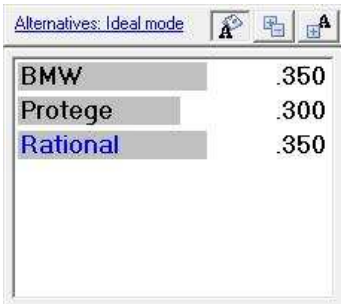
	BMW	Protégé	Rational
BMW		2	-4
Protégé			-6
Rational			

7.4.2.2. Resultados obtenido del proceso de síntesis

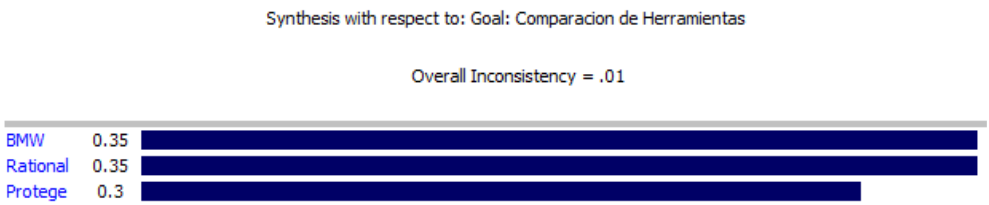
Aplicando el software Expert Choice 11.5 para la resolución por medio del método AHP, se han obtenido los resultados que se muestran en las Pantallas 6.4, 6.5 y 6.6:



Pantalla 7.4. Expert Choice - comparación de herramientas para AHP



Pantalla 7.5. Expert Choice - resultado numérico de la comparación de herramientas



Pantalla 7.6. Expert Choice - resultado gráfico de la comparación de herramientas

7.4.2.3. Herramienta como mejor alternativa

A través, de la aplicación del método AHP, sobre la valoración de los criterios aplicados a las herramientas, se puede observar que las alternativas BMW y Rational poseen ponderaciones equivalentes lo que determina que han sido elegidas como las mejores en el proceso de selección.

7.4.3. Resumen integrado de resultados obtenidos.

A continuación se presenta una tabla que integra los resultados obtenidos mediante el proceso de Análisis Jerárquico (AHP) mediante la selección de criterios y las ponderaciones correspondientes al proceso de evaluación de las diferentes Metodologías/Herramientas aplicadas sobre el dominio simplificado de "Docentes a cursos"

Tabla 7.17 Resultados de comparaciones con proceso AHP

RUP	*0.396*	**RATIONAL**	*0.350*
LEL	0.319	***BMW***	0.350
ONTOLOGIAS	0.285	**PROTÉGÉ**	0.300

En forma coincidente la metodología RUP y la herramienta RATIONAL han quedado posicionadas a partir de la aplicación del proceso de comparación AHP (Proceso de Análisis Jerárquico) como las de mejor perfomance para construir el Modelo Conceptual para la especificación de requerimientos

A su vez, también coincide la metodología LEL con su herramienta BMW en segundo lugar, y ONTOLOGIAS y PROTÉGÉ en tercer lugar.

7.5. APLICACIÓN MÉTODO AHP EN "GRAMATICAS FORMALES Y MAQUINAS ABSTRACTAS"

En esta sección se procede a aplicar el método de Análisis Jerárquico de procesos (AHP) descripto en el capítulo VI con la ponderación de los criterios y subcriterios, aplicados sobre las valoraciones efectuadas de los criterios seleccionados tanto para las metodologías y herramientas sobre el dominio "Gramáticas Formales y Máquinas Abstractas" objeto de estudio.

7.5.1. Modelo Jerárquico de Metodologías

Se aplicará la ponderación de criterios y Modelo jerárquico detallado para la evaluación de metodologías descripto en el capítulo VI

7.5.1.1. Matrices de comparación de pares de criterios

A partir de la ponderación de criterios y subcriterios son creadas las relaciones entre los mismos en matrices de comparación las cuales son el insumo básico para la evaluación mediante el software Expert Choice. Los valores de estas matrices de cálculo son las definidas a continuación

Comparación de Criterios

	1	2	3
1		-3	-2
2			2
3			

Comparación de Subcriterios

- Facilidad de entendimiento, comprensión y validación de los modelos por parte del experto del dominio

	1.1	1.2	1.3	1.4	1.5	1.6
1.1		-3	2	-5	-4	-2
1.2			4	3	-2	2
1.3				-6	-5	-3
1.4					2	4
1.5						3
1.6						

- Facilidad de aplicación de la Metodología y grado de representación del dominio a través del modelo obtenido

	2.1	2.2	2.3	2.4
2.1		2	3	4
2.2			2	3
2.3				2
2.4				

- Facilidad de comunicación entre analistas y desarrolladores aportados por la Metodología y soporte documental para todas las etapas en el proceso de desarrollo

	3.1	**3.2**	**3.3**	**3.4**
3.1		2	3	-2
3.2			2	-3
3.3				-4
3.4				

Comparación de las alternativas con respecto a cada Subcriterio

1.1. Claridad Conceptual

	RUP	LEL	ONTOLOGIAS
RUP		1	2
LEL			2
ONTOLOGIAS			

1.2. Potencialidad para abstraer esencia del dominio

	RUP	LEL	ONTOLOGIAS
RUP		2	2
LEL			1
ONTOLOGIAS			

1.3. Identificación de la fuente de Requerimientos

	RUP	LEL	ONTOLOGIAS
RUP		4	4
LEL			1
ONTOLOGIAS			

1.4. Facilidad de entendimiento con el usuario del Dominio

	RUP	LEL	ONTOLOGIAS
RUP		-2	1
LEL			2
ONTOLOGIAS			

1.5. Validación del modelo resultante

	RUP	LEL	ONTOLOGIAS
RUP		1	1
LEL			1
ONTOLOGIAS			

1.6. Facilidad de trazabilidad de requerimientos

	RUP	LEL	ONTOLOGIAS
RUP		2	2
LEL			1
ONTOLOGIAS			

2.1. Reducción de ambigüedades sobre conceptos y manejo de sinónimos

	RUP	LEL	ONTOLOGIAS
RUP		-2	1
LEL			-2
ONTOLOGIAS			

2.2. Facilidad de aplicación y flexibilidad para adopción de criterios de diseño

	RUP	LEL	ONTOLOGIAS
RUP		2	2
LEL			1
ONTOLOGIAS			

2.3. Mantenibilidad del Modelo

	RUP	LEL	ONTOLOGIAS
RUP		2	2
LEL			1
ONTOLOGIAS			

2.4. Reutilización del Modelo

	RUP	LEL	ONTOLOGIAS
RUP		2	1
LEL			-2
ONTOLOGIAS			

3.1. Documentación del modelo

	RUP	LEL	ONTOLOGIAS
RUP		1	1
LEL			1
ONTOLOGIAS			

3.2. Jerarquización de los conceptos del modelo

	RUP	LEL	ONTOLOGIAS
RUP		4	1
LEL			-4
ONTOLOGIAS			

3.3. Versionado en proceso Iterativo

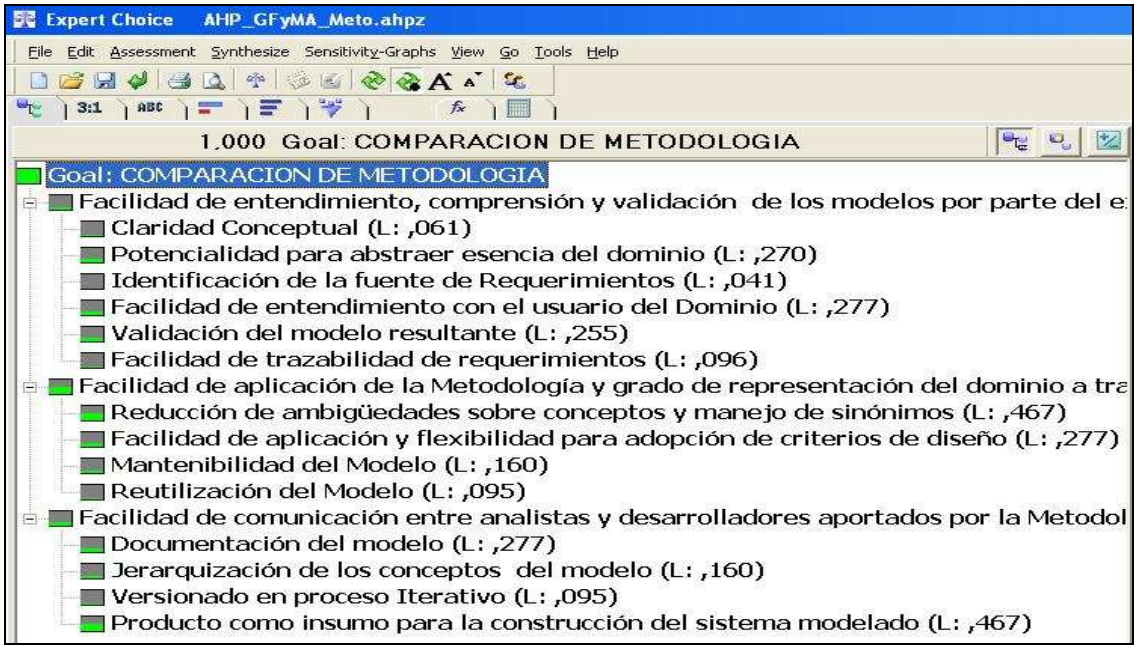
	RUP	LEL	ONTOLOGIAS
RUP		3	4
LEL			2
ONTOLOGIAS			

3.4. Producto como insumo para la construcción del sistema modelado

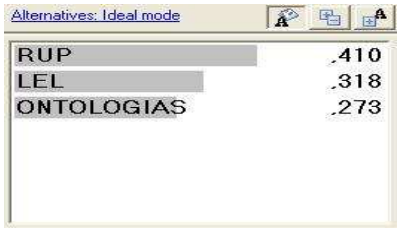
	RUP	LEL	ONTOLOGIAS
RUP		3	3
LEL			1
ONTOLOGIAS			

7.5.1.2. Resultados obtenido del proceso de síntesis

Una vez determinadas la estructura jerárquica y las matrices de comparaciones pareadas se procedió a la carga de dichos datos en el software Expert Choice 11.5, como se muestra en la pantalla 7.7., 7.8 y 7.9, para la resolución por medio del método AHP, con el objetivo de calcular las prioridades de las alternativas con respecto a la meta global.



Pantalla 7.7. Expert Choice - comparación de metodologías para AHP



Pantalla 7.8. Expert Choice - resultado numérico de la comparación de Metodologías

Pantalla 7.9. Expert Choice - resultado gráfico de la comparación de Metodologías

7.5.1.3. Metodología como mejor alternativa

A través, de la aplicación del método AHP, sobre la valoración de los criterios aplicados a las metodologías, se puede observar la alternativa con ponderación prevalente es la RUP, lo que determina que es la mejor alternativa resultante a partir del proceso de selección

7.5.2. Modelo Jerárquico de Herramientas

Se aplicará la ponderación de criterios y Modelo jerárquico detallado para la evaluación de metodologías descripto en el capítulo VI

7.5.2.1. Matrices de comparación de pares de criterios

Comparación de Criterios

	1	2	3
1		-2	2
2			3
3			

Comparación de Subcriterios

- Facilidad de uso y aprendizaje

	1.1	1.2	1.3
1.1		-2	2
1.2			3
1.3			

- Prestaciones en la actividad de modelado

	2.1	2.2	2.3	2.4	2.5
2.1		4	5	3	2
2.2			2	-2	-3
2.3				-3	-4
2.4					-2
2.5					

- Facilidad de comunicación con etapas posteriores de desarrollo

	3.1	3.2
3.1		2
3.2		

Comparación de las alternativas con respecto a cada Subcriterio

1.1. Facilidad de instalación y configuración

	BMW	Protégé	Rational
BMW		1	1
Protégé			1
Rational			

1.2. Curva de Aprendizaje de la Herramienta

	BMW	Protégé	Rational
BMW		1	2
Protégé			2
Rational			

1.3. Portabilidad

	BMW	Protégé	Rational
BMW		-2	-3
Protégé			-2
Rational			

2.1. Capturar en forma fiel y precisa la abstracción del modelo

	BMW	Protégé	
BMW		2	1
Protégé			-2
Rational			

2.2. Facilidad de introducir cambios en el diseño

	BMW	Protégé	Rational
BMW		-2	1
Protégé			2
Rational			

2.3. Soporte a proceso iterativo y manejo de versionado

	BMW	Protégé	Rational
BMW		3	1
Protégé			-3
Rational			

2.4. Visualización a través de diferentes vistas del modelo

	BMW	Protégé	Rational
BMW		1	-4
Protégé			-4
Rational			

2.5. Uso de notaciones y simbología que faciliten el entendimiento del usuario

	BMW	Protégé	Rational
BMW		-2	1
Protégé			2
Rational			

3.1. Uso de notaciones y simbología que faciliten el entendimiento del equipo de desarrollo en la construcción del modelo

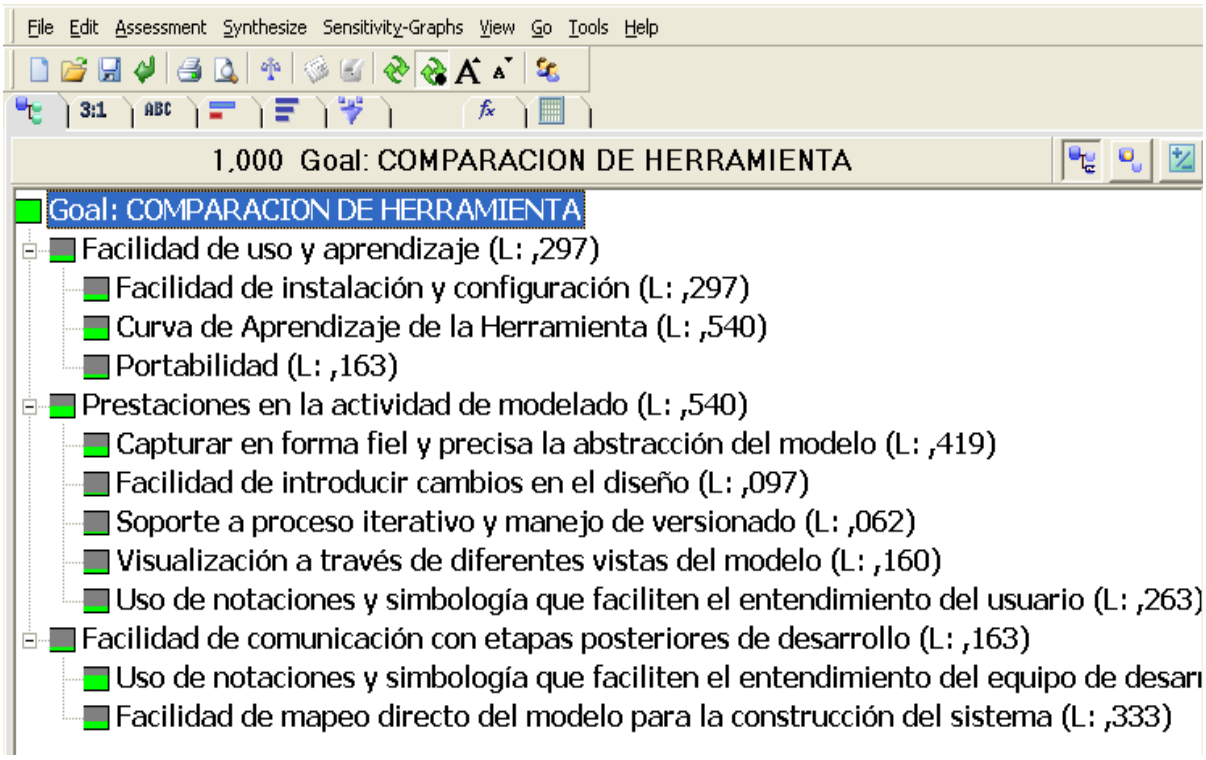
	BMW	Protégé	Rational
BMW		2	1
Protégé			2
Rational			

3.2. Facilidad de mapeo directo del modelo para la construcción del sistema

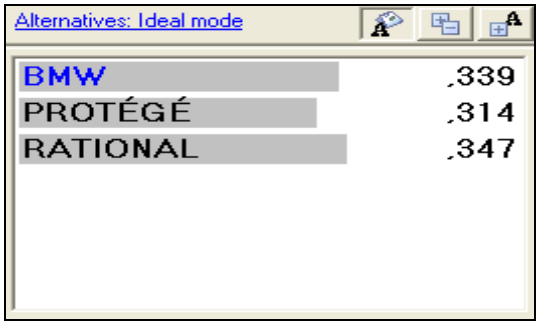
	BMW	Protégé	Rational
BMW		2	-4
Protégé			-6
Rational			

7.5.2.2. Resultados obtenido del proceso de síntesis

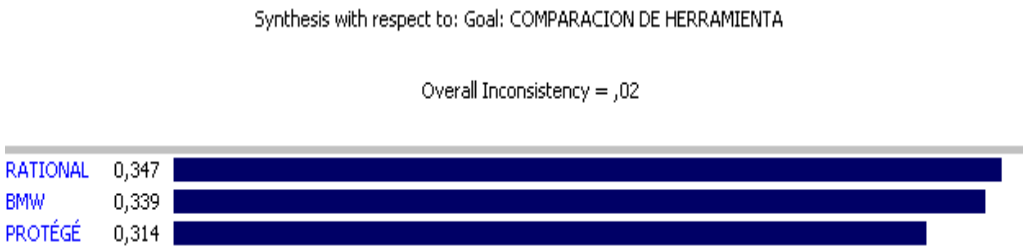
Una vez determinadas la estructura jerárquica y las matrices de comparaciones pareadas se procedió a la carga de dichos datos en el software Expert Choice 11.5, como se muestra en la pantalla 7.10., 7.11 y 7.12, para la resolución por medio del método AHP, con el objetivo de calcular las prioridades de las alternativas con respecto a la meta global.



Pantalla 7.10. Expert Choice - comparación de herramientas para AHP



Pantalla 7.11.Expert Choice - resultado numérico de la comparación de Herramientas



Pantalla 7.12.Expert Choice - resultado gráfico de la comparación de Herramientas

7.5.2.3. Herramienta como mejor alternativa

A través, de la aplicación del método AHP, sobre la valoración de los criterios aplicados a las herramientas, se puede observar la alternativa con ponderación prevalente es la Rational Rose, lo que determina que es la mejor alternativa resultante a partir del proceso de selección

7.5.3. Resumen integrado de resultados obtenidos.

A continuación se presenta una tabla que integra los resultados obtenidos mediante el proceso de Análisis Jerárquico (AHP) mediante la selección de criterios y las ponderaciones correspondientes al proceso de evaluación de las diferentes Metodologías/Herramientas aplicadas sobre el dominio bajo estudio de Gramáticas Formales y Máquinas Abstractas

Tabla 7.18 Resultados comparaciones de Gramáticas Formales y Maquinas Abstractas con AHP

RUP	*0.410*	**RATIONAL**	*0.347*
LEL	0.318	***BMW***	0.339
ONTOLOGIAS	0.273	**PROTÉGÉ**	0.314

En forma coincidente la metodología RUP y la herramienta Rational Rose han quedado posicionadas a partir de la aplicación del método de comparación Proceso de Análisis Jerárquico AHP como las de mejor perfomance para construir el modelo conceptual para la especificación de requerimientos

A su vez, también coincide la metodología LEL con su herramienta BMW en segundo lugar, y Ontologías y Protégé en tercer lugar.

CAPÍTULO 8

RESUMEN *En este capítulo se evalúan las diferentes metodologías/herramientas objetos de comparación. Se procede a establecer las conclusiones en referencia a: a) Comparación para cada una de las metodologías/herramientas de los resultados obtenidos en los diferentes dominios Sistema Docentes y Máquinas Abstractas/Gramáticas Formales. b) Comparación entre las metodologías/herramientas de acuerdo a la aplicación del método AHP seleccionado para la comparación multicriterio. c) Mantenimiento del isomorfismo en los modelos obtenidos. A partir de estas tres dimensiones de análisis se establece la conclusión general del trabajo y los futuros trabajos de Investigación.*

8.1. CONCLUSIONES

En este capítulo se presentarán las conclusiones referidas al trabajo de tesis sobre las metodologías y las herramientas que se compararán en el desarrollo de este trabajo, las cuales son las siguientes: Léxico Extendido del Lenguaje (LEL), Escenarios y Tarjetas CRC, utilizando como herramienta de descripción al Baseline Mentor Workbench (BMW); desarrollo de ontologías utilizando protégé-2000 como herramienta de modelado y edición de Ontologías; casos de uso obtenidos a partir de la metodología Rational Unified Process (RUP), con la utilización de Rational Rose.

8.2. DIMENSIONES DE ANALISIS EN EL PROCESO DE COMPARACION

Para poder efectuar el proceso de comparación lo efectuaremos a través de la valoración abordada desde tres perspectivas diferentes, como se puede ver en la Figura 8.2:

- **Aplicación de las Metodologías/Herramientas en diferentes dominios**: comparación de la valoración de los criterios seleccionados en el proceso de comparación, en los diferentes dominios Sistema Docentes y Máquinas Abstractas/Gramáticas Formales para cada una de las metodologías/herramientas objetos de estudio.
- **Valoración multicriterio con AHP sobre los criterios seleccionados:** comparación de las diferentes metodologías/herramientas de acuerdo a la aplicación del método Análisis Jerárquico de Procesos (AHP) sobre los valores obtenidos en la construcción de los modelos de aplicación.

- **Reepresentatividad de los modelos**: Grado de mantenimiento del isomorfismo de los modelos obtenidos a partir del isomorfismo existente entra Gramáticas Formales y Máquinas Abstractas.

Figura 8.2. Esquema de comparaciones

8.2.1. Conclusiones en aplicación de las metodologías/herramientas en diferentes dominios

Para efectuar la misma, se procede a realizar una comparación observacional directa sobre cada uno de los criterios seleccionados tanto para las Metodologías como las Herramientas dentro de cada una de las metodologías/herramientas objeto de estudio.

Esta comparación intenta realizar una valoración en comparación a los resultados previamente obtenidos, es decir, al ser las características del nuevo dominio de aplicación muy particulares, dominio fuertemente abstracto y cercano a las matemáticas, se intentará apreciar si las diferencias, impedimentos o particularidades de las Herramientas/Metodologías de este contexto de aplicación en relación a las apreciaciones obtenidas en la aplicación sobre un simple modelo de sistemas de información previamente realizado.

8.2.1.1. Léxico extendido del lenguaje / BMW

Aplicación de la metodología

Tabla 8.1 Valoración comparativa metodología léxico extendido del lenguaje

Metodología: Léxico Extendido del Lenguaje			
Criterio evaluado	Dominio Docentes / Cursos	Dominio Gramáticas / Máquinas	Nuevas Apreciaciones
Claridad Conceptual.	E	MB	Al crecer el dominio en tamaño y complejidad se observan dificultades en la implantación del principio de circularidad y de vocabulario mínimo
Potencialidad para abstraer esencia del dominio.	MB	B	Al crecer el dominio en tamaño y complejidad se observan dificultades en representar taxonomías de conceptos
Identificación de la fuente de Requerimientos.	B	B	
Reducción de ambigüedades sobre conceptos y manejo de sinónimos	MB	MB	.
Facilidad de aplicación y flexibilidad para adopción de criterios de diseño	B	B	
Facilidad de entendimiento con el usuario del Dominio.	E	MB	Al Crecer el Dominio las relaciones y circularidad complican el seguimiento
Mantenibilidad del Modelo	MB	MB	El modelo es mantenible a través del tiempo sin demasiada complejidad.
Reutilización del Modelo	B	B	
Documentación del modelo.	MB	MB	
Jerarquización de los conceptos del modelo	R	R	Incrementa las dificultades en las taxonomías de conceptos.
Validación del modelo resultante.	MB	MB	
Versionado en proceso Iterativo.	B	B	
Facilidad de trazabilidad de requerimientos.	MB	MB	
Producto como insumo para la construcción del sistema modelado.	R	R	

Aplicación de la herramienta

Tabla 8.2 Valoración comparativa herramienta BMW

Herramienta: Baseline Mentor Workbench			
Criterio evaluado	Dominio Docentes / Cursos	Dominio Gramáticas / Máquinas	Nuevas Apreciaciones
Facilidad de instalación y configuración.	E	E	
Curva de Aprendizaje de la Herramienta	MB	MB	
Capturar en forma fiel y precisa la abstracción del modelo.	MB	MB	

Facilidad de introducir cambios en el diseño.	B	B	
Soporte a proceso iterativo y manejo de versionado.	MB	MB	
Visualización a través de diferentes vistas del modelo.	B	R	Al incrementarse el dominio se hace más necesario contar con una interfaz gráfica
Uso de notaciones y simbología que faciliten el entendimiento del usuario.	E	MB	El conjunto de Símbolos admitidos en la descripción no presenta riqueza para describir fórmulas matemáticas
Uso de notaciones y simbología que faciliten el entendimiento del equipo de desarrollo en la construcción del modelo.	E	E	
Facilidad de mapeo directo del modelo para la construcción del sistema	B	B	
Portabilidad	B	B	

Conclusiones

En la construcción del Modelo de las Gramáticas Formales y las Máquinas Abstractas, podemos adicionar a las observaciones identificadas anteriormente las siguientes apreciaciones: Se incrementa la dificultad de implementar una taxonomía de conceptos. El poder expresivo del vocabulario disponible para utilizar no es el adecuado cuando el dominio a modelar es abstracto y cercano a las matemáticas. Para la introducción a cambios en el modelo las relaciones entre conceptos dificultan la tarea.

En referencia a la herramienta, las apreciaciones realizadas sobre el primer dominio, no se ven afectadas en gran medida con la aplicación del nuevo dominio, salvo en lo referente a que la herramienta BMW no presenta posibilidad de escribir símbolos necesarios para describir formulas matemáticas, y la necesidad de contar con una interfaz gráfica para facilitar la visualización de conceptos.

8.2.1.2. Ontologías / Protégé-2000

Aplicación de la metodología

Tabla 8.3 Valoración comparativa metodología Ontology Development 101

Metodología: Ontology Development 101			
Criterio evaluado	Dominio Docentes / Cursos	Dominio Gramáticas / Máquinas	Nuevas Apreciaciones
Claridad Conceptual.	MB	B	A medida que el dominio crece es más difícil visualizar la conceptualización del mismo.
Potencialidad para abstraer esencia del dominio.	B	B	
Identificación de la fuente de Requerimientos.	B	B	

Reducción de ambigüedades sobre conceptos y manejo de sinónimos	B	B	
Facilidad de aplicación y flexibilidad para adopción de criterios de diseño	B	B	
Facilidad de entendimiento con el usuario del Dominio.	E	E	
Mantenibilidad del Modelo	MB	MB	
Reutilización del Modelo	MB	MB	
Documentación del modelo.	MB	MB	
Jerarquización de los conceptos del modelo	E	E	
Validación del modelo resultante.	E	MB	La herramienta básica no permite realizar inferencias complejas sobre el modelo, para dar respuestas a las preguntas de competencia, sin embargo existen razonadores adicionales.
Versionado en proceso Iterativo.	R	R	
Facilidad de trazabilidad de requerimientos.	MB	MB	
Producto como insumo para la construcción del sistema modelado.	R	R	

Aplicación de la herramienta

Tabla 8.4 Valoración comparativa herramienta Protégé-2000 Ver. 1.7

Herramienta: Protégé-2000 Ver. 1.7			
Criterio evaluado	Dominio Docentes / Cursos	Dominio Gramáticas / Máquinas	Nuevas Apreciaciones
Facilidad de instalación y configuración.	E	E	
Curva de Aprendizaje de la Herramienta	MB	MB	
Capturar en forma fiel y precisa la abstracción del modelo.	B	B	
Facilidad de introducir cambios en el diseño.	MB	MB	
Soporte a proceso iterativo y manejo de versionado.	R	R	
Visualización a través de diferentes vistas del modelo.	R	R	
Uso de notaciones y simbología que faciliten el entendimiento del usuario.	MB	MB	
Uso de notaciones y simbología que faciliten el entendimiento del equipo de desarrollo en la construcción del modelo.	E	E	
Facilidad de mapeo directo del	R	R	

modelo para la construcción del sistema			
Portabilidad	MB	MB	

Conclusiones

En la construcción del Modelo de las Gramáticas Formales y las Máquinas Abstractas, sólo podemos adicionar a las observaciones identificadas anteriormente la siguiente apreciación: La gran ventaja que presenta el conjunto Metodología/herramienta que es la posibilidad de aplicar al modelo las preguntas de competencias desaparece, ya que en este nuevo dominio presenta una particularidad que es el dinamismo que presenta en el modelo el reconocimiento y la generación de cadenas

8.2.1.3. RUP/rational (rational unified process) / rational rose

Aplicación de la metodología

Tabla 8.5 Valoración comparativa metodología RUP (rational unified process)

Metodología: RUP (Rational Unified Process)			
Criterio evaluado	Dominio Docentes / Cursos	Dominio Gramáticas / Máquinas	Nuevas Apreciaciones
Claridad Conceptual.	E	MB	En dominios poco triviales o complejos, es más difícil ver claramente la representación conceptual de los elementos del dominio.
Potencialidad para abstraer esencia del dominio.	E	MB	La abstracción se vuelve complicada al momento de que el dominio se convierte en dominios no triviales. Ejemplo CU: Tipos de Gramática.
Identificación de la fuente de Requerimientos.	E	E	
Reducción de ambigüedades sobre conceptos y manejo de sinónimos	B	B	
Facilidad de aplicación y flexibilidad para adopción de criterios de diseño	B	MB	Al tratarse de un dominio complejo la aplicación de criterios de diseño se vuelve muy útil.
Facilidad de entendimiento con el usuario del Dominio.	MB	MB	
Mantenibilidad del Modelo	E	E	
Reutilización del Modelo	MB	MB	
Documentación del modelo.	E	MB	Posibilita una documentación detallada, pero se debe conocer sobre la simbología específica del proceso, y la mayor documentación resulta de difícil manejo.
Jerarquización de los conceptos del modelo	E	E	
Validación del modelo	MB	MB	

resultante.			
Versionado en proceso Iterativo.	E	E	
Facilidad de trazabilidad de requerimientos.	E	E	
Producto como insumo para la construcción del sistema modelado.	MB	MB	

Aplicación de la herramienta

Tabla 8.6 Valoración comparativa herramienta IBM Rational Rose 7.0

Herramienta: IBM Rational Rose 7.0			
Criterio evaluado	Dominio Docentes / Cursos	Dominio Gramáticas / Máquinas	Nuevas Apreciaciones
Facilidad de instalación y configuración.	E	E	
Curva de Aprendizaje de la Herramienta	B	B	
Capturar en forma fiel y precisa la abstracción del modelo.	MB	MB	
Facilidad de introducir cambios en el diseño.	B	B	
Soporte a proceso iterativo y manejo de versionado.	MB	MB	
Visualización a través de diferentes vistas del modelo.	E	E	
Uso de notaciones y simbología que faciliten el entendimiento del usuario.	MB	B	En dominios poco triviales, abstractos o cercanos a las Matemáticas es difícil representarlos
Uso de notaciones y simbología que faciliten el entendimiento del equipo de desarrollo en la construcción del modelo.	MB	MB	
Facilidad de mapeo directo del modelo para la construcción del sistema	E	E	
Portabilidad	E	E	

Conclusiones

La aplicación de este conjunto de metodologías/herramientas la construcción del Modelo de las Gramáticas Formales y las Máquinas Abstractas, sólo podemos adicionar a las observaciones identificadas no es necesario adicionar nuevas apreciaciones, pero reforzar lo ya vertido con respecto a la extensa simbología y generación de documentación y reafirmar las potencialidades de aplicar estrategias de diseño tendientes a facilitar la futura implementación del modelo.

8.2.2. Conclusiones sobre la valoración multicriterio con AHP sobre los criterios seleccionados

En esta sección se analizarán y compararán los resultados arrojados mediante el proceso de síntesis del método de evaluación utilizando para el análisis multicriterio sobre los criterios identificados para la valoración tanto de las metodologías y las herramientas. La metodologías de análisis fue realizada sobre ambos dominios, Sistema Docentes y el de Gramáticas Formales y Máquinas Abstractas con el método de proceso de Análisis Jerárquico (AHP) con la utilización de la herramienta Expert choice versión 11.5.

8.2.2.1. Metodologías

Tabla 8.7 Comparativa de síntesis AHP sobre metodologías

Metodologías		
	Sistema Docentes	Gramáticas Formales y Máquinas Abstractas
RUP (Rational Unified Process)	*0.396*	*0.410*
Client Oriented Requirements Baseline	0.319	0.318
Ontology Development 101	0.285	0.273

La variación numérica que se produce en la valoración en el proceso de síntesis es menor y no varía la ponderación para la selección de la misma

8.2.2.2. Herramientas

Tabla 8.8 Comparativa de síntesis AHP sobre herramientas

Herramientas		
	Sistema Docentes	Gramáticas Formales y Máquinas Abstractas
Rational Rose	*0.350*	*0.347*
Baseline Mentor Workbench	0.350	0.339
Protégé	0.300	0.314

La variación numérica que se produce sobre el dominio de las GF y MA nos ayuda a obtener una jerarquía en las herramientas ya que la misma no había podido ser apreciada en el Sistemas Docentes.

8.2.3. Conclusiones sobre la representatividad del modelo

Para poder realizar este análisis, no apoyamos en la particularidad que presenta el dominio seleccionado bajo estudio de Gramáticas Formales y Máquinas Abstractas, ya que el mismo presenta la particularidad de tener propiedades isomorfas.

En la construcción de los modelos conceptuales, para todas las metodologías/herramientas se evidencia que los modelos obtenidos conservan la propiedad

isomorfa del dominio, incluso con la posibilidad de establecer distintas alternativas de modelado, con lo cual todas las metodologías/herramientas resultan aptas para la actividad de modelado.

8.3. CONCLUSIÓN GENERAL

Los modelos obtenidos con las diferentes Metodologías/Herramientas sobre el dominio de Gramáticas Formales y Máquinas Abstractas presentan diferencias en la valoración de criterios más marcadas en las metodologías que en las herramientas, en referencia a las identificadas en el Modelado del sistema Docentes. Estas diferencias se introducen dada la particularidad que presenta el dominio de GF y MA ya que es fuertemente abstracto y cercano a las matemáticas, sobre todo en la aplicación de la metodologías ya que salvo en pocos ítems los criterios seleccionados para valorar las herramientas se ven influenciaos por el nuevo dominio de aplicación. Estas diferencias y corrección en las valoraciones se dan en diferentes criterios, en algunos casos disminuyendo la valoración o en algunos casos potenciando la misma, pero al ponderar los mismos mediante el proceso de Síntesis obtenidos a través de la aplicación del método de Análisis Jerárquico de Procesos AHP se obtiene idéntico orden de de prioridades tanto en las metodologías y herramientas en ambos dominios.

En lo que respecta a las Metodologías resulta a través del proceso AHP de multicriterio que el ranking de alternativas evaluadas de acuerdo a los criterios seleccionados es RUP con un porcentaje cercano al 40 % siguiendo LEL con un 32 % y por último las ONTOLOGIAS con un 28 %

Si bien RUP presenta un lenguaje extenso con múltiples símbolos de representación, los cuales tomados todos en su conjunto complican el entendimiento por parte de los diferentes actores, sobre todo con los expertos del dominio que tendrán que validar el modelo. Afortunadamente para la mayoría de los modelos, es posible seleccionar un conjunto simplificado de esto símbolos y convenciones del lenguaje UML, presentando diferentes vistas del modelo. La documentación generada es extensa pero resulta fácil de mantener con el uso de la herramienta apropiada, en esta caso Rational Rose. El modelo obtenido presenta un mapeo prácticamente directo en las futuras etapas de desarrollo y construcción del modelo.

Con respecto a LEL resulta muy útil cuando el equipo de especificación para la construcción del modelo, no tiene experiencia y conocimiento del dominio a modelar, partiendo de los principios básicos de circularidad y vocabulario mínimo es posible lograr un cabal entendimiento, dado que las fuentes (expertos del dominio) utilizan múltiples sinónimos dependiendo del contexto organizacional. BMW da soporte para mantener la consistencia y versionado. La descripción de escenarios, es vital para evidenciar la relación de conceptos y la generación de fichas CRC. El modelo resultante no es directamente maleable y la no

existencia de un entorno simbólico y visual dificulta el entendimiento con las posteriores etapas en la construcción del sistema.

Por otro lado, las Ontologías permiten a través de una pequeña cantidad de simbologías, modelar los conceptos del dominio y lograr un entendimiento común tanto con los usuarios que validarán el modelo, como en las futuras implementaciones, no llegando a un mapeo directo, ya que en alguna medida, se estará condicionado con la priorización del modelado del concepto y a dar respuesta a las preguntas de competencia, se pierde la posibilidad de todos sus beneficios cuando el dominio a modelar tiene conceptos asociados de dinamismo o ejecución en tiempo real.

En referencia a la evaluación de las herramientas seleccionadas de acuerdo a los criterios valorados y mediante el proceso de síntesis del método de evaluación multicriterio AHP, se refleja que Rational Rose se perfila como la herramienta que mejor se ajusta a las actividades de Modelado con un porcentaje cercano al 35 % BMW con un 34 % y Protégé con un 31 %

En resumen todas las metodologías/herramientas analizadas cumplen con el objetivo de capturar fielmente los requerimientos en la actividad de construcción del modelo conceptual. En sistemas triviales, en donde esté consolidado el equipo de relevamiento, es recomendable la utilización de RUP/Rational, que provee buen nivel de documentación asociado aunque de compleja validación por parte de los expertos del dominio. En dominios en donde se necesita explorar y validar los conceptos, o por que el dominio es complejo y/o con múltiples significados de los diferentes actores y en diferentes áreas de aplicación es recomendable la utilización de LEL/BMW, estando ausente en la herramienta un entorno visual para otorgar diferentes vistas del modelo y poder asignar a los objetos identificados caracterización de los atributos. Por otra parte las Ontologías/Protégé proveen un marco simple y formal en donde la base de conocimiento obtenida, puede ser reutilizada, ampliada y adicionado de especificidad a partir de la misma y por lo tanto reutilizada y ampliada.

8.4. FUTUROS TRABAJOS

A partir de los resultados obtenidos en el presente trabajo de tesis se abren en forma directa dos posibilidades para la continuación con trabajos de investigación.

Por un lado, el mismo puede ser extendido, a como los modelos obtenidos pueden ser utilizados para ser optimizados utilizando una nueva forma de descomponer los sistemas mediante la orientación a aspectos de manera de lograr una mayor perfomance en el momento de la construcción de los modelos

Y por otro, a ver como pueden ser extendidos los modelos obtenidos a partir de las metodologías Client Oriented Requirements Baseline, y de Ontologías para lograr la automatización de la gestión de los procesos de desarrollo de software, para la obtención de un Workflow que defina los procesos de negocio automatizados, en forma similar a la extensión que se realiza sobre UML a través de SPEM.

Además se podría profundizar en el estudio comparativo de éstas metodologías y herramientas, ampliando este trabajo para estudiar las mismas metodologías y herramientas de acuerdo a otras características como calidad, lo que involucra estudio de métricas, esquemas de trazabilidad, facilidad y confiabilidad de los esquemas de validación y verificación. De la misma forma se podría utilizar el mismo proceso de comparación pero abarcando otras metodologías y herramientas.

También se podría extrapolar este estudio a diferentes dominios, como por ejemplo el de tiempo real, de sistemas de misión crítica, y evaluar si los resultados son similares a los obtenidos para cada una de las metodologías y herramientas evaluadas en el presente trabajo.

BIBLIOGRAFÍA

Ackoff 1974] R. Ackoff, *Rediseñando el futuro.* Willey, 1974

[Andriano 2006] N. Andriano, *Comparación del Proceso de Elicitación de Requerimientos en el desarrollo de Software a Medida y Empaquetado. Propuesta de métricas para la elicitación.* Tesis presentada a la Facultad de Informática de la Universidad Nacional de La Plata, Agosto, 2006

[ANSI 1975] *Study Group on Database Management Systems, Interim Report*. FDT, Bulletin of ACM, SIGMOD (7)2, pp. 1-140.

[ANSI/IEEE 1984] ANSI/IEEE *Standard 830-1984: Standard for software Requirements Specifications*. The institute of Electrical and Electronic Engineers, New York, 1984.

[ANSI/IEEE 1990] ANSI/IEEE *Standard 610-1990: IEEE Standard glossary of software engineering terminology.* The institute of Electrical and Electronic Engineers, New York, 1990.

[Antonelli 2002] L. Antonelli, A. Oliveros, *Fuentes utilizadas por desarrolladores de software en Argentina para elicitar requerimientos*. V Workshop on Requirements Engineering, WER, 2002 Disponible en: wer.inf.puc-rio.br

[Antonelli 2003] L. Antonelli, A. Oliveros, G. Rossi, *Traceability en la Elicitación y Especificación de Requerimientos*. Tesis de Maestría Universidad Nacional de la Plata, http://postgrado.info.unlp.edu.ar/Carrera/Magister/Ingenieria%20de%20Software/Tesis/Antonelli.pdf ; Fecha consulta: 01/12/06

[Bernaras 1996] A. Bernaras, I. Laresgoiti, J. Corera, *Building and Reusing Ontologies for Electrical Network Applications*. In Proceedings of the European Conference on Artificial Intelligence, 1996 (ECAI96):298-302.

[Berners 2001] T. Berners, J. L. Hendler, O. Lassila, *The Semantic Web*. Scientific American, Mayo 2001.

[Booch 1998] G. Booch, J. Rumbaugh, I. Jacobson, *The Unified Language User Guide.* Addisson Wesley, 1998.

[Booch 2006] G. Booch, J. Rumbaugh, I. Jacobson, *El Lenguaje unificado de modelado.* 2da edición, Addisson Wesley, 2006.

[Boehm 2001] B. Boehm, V.R. Basili, *Software defect reduction top 10 list.* IEEE Computer, 01/01/01

[Castells 2002] P. Castells, *Aplicación de Técnicas de la Web Semántica*. Disponible en: http://giig.ugr.es/~mgea/coline02/Articulos/pcastells.pdf
Fecha consulta: 25/09/06.

[Cockburb 1999] A. Cockburn, *Responsibility-based Modeling.* Technical Memo Hat TR-99.02 Disponible en: http://members/aol.com/humansandt/techniques/responsabilities.html.
Fecha Consulta: 01/10/06.

[Davis 1993] A. Davis, *Software requirements Object, functions and states.* Pretince Hall international Inc, 1993.

[Duran 2002] A.Durán, B.Bernández, *Metodología para la Elicitación de Requisitos de Software.* Universidad de Sevilla, 2002.

[Fernández 1997] M. Fernández, A. Gómez-Pérez, N. Juristo, *METHONTOLOGY: From Ontological Art to Ontological Engineering*. Workshop on Ontological Engineering, Spring Symposium Series de la AAAI (American Association for Artificial Intelligence): 33-40, Stanford, USA, 1997.

[Fernández 1999] M. Fernández-López, A. Gómez-Pérez, A. Pazos-Sierra, J. Pazos-Sierra, *Building a Chemical Ontology Using METHONTOLOGY and the Ontology Design Environment*. IEEE Intelligent Systems & their applications, January/February Pages 37-46, 1999

[Fernández 2000] M. Fernández-López, A. Gómez Pérez, M. D. Rojas Amaya, *Ontologies crossed life cycles*. Workshop on Knowledge Acquisition, Modelling and Management (EKAW):65-79, Editor Springer Verlag, Jean Les Pins, Francia, 2000.

[Gil 2003] G. Gil, A. Oliveros, G. Rossi. *Herramienta para implementar LEL y Escenarios (TILS)*. Tesis de Maestría Universidad Nacional de la Plata. Fecha consulta: 01/12/06, disponible en: http://postgrado.info.unlp.edu.ar/Carrera/Magister/Ingenieria%20de%20Software/Tesis/Gil.pdf

[Goguen 1994] J. Goguen, *Requeriments Engineering as the Recontiliation of social and Technical Issue*. Academic Press, 1994

[Griethuysen 1982] J.J. van Griethuysen, *Concepts and terminology for the Conceptuals Schema and the Information Base.* ISO Report ISO/TC97/SC95/N695, 1982.

[Gruber 1993] T. R. Gruber, *A Translation Approach to portable Ontology Specifications*. Knowledge Acquisition, 5(2), pp. 199-200, 1993.
Disponible en: http://tomgruber.org/writing/ontolingua-kaj-1993.pdf. Fecha consulta: 20/02/07

[Gruninger 1995] M. Gruninger, M.S. Fox, *The logic of enterprise modeling*. In J. Brown & D.O. Sullivan, Eds, Reenginering the Enterprise: 83-98. London: Chapman & Hall. 1995.

[Guarino 1995] N. Guarino, *Formal Ontology, Conceptual Analysis and Knowledge Representation*. 1995 Disponible en web: http://cs.umbc.edu/771/papers/FormOntKR.pdf Fecha consulta: 15/02/08

[Guarino 1998] N. Guarino, *Formal Ontology and Information System.* Internacional Conference on Formal Ontology in Information Systems, Trento, Italy, pp. 3-15-

[Hirschheim 1995] R. Hirschheim, H.K. Klein, K. Lyytinen, *Information Systems Development and data Modeling Conceptual and Philosophical Foundations.* Cambridge University Press, 1995.

[Hurtado 2007] T. Hurtado, G. Bruno, *El Proceso de Análisis Jerárquico (AHP) como Herramienta para la. Toma de Decisiones en la Selección de Proveedores.* Disponible en web: sisbib.unmsm.edu.pe/bibvirtualdata/Tesis/Basic/...hg/ Fecha consulta: 15/02/09

[IEEE 1999] *IEEE Standard for Conceptual Modeling Language Syntax and Semantic for IDEF1X97 (IDEF Object)*. Software Committee of the IEEE Computer Society, IEEE STD 1320.2, 1998.

[Insfrán 2002] E. Insfrán, I. Díaz, M. Burbano, *Modelado de Requisitos para la Obtención de esquemas conceptuales*. Disponible en: http://www.dsic.upv.es/~einsfran/papers/39-ideas2002.pdf Fecha Consulta: 02/10/06

[Insfrán 2002b] E. Insfrán, E. Tejadillos, S. Marti, M. Burbano, *Transformación de Especificación de requisitos en esquemas conceptuales usando Diagramas de Interacción*. Disponible en: www.inf.puc-rio.br/~wer02/zip/Transformacion_Espec(7).pdf Fecha Consulta: 04/12/07

[Jacobson 1995] I. Jacobson, M. Erikson, A. Jacobson, *The Object Adventage: Business Process Reengineering With Object Technology.* Addisson Wesley, Object Tecnology series, 1995.

[Jackson 1995] M. Jackson, *Software Requirements & Specifications*. Addisson Wesley, 1995.

[Kaplan 2000] G. N. Kaplan, G. D. S. Hadad, J. H. Doorn, J. C. Sampaio do Prado Leite, *Inspección del Léxico Extendido del Lenguaje.* Disponible en: http://wer.inf.puc-rio.br/WERpapers/artigos/artigos_WER00/kaplan.pdf Fecha consulta: 01/12/06

[Kotonya 1998] G. Kotonya, I. Sommerville, *Requirements Engineering, Process and Techniques*. John Wiley & sons, 1998.

[Kruchten 2000] P. Kruchten, *The Rational Unified Process: An Introduction.* Addison Wesley, 2000

[Latorres 2002] E. Latorres, J. Echagüe, *Reuso de reglas de negocio: Una experiencia en el reuso de ontologías*. Tesis de Maestría, Disponible en web, Fecha Consulta 01/11/06.

[Leite 1997] J.C.S. Leite, G. Rossi, *Enhancing a Requeriments Baseline with Scenarios.* Proceedings of RE 97: IEEE Third international Requirements Engineering Symposium, IEEE Computer Society Press, 1997 pp. 44-53.

[Lenat 1990] D.B. Lenat., R.V. Guha, *Building Large Knowledge Based Systems*. Representation and Inference in the CYC Project, Addison-Wesley, Reading, MA, 1990.

[Leonardi 2001] C. Leonardi, J.C.S. Leite, G. Rossi, *Una estrategia de Modelado Conceptual de Objetos, basada en Modelos de requisitos en lenguaje natural*. Tesis de Maestría Universidad Nacional de la Plata, Fecha consulta: 01/12/06, disponible en: http://postgrado.info.unlp.edu.ar/Carrera/Magister/Ingenieria%20de%20Software/Tesis/Leonardi.pdf

[Leue 2000] Dr. S. Leue, *Metrics for Software Requirements Specifications (SCR).* Material del Curso E&CE, Disponible en: http://watfast.uwaterloo.ca/~salah/751_R.HTM Fecha consulta: 01/09/00.

[Loucopoulos 1995] P. Loucopoulos, V. Karakostas, *System Requirements Engineering*. McGraw-Hill, 1995.

[McGuinness 2000] D.L. McGuinness, R. Fikes, J.Rice, S. Wilder. "An Environment
For Merging and Testing Large Ontologies". Principles of Knowledge Representación and Reasoning: Proceedings of the Seventh Internacional Conference (KR2000). A.G.Cohn, F.Giunchiglia, B. Selman, editors. San Francisco, CA, Morgan Kaufmann Publishers.

[Mizuno 1993] Y. Mizuno, *Software Quality Improvement.* IEEE Computer, Vol. 16 Nro. 3 March 1993, pp 66-72.

[Muñoz 2002] L. S. Muñoz, *Representación de Ontologías en la Web Semántica*. Disponible en: http://www.inf.ufrgs.br/~clesio/cmp151/cmp15120021/artigo_lydia.pdf, Fecha Consulta: 25/0906.

[Moreira 2003] A. Moreira, *Tesauros e Ontologias: Estudo de Definições Presentes na Literatura das Áreas das Ciências da Computação e da Informação, Utilizando-se o Método Analítico Sintético*. Fecha Consulta: 20/02/08, Disponible en:

http://dspace.lcc.ufmg.br/dspace/bitstream/1843/LHLS-69UQKU/1/mestrado___alexandra_moreira.pdf

[Musen 1992] M.A. Musen, *Dimensions of knowledge sharing and reuse*. Computers and Biomedical Research 25: 435-467.

[Noy 2005] N. F. Noy, D. L. McGuinness, *Ontology Development 101: A Guide to Creating Your First Ontology*. September 19, 2005 Disponible en: http://www.ksl.stanford.edu/people/dlm/papers/ontology-tutorial-noy-mcguinness-abstract.html Fecha Consulta 18/0906.

[Olsina 1999] L. Olsina, *Metodología Cuantitativa para la Evaluación y Comparación de la Calidad de Sitios Web.* Tesis Doctoral en la Facultad de Ciencias Exactas, Universidad Nacional de La Plata, Argentina, Noviembre, 1999.

[Pinto 1999] H. S. Pinto, A. Gomez-Perez, J. P. Martins, *Some Issues on Ontology Integration.* Proceedings of the IJCAII99`s Workshop on Ontologies and problem solving methods: Lessons Learned and Future Trends.

[Poli 2001] R. Poli, *Ontological methodology.* Int. J. Human-Computer Studies 56, 639–664 doi:10.1006/ijhc.1003 Disponible en: http://www.unitn.it/events/do/download/Poli_Ontological_Methodology.pdf

[Pressman 1993] R.S. Pressman. *Ingeniería del Software: Un enfoque práctico*. McGraw-Hill, Madrid, 1993.

[Proyecto RODA] *Red de conocimiento descentralizado a través de anotaciones*, Disponible en: http://roda.ibit.org/ Fecha Consulta: 13/09/06

[Ramos 2007] E. Ramos, H. Nuñez, *Ontologías: Componentes, metodologías, lenguajes, herramientas y aplicaciones* Disponible en: http://dircompucv.ciens.ucv.ve/Documentos/RT-2007-12.pdf/ Fecha consulta: 02/02/09

[Rational 2009] Sitio Web de Rational. Disponible en: http://www-01.ibm.com/software/rational/

[Robertson 1997] J. Robertson, S. Robertson, *Requirements: Made to Measure.* American Programmer, Volume X, No. 8, Agosto, 1997.

[Rumbaugh 1991] J. Rumbaugh, M. Blaha, *Object-Oriented Modeling and Design.* Prentice Hall, 1991.

[Rumbaugh 1999] J. Rumbaugh, I. Jacobson, G.Booch, *The Unified Modeling Language Reference.* Addisson Wesley, 1999.

[Sánchez 2005] D. M. Sánchez, J. M. Cavero, E. Marcos, *Ontologías y MDA: una revisión de la literatura.* Disponible en: http://ftp.informatik.rwth-aachen.de/Publications/CEUR-WS/Vol-157/paper03.pdf Fecha consulta: 15/09/06

[Saaty 1980] T. Saaty, *The analytic hierarchy process*. J. Wiley, New York, 1980.

[Sesé 2006] F. Sesé Muniátegui, Tesis Doctoral: *Propuesta de un método de validación de esquemas conceptuales y análisis comparativo de la noción de información en los métodos de desarrollo de Sistemas de información* Disponible en: www.tesisenxarxa.net/TDX-0517107-131929/ Fecha consulta: 20/05/08

[Shamsfard 2004] M. Shamsfard, A. Barforoush, *The state of the art in ontology learning: a framework for comparison*. The Knowledge Engineering Review, Vol 18:4, p 293-316, Cambridge University Press.

[Shanks 2003] G. Shanks, E. Tansley, R. Weber, *Using Ontology to Validate Conceptual Model.* Communications of the ACM, (46) 10, pp. 85-89, 2003.

[Sommerville 1995] I. Sommerville, *Software Engineering.* Addisson Wesley, 1995.

[Sommerville 1997] I. Sommerville, P. Sawyer, *Requirements Engineering: A Good Practice Guide*. Computing Department, Lancaster University, John Willey & Sons Ltd. ISBN 0 471 974444 7, 1997.

[Sommerville 1999] I. Sommerville, P. Sawyer, *Requirements Engineering: A Good Practice Guide.* Lancaster University, Chichester, John Willey & Sons, 1999.

[Sommerville 2005] I. Sommerville, *Ingeniería del Software.* ISBN 9788478290741, Pearson Educación.

[Swartout 1997] B. Swartout, R. Patil, K. Knight, T. Russ, *Toward distributed use of large-scale ontologies*. In AAAI-97 Spring Symposium Series on Ontological Engineering, 1997.

[Torres 2006] A. Torres, *Capitulo 1: Simulación Discreta. Introducción a la Simulación*. Fecha de consulta: 01/06/08. Disponible en:
http://www.virtual.unal.edu.co/cursos/sedes/manizales/4060010/lecciones/Capitulo1/riesgos.htm

[Uschold 1995] M. Uschold, M. King, *Towards a Methodology for Building Ontologies*. Workshop on Basic Ontological Issues in Knowledge Sharing, 1995.

[Uschold 1996] M. Uschold, M. Groninger, *Ontologies: Principles, Methods and Applications*. Knowledge, Engineering Review, Vol. 2, 1996.

[Verrijn-Stuart 2001] A. Verrijn-Stuart, *A Framework of Information System Concepts*. Revised FRISCO Report (Draft), International Federation for Information Processing, 2001.

[Villela 2004] M. L. Bento Villela, A. Paiva Oliveira, J. L. Braga, *Modelagem Ontológica no Apoio à Modelagem Conceitual.* 18º Simpósio Brasileiro de Engenharia de Software, pp 241-256.
Disponible web: http://www.lbd.dcc.ufmg.br:8080/colecoes/sbes/2004/017.pdf, Fecha consulta: 15/02/08

[Wand 1998] Y. Wand , R. Weber, *An Ontological Analysis of same Fundamental Information Systems Concepts.* 9th International Conference on Information Systems, Minneapolis, Minnesota, 1998.

[Wiegers 1999] K.E. Wiegers, *Software Requirements.* Microsoft Press, Redmond, 1999.

[Wiegers 1999b] K.E. Wiegers, *Writing Quality Requirements – Process Impact*. Mayo, 1999.
Disponible en: http://www.processimpact.com/articles/qualreqs.pdf Fecha de Consulta: 01/03/08.

[Wirfs-Brock 1995] R. Wirfs-Brock, *Designing Objects and their interactions.* En Scenarios Based Design edit by John Carol, John Willey & Sons, Inc. 1995 pp. 337-359.

[W3C (World Wide Web Consortium)]
Disponible en http://www.w3.org/ Fecha consulta: 13/09/06.

ANEXOS

ANEXO I: PLANILLA DE VALORACIÓN DE CRITERIOS SELECCIONADOS

Es el instrumento que será utilizado, en el proceso de comparación de las Herramientas/Metodologías.

Valoración Metodologías/Herramientas	
Metodología	
Herramienta	

Metodología		
Criterio a evaluar	Valoración	Observaciones
Claridad Conceptual.		
Potencialidad para abstraer esencia del dominio.		
Identificación de la fuente		
Reducción de ambigüedades sobre conceptos y manejo de sinónimos		
Facilidad de aplicación y flexibilidad para adopción de criterios de diseño		
Facilidad de entendimiento con el usuario del Dominio.		
Mantenibilidad del Modelo		
Reutilización del Modelo		
Documentación del modelo.		
Jerarquización de los requerimientos del modelo		
Validación del modelo resultante.		
Versionado en proceso Iterativo.		
Facilidad de trazabilidad de requerimientos.		
Producto como insumo para la construcción del sistema modelado.		

Herramienta		
Criterio a evaluar	Valoración	Observaciones
Facilidad de instalación y configuración.		
Curva de Aprendizaje de la Herramienta		
Capturar en forma fiel y precisa la abstracción del modelo.		
Facilidad de introducir cambios en el diseño.		
Soporte a proceso iterativo y manejo de versionado.		
Visualización a través de diferentes vistas del modelo.		
Uso de notaciones y simbología que faciliten el entendimiento		

del usuario.		
Uso de notaciones y simbología que faciliten el entendimiento del equipo de desarrollo en la construcción del modelo.		
Facilidad de mapeo directo del modelo para la construcción del sistema		
Portabilidad		

ANEXO II: LEL/ESCENARIOS/CRC - APLICACIÓN AL MODELO DE DOCENTES

1. Construcción del LEL.
2. Construcción de los Escenarios a partir del LEL.
3. A partir del LEL y escenarios se construyen las fichas CRC

1. Proceso de construcción del LEL

1.1. Generación de la lista de símbolos

Son los detectados al comienzo del proceso, los cuales son ampliados en las sucesivas iteraciones:

Persona; Docente, Alumno, Cursos, Año dictado; Tipo Docente (Profesor; Auxiliar Docente) Categoría Docentes (Titular, Asociado, Adjunto) Categoría Auxiliares (JTP; Ayte 1ra/ Ayte Graduado; Ayte2da/Ayte Alumno); Año Ingreso Docente; Año Ingreso Alumno; Cantidad de cursos asignados; Cantidad posibles de cursos; Creación de curso; asignar docentes y auxiliares a un curso; asignación de alumnos a cursos.

1.2. Clasificación de los Símbolos

Todos los símbolos deberán quedar categorizados en alguna de las siguientes categorías (Sujeto / Objeto / Verbo / Estado)

- Sujeto: Elemento activo dentro del dominio que realiza acciones utilizando objetos. El sujeto (persona, maquina, dispositivo) puede llegar a pasar por distintos estados.
- Objeto: Elemento pasivo con los cuales se realizan acciones que puede pasar por distintos estados.
- Verbo: Acción que realiza un sujeto, servicio que brinda un objeto, desencadenante para pasar de un estado a otro.
- Estado: Situación en la que se encuentra un sujeto o un objeto

Símbolo	*Clasificación*
Persona	Objeto
Administrador	Sujeto
Docente	Objeto
Alumno	Sujeto
Curso	Objeto
Tipo Docente	Objeto
Profesor	Sujeto

Auxiliar Docente	Sujeto
Categoría Profesor	Objeto
Titular	Estado
Asociado	Estado
Adjunto	Estado
Categoría Auxiliares	Objeto
JTP	Estado
Ayte 1ra/ Ayte Graduado	Estado
Ayte2da/Ayte Alumno	Sujeto
Cantidad posibles de cursos	Objeto
Cantidad de cursos asignados	Objeto
Creación de curso.	Verbo
Asignación de Profesores en cursos.	Verbo
Asignación de Auxiliares Docentes en cursos.	Verbo
Asignación de Ayte Alumno en cursos.	Verbo
Inscripción a cursado.	Verbo
Legajo Docente	Objeto
Año Ingreso Docente	Objeto
Legajo Alumno	Objeto
Año Ingreso Alumno	Objeto
Cantidad Cursos Aprobados	Objeto
Alumnos inscriptos en curso	Objeto
Código de curso	Objeto
Nombre curso	Objeto
Año de dictado	Objeto
Cantidad de Alumnos inscriptos	Objeto
DNI	Objeto
Apellido y Nombre	Objeto
Edad	Objeto
Sexo	Objeto
Dictado de Cursos	Verbo

1.3. Descripción de los símbolos

Sujeto	*Nociones:* describen quien es el sujeto.
	Impactos: registran acciones ejecutadas por el sujeto

Sujetos		
Entrada	Nociones	Impactos
Alumno	Es una Persona Puede actuar de Auxiliar Docente como Ayte Alumno Su definición incluye Legajo Alumno , Año Ingreso Alumno , Cantidad Cursos Aprobados	Inscripción a cursado en Curso. Desempeñarse como Ayte Alumno en Cursos

Profesor	Es un Tipo Docente del tipo Profesor que realiza Dictado de cursos Puede tener Categoría Profesor : Titular ; Asociado ; Adjunto	Dictado de Cursos en Curso
Auxiliar Docente	Es un Tipo Docente del tipo Auxiliar Docente que colabora en Dictado de cursos Puede Tener Categoría Auxiliares : JTP ; Ayte Graduado	Colabora en Dictado de Cursos en Curso
Ayte Alumno / Ayte 2da	Es un Tipo Docente del tipo Auxiliar Docente. que colabora en Dictado de cursos Es un Alumno que se puede desempeñar como Auxiliar Docente Se le asigna además el Legajo Alumno	Colabora en Dictado de Cursos en Curso
Administrador	Es quien coordina las actividades académicas	Es quién realiza la Creación de curso. Realiza la asignación de Profesor a Curso mediante la Asignación Profesores en cursos. Realiza la asignación de Auxiliares Docentes a Curso mediante la Asignación Auxiliares Docentes en cursos. Realiza la asignación de Ayte Alumno a Curso mediante la Asignación Ayte Alumno en cursos. Realiza la Inscripción a cursado de Alumno en Curso

Objeto	*Nociones:* definen al objeto e identifica a otros términos con los cuales el objeto tiene algún tipo de relación.
	Impactos: describen las acciones que pueden ser aplicadas al objeto.

Objetos		
Entrada	Nociones	Impactos
Curso	Es donde se asignarán Profesor, Auxiliar Docente, y eventualmente uno o dos Ayte Alumno donde se efectuará el dictado de cursos. Se realizará la inscripción a cursado los Alumno. Queda definido mediante Código de curso , nombre curso y año dictado	Asignación Profesores en cursos. Asignación Auxiliares docentes en cursos. Asignación Ayte Alumno en cursos. Inscripción a cursado de Alumno.

	Los Alumno inscriptos de detallan en la lista de Alumnos inscriptos a cursado	
Docente	Es una Persona que puede ser Tipo Docente Profesor o Auxiliar Docente Su definición incluye Legajo Docente y Año Ingreso Docente Cantidad de cursos asignados	Puede ser Tipo Docente Profesor o Auxiliar Docente
Persona	Su definición incluye DNI , Apellido y Nombre , Edad y Sexo	Persona física que puede ser Docente o Alumno
Cantidad posibles de cursos	Es la cantidad de cursos a las que puede ser asignado un profesor o auxiliar docente	Limita la cantidad de Curso a asignar Profesores y Auxiliares Docentes no pueden superar los 3 Curso. Un Ayte Alumno solo 1 Curso
Cantidad de cursos asignados	Es la cantidad de Curso efectivos en la que se encuentra asignado un Profesor o Auxiliar Docente	Registra la cantidad de Curso asignados
Código de curso	Es el código asignado a un Curso	Asignar valor que corresponda
Nombre curso	Es el Nombre asignado a un Curso	Asignar valor que corresponda
Año de dictado	Es el año de dictado de un Curso	Asignar valor que corresponda
Cantidad de Alumnos inscriptos	Es la cantidad de Alumno inscriptos en el Curso	Incrementarse en uno por Alumno que se inscriba en el Curso No puede superar los 30 Alumno
Legajo Docente	Es el número de legajo con el que se identifica a un Docente	Se asigna un valor incremental
Año Ingreso Docente	Registro del año de ingreso	Registra el año de ingreso
Legajo Alumno	Es el número de legajo con el que se identifica a un Alumno	Se asigna un valor incremental
Año Ingreso Alumno	Registro del año de ingreso de un Alumno	Registra el año de ingreso
Cantidad Cursos Aprobados	Cantidad de Curso aprobados por un Alumno	Se incrementa por cada Curso aprobado
Alumnos inscriptos en curso	Detalle de los Alumno que se encuentran inscriptos en un Curso	Por cada Alumno que se inscribe a un Curso se registra el Legajo Alumno
DNI	Número de Documento de la Persona	Asignar valor que corresponda
Apellido y Nombre	Apellido y Nombre de la Persona	Asignar valor que corresponda
Edad	Edad de la Persona	Asignar valor que corresponda

Sexo	Sexo de la Persona	Asignar valor que corresponda
Tipo Docente	Es una clasificación de Docente que pueden ser Profesor y Auxiliar Docente	Dependiendo de su clasificación podrán tener Asignación Profesores en cursos. o Asignación Auxiliares Docentes en cursos.
Categoría Profesor	Es una de las categorizaciones que puede tener un Profesor	Puede ser Titular , Asociado o Adjunto
Categoría Auxiliares	Es una de las categorizaciones que puede tener un Auxiliar Docente	Puede ser JTP , Ayte 1ra/ Ayte Graduado o Ayte2da/Ayte Alumno

Verbo	*Nociones:* describen quien ejecuta la acción, cuando ocurre, y cuáles son los procedimientos involucrados.
	Impactos: describen las restricciones sobre la acción, cuáles son las acciones desencadenadas en el ambiente y las nuevas situaciones que aparecen como resultado de la acción.

Verbos		
Entrada	Nociones	Impactos
Creación de curso.	Es iniciado por el Administrador. Al inicio de cada año. Identificación de: Profesor Auxiliar Docente y opcionalmente dos Ayte Alumno	Queda habilitado para: Asignación Profesores en cursos. Asignación Auxiliares Docentes en Cursos Asignación Ayte Alumno en cursos. Inscripción a cursado de Alumnos
Asignación Profesores en cursos.	Es realizado por el Administrador., cuando se efectúa la creación de Curso	Tiene que ser tipo docente profesor La Cantidad de cursos asignados no puede superar a Cantidad posibles de cursos Debe incrementarse la Cantidad de cursos asignados
Asignación Auxiliares Docentes en cursos.	Es realizado por el Administrador., cuando se efectúa la creación de Curso	Tiene que ser tipo docente Auxiliar Docente o profesor La Cantidad de cursos asignados no puede superar a Cantidad posibles de cursos Debe incrementarse la Cantidad de cursos asignados
Asignación Ayte Alumno en cursos.	Es realizado por el Administrador., cuando se efectúa la creación de Curso	Ser Auxiliar Docente categoría Ayte2da/Ayte Alumno La Cantidad de cursos asignados no puede superar a Cantidad posibles de cursos Debe incrementarse la Cantidad de cursos asignados
Inscripción a	Es realizado por el	No puede haber más de 30

cursado.	Administrador., cuando un Alumno lo solicita	Alumnos Se actualiza la nómina de Alumnos inscriptos a cursado en Curso Se incrementa en uno la Cantidad de Alumnos inscriptos
Dictado de Cursos	Es la actividad realizada por Profesores o Auxiliares Docentes dentro de un Curso	Tiene que haber una Creación de Curso

Estado	*Nociones:* describen que significa y que acciones pueden desencadenarse como consecuencia de ese estado.
	Impactos : describen otras situaciones y acciones relacionadas

Estados		
Entrada	Nociones	Impactos
Titular	Categoría de Profesor	Se encuentra Habilitado para tener Asignación Profesores en cursos. o Asignación Auxiliares Docentes en Cursos.
Asociado	Categoría de Profesor	Se encuentra Habilitado para tener Asignación Profesores en cursos. o Asignación Auxiliares Docentes en cursos.
Adjunto	Categoría de Profesor	Se encuentra Habilitado para tener Asignación Profesores en cursos. o Asignación Auxiliares Docentes en cursos.
JTP	Categoría de Auxiliar Docente	Se encuentra Habilitado para tener Asignación Auxiliares Docentes en cursos.
Ayte 1ra/ Ayte Graduado	Categoría de Auxiliar Docente	Se encuentra Habilitado para tener Asignación Auxiliares Docentes en cursos.

Nota Aclaratoria: Las entradas al LEL propuestas en el presente modelo se corresponden con el enunciado del Sistema Docente a modelar. Podría ser ampliada la funcionalidad incluyendo Verbos (Altas de Alumnos, altas de Docentes; cambios de categoría en Profesores y Auxiliares, Altas de Ayudantes Alumnos) pero están fuera de la definición.

1.4. Validación

Se valida el significado de los símbolos y se corrobora el lenguaje del vocabulario sea correcto, se introducen en las sucesivas iteraciones las entradas no detectadas en una etapa anterior

1.5. Control del LEL

Se verifica que el LEL terminado sea consistente y homogéneo, se controlan los símbolos verificando que pertenezcan a la clasificación correcta y que no existan sinónimos como símbolos diferentes, es fundamental que este proceso comience con las primeras descripciones de los símbolos.

2. Proceso de construcción de escenarios

La construcción de los escenarios se realiza a partir de la derivación del LEL, los pasos para realizar esta derivación son los siguientes:

2.1. Identificación de los actores de la aplicación.

Actores: Usuarios de la aplicación y deben ser englobados bajo la categoría sujeto en el LEL, los actores serán primarios o secundarios según el rol que desempeñe cada usuario, los primarios son los que se encuentran en contacto directo con la aplicación, el secundario solo recibe información.

Actores Primarios:

- Administrador

Actores Secundarios:

- Alumno
- Profesor
- Auxiliar Docente

2.2. Generación de la lista de escenarios candidatos, a partir de los actores principales.

La acción descripta como impacto en el LEL de los actores serán escenarios candidatos y se describen con la misma acción en infinitivo

Escenarios Candidatos:

- Crear Curso
- Asignar profesores a Curso
- Asignar Auxiliares Docentes a Curso
- Asignar Ayte Alumno a Curso
- Inscribir Alumnos en Curso

2.3. Descripción de los escenarios candidatos, provenientes de los actores principales.

Características:

- Un escenario describe situaciones con énfasis en la descripción del comportamiento.
- Un escenario utiliza la descripción textual como representación básica.
- Un escenario está naturalmente ligado al LEL al describirse con el vocabulario definido en el este último.

Los atributos que definen a un escenario son: título, objetivo, contexto, recursos, actores y episodios. Título, objetivo, contexto, recursos y actores son oraciones declarativas, mientras que episodios es un conjunto de oraciones de acuerdo con un lenguaje muy simple que hace posible la descripción operacional del comportamiento

- Título: Es la identificación del escenario. En el caso de un subescenario el título es el mismo que la oración del episodio desde el cual es referenciado.
- Objetivo: La meta que debe ser alcanzada.
- Contexto: Describe el estado inicial del escenario.
- Recursos: Son los elementos con los cuales se llevará a cabo el escenario.
- Actor: Es una persona o objeto que lleva adelante el escenario.
- Episodios: Se forman con una serie de oraciones que detallan el comportamiento del escenario. Cada oración indica una tarea en la secuencia del escenario. Hay un par de símbolos especiales. El símbolo "#" se utiliza para indicar acciones no secuenciales. Cuando no importa la secuencia en un grupo de pasos se los debe encerrar entre "#". Para indicar tareas condicionales se utiliza if then con la semántica tradicional.

Descripción de escenarios:

Título	Crear Curso
Objetivo	Creación de un Curso
Contexto	Institución Educativa
Recursos	Profesores , Auxiliares Docentes, Ayte Alumno
Actores	Administrador
Episodios	El Administrador asigna Código de curso , nombre curso y año dictado El Administrador inicializa a Cantidad de Alumnos inscriptos en cero El Administrador procede a Asignar profesor a Curso El Administrador procede a Asignar Auxiliar Docente a Curso El Administrador procede a Asignar Ayte Alumno a Curso

Título	Asignar Profesor a Curso
Objetivo	Asignar a un Profesor para el dictado del Curso
Contexto	Institución Educativa
Recursos	Profesor , Curso
Actores	Administrador
Episodios	El Administrador selecciona un tipo docente profesor **If** la Cantidad de cursos asignados es menor a Cantidad posibles de cursos **then** asigna Profesor a Curso e incrementa la Cantidad de cursos asignados

Título	Asignar Auxiliar Docente a Curso
Objetivo	Asignar a un Auxiliar Docente para colaborar con el dictado del Curso
Contexto	Institución Educativa
Recursos	Profesor , Auxiliar Docente, Curso
Actores	Administrador
Episodios	El Administrador selecciona un tipo docente Auxiliar Docente o profesor **If** la Cantidad de cursos asignados es menor a Cantidad posibles de cursos **then** asigna Auxiliar Docente a Curso e incrementa la Cantidad de cursos asignados

Título	Asignar Ayte Alumno a Curso
Objetivo	Asignar Ayte Alumno para colaborar con el dictado del Curso
Contexto	Institución Educativa
Recursos	Ayte Alumno, Curso
Actores	Administrador
Episodios	El Administrador selecciona uno o dos Ayte Alumno **If** la Cantidad de cursos asignados es cero **then** asigna Ayte Alumno a Curso e incrementa la Cantidad de cursos asignados

Título	Inscribir Alumno en Curso
Objetivo	Asignar Alumno a un Curso
Contexto	Institución Educativa
Recursos	Alumno , Curso
Actores	Administrador
Episodios	El Administrador identifica un Alumno a inscribirse **If** Cantidad de Alumnos inscriptos es menor que 30 **Then** Se actualiza la nómina de Alumnos inscriptos a cursado en Curso y se incrementa en uno la Cantidad de Alumnos inscriptos

2.4. Ampliación de la lista de escenarios candidatos, a partir de los actores secundarios.

No generan nuevos escenarios

2.5. Descripción de los escenarios candidatos, provenientes de actores secundarios.

No Hay

2.6. Revisión de de los escenarios.

Se revisan en función de los impactos de los verbos

2.7. Validación de escenarios.

Se contrasta con el dominio a modelar

3. Proceso de construcción de tarjetas CRC

Las Tarjetas CRC se obtienen a través de la derivación del LEL y escenarios, pudiendo dividirse en tres partes

3.1. Encontrar CRC primarias.

Los actores de los escenarios son candidatos a convertirse en CRC primarias

CRC Primaria	Administrador
Responsabilidades	Es quién realiza la Creación de curso. Realiza la asignación de Profesores a Curso mediante la Asignación Profesores en cursos. Realiza la asignación de Auxiliares Docentes a Cursos mediante la Asignación Auxiliares docentes en cursos. Realiza la asignación de Ayte Alumno a Cursos mediante la Asignación de Ayte Alumno en cursos. Realiza la Inscripción a cursado de Alumnos en Cursos
Colaboraciones	Curso ; Profesores ; Auxiliares Docentes ; Ayte Alumno ; Alumno Creación de curso. Asignación de Profesores en cursos. Asignación de Auxiliares docentes en cursos. Asignación de Ayte Alumno en cursos. Inscripción a cursado

3.2. Encontrar CRC secundarias.

Son aquellos colaboradores de las CRC primarias que las ayudan a cumplir con sus responsabilidades

CRC Secundaria	Curso
Responsabilidades	Asignación Profesores en cursos. Asignación Auxiliares docentes en cursos. Asignación Ayte Alumno en cursos. Inscripción a cursado de Alumnos.
Colaboraciones	Asignación Profesores en cursos. Asignación Auxiliares docentes en cursos. Asignación Ayte Alumno en cursos. Inscripción a cursado; Alumnos.

CRC Secundaria	Profesores
Responsabilidades	Dictado de Cursos en Curso
Colaboraciones	Curso ; Dictado de Cursos

CRC Secundaria	Auxiliares docentes
Responsabilidades	Colabora en Dictado de Cursos en Cursos

Colaboraciones	Curso ; Dictado de Cursos

CRC Secundaria	Ayte Alumno
Responsabilidades	Colabora en Dictado de Cursos en Cursos
Colaboraciones	Curso ; Dictado de Cursos

CRC Secundaria	Alumno
Responsabilidades	Inscripción a cursado en Cursos. Desempeñarse como Ayte Alumno en Cursos
Colaboraciones	Inscripción a Cursado ; Cursos ; Ayte Alumno

CRC Secundaria	Creación de Curso
Responsabilidades	Queda habilitado para: Asignación Profesores en cursos. Asignación Auxiliares Docentes en Cursos Asignación Ayte Alumno en cursos. Inscripción a cursado de Alumnos
Colaboraciones	

CRC Secundaria	Asignación Profesores a Curso
Responsabilidades	Tiene que ser tipo docente : profesor La Cantidad de cursos asignados no puede superar a Cantidad posibles de cursos Debe incrementarse la Cantidad de cursos asignados
Colaboraciones	

CRC Secundaria	Asignación Auxiliares Docentes a Curso
Responsabilidades	Tiene que ser tipo docente : Auxiliar Docente o profesor La Cantidad de cursos asignados no puede superar a Cantidad posibles de cursos Debe incrementarse la Cantidad de cursos asignados
Colaboraciones	

CRC Secundaria	Asignación Ayte Alumno a Curso
Responsabilidades	Ser Auxiliar Docente categoría Ayte2da/Ayte Alumno La Cantidad de cursos asignados no puede superar a Cantidad posibles de cursos Debe incrementarse la Cantidad de cursos asignados
Colaboraciones	

CRC Secundaria	Inscripción a Cursado
Responsabilidades	No puede haber más de 30 Alumnos Se actualiza la nómina de Alumnos inscriptos a cursado en Curso Se incrementa en uno la Cantidad de Alumnos inscriptos
Colaboraciones	

Encontrar colaboraciones.

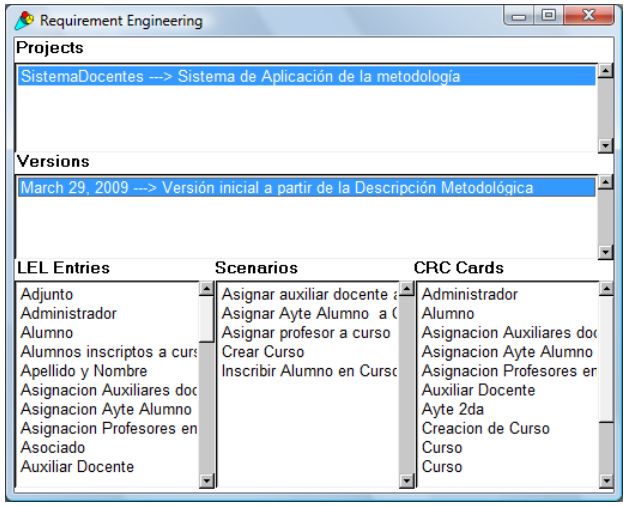
En los episodios de los escenarios se deben buscar las entradas de LEL que originan a las CRC tanto primarias como secundarias. Las tarjetas CRC que participan de un mismo escenario colaboran entre ellas.

USO DE LA HERRAMIENTA **BMW**

Pasos:

1) Se procede a la carga de todos los símbolos Identificados.
2) Se carga las nociones e impactos de los Sujetos / Objetos / Verbos / estados.
3) Se cargan los escenarios
4) Se generas las tarjetas CRC
5) Se comparan las tarjetas CRC obtenidas con la Metodología con las generadas automáticamente por la herramienta (Resultando Iguales)

Pantalla de Ejecución del BMW



Exportar modelo a HTML

Genera un Hipertexto HTML en el que se puede navegar sobre los conceptos, lo que facilita la validación del modelo.

ANEXO III: ONTOLOGÍAS/PROTÉGÉ - APLICACIÓN AL MODELO DE DOCENTES

Pasos:

• **1.- Determinar el dominio y alcance de la ontología, además del propósito u objetivo de la misma**,

- ¿Qué dominio cubrirá la ontología?
  Sistema educativo
- ¿Para qué se va a emplear la ontología?
  Organización de actividades académicas, asignando docentes afectados al dictado y participación de alumnos en los mismos
- ¿Qué preguntas debería contestar la ontología?
  Que cursos se dictan en un determinado período
  Que docentes imparten un curso determinado
  Que Alumnos asisten a un curso determinado
  Que alumnos se desempeñan como auxiliares docentes
  Cuales diferentes cursos, dicta un docente
  Cuales diferentes cursos asiste un alumno
  Cuales cursos y cuantos ha realizado un alumno
  Informar la categoría Docente
  Informar para las diferentes categorías que docentes existen
- ¿Quién usará y mantendrá la ontología?
  Uso: La organización institucional y el área de sistemas
  Mantenimiento: el área de Sistemas.

• **2.- Considerar la reutilización de ontologías existentes**.

No es pertinente, ya que lo que se pretende evaluar la construcción de un modelo conceptual a partir de la definición de la misma.

• **3.- Enumerar términos importantes de la ontología**.

Docente, Alumno, Cursos, período dictado; Tipo Docente (Profesor; Auxiliar Docente) categoría docentes (Titular, Asociado, Adjunto) Categoría Auxiliares (JTP; Ayte 1ra/ Ayte Graduado; Ayte2da/Ayte Alumno), año dictado Curso; Año Ingreso Docencia; Año Ingreso Alumno, cantidad de cursos asignados, cantidad de cursos máximos

• **4.- Definir las clases y la jerarquía de clases**.

En este punto se comienza con la enumeración de los conceptos o clases más destacadas y posteriormente se generalizan y especializan apropiadamente.

Se comienza identificando de la enumeración de términos, seleccionando aquellos que describan objetos con existencia independiente para constituir los conceptos o clases, en detrimento de aquellos términos que describan cómo son esos objetos.
Por lo tanto se parte analizando la creación de la clase Docente y Alumnos, en donde se ve la necesidad de contar con una clase abstracta Persona que agrupe los concentos comunes de docentes y alumnos.

Clases Identificadas:
Persona
    Docente
    Alumno
Curso

Se presentan alternativas de diseño sobre las jerarquías de clase Docente
Alternativa 1: Especializar a la clase docente en tres clases derivadas
Docente
        Profesor
        Auxiliar_docente
        Auxiliar_alumno

Alternativa 2: Especializar a la clase docente en clase Profesor y Auxiliar la cuál a su vez debe especializarse en Graduados y Alumnos
Docente
        Profesor
        Auxiliar_docente
                Auxiliar_docente_alumno
                Auxiliar_docente_graduado

Seleccionamos a la alternativa 2 ya que la misma representa más fielmente el concepto de las categorías docentes, facilitando la exploración y navegación de los términos.

Ahora las clases nos quedan:
Persona
    Docente
        Profesor
        Auxiliar_docente
                Auxiliar_docente_alumno
                Auxiliar_docente_graduado

Alumno

Curso

• **5.- Definir las propiedades/slots de las clases**.

Este paso, es realizado en forma conjunta con las definiciones de las clases. Describimos sus características y estructura. Al definir las clases y sus propiedades, no deberían quedarnos términos entre la lista de términos sin utilizar.

Propiedades de las clases:

- Persona (dni; edad; nombre; sexo)
  + Docente (legajo_docente; año_ingreso_docente, cantidad_cursos_asignados)
    + Profesor (Categoría_profesor)
      - Auxiliar_docente (Categoría_auxiliar)
        +Auxiliar_docente_alumno (legajo_alumno; cantidad_cursos_max_docente_alumno)
        + Auxiliar_docente_graduado (cantidad_cursos_max_docente_graduado)
  + Alumno (legajo_alumno; año_ingreso_alumno; cantidad_cursos_aprobados)

+ Curso (cod_curso; nom_curso; año_dictado; cod_profesor; cod_auxiliar; cod_ayte_alumno; alumnos_inscriptos)

• **6.- Definir las facetas o restricciones de las propiedades o slots**.

Para cada una de las propiedades definiremos (Cardinalidad; Tipo de valor; Dominio y Rango)

Clase	Slot	Card.	Tipo	Restricción
Persona	Dni	1	Int	1-100000000
	Edad	1	Int	0-99
	Nombre	1	Str	
	Sexo	1	Sym	M / F
Docente	legajo_docente	1	int	0-99999
	año_ingreso_docente	1	Int	1900-2099
	Cantidad_cursos_asignados	1	int	0-3
Profesor	categoría_profesor	1	Sym	TITULAR ASOCIADO ADJUNTO
	cantidad_cursos_max_profesor	1	int	0-3
Auxiliar_docente	categoría_auxiliar	1	Sym	JTP AYTE1RA AYTE2DA
Auxiliar_docente_alumno	legajo_docente_auxiliar_alumno	1	int	0-99999
	cantidad_cursos_max_docente_alumno	1	int	0-1
Auxiliar_docente_graduado	cantidad_cursos_max_docente_graduado	1	int	0-3
Alumno	legajo_alumno	1	int	0-1000000

	año_ingreso_alumno	1	Int	1900-2099
	cantidad_cursos_aprobados	1	int	0-50
Curso	codigo_curso	1	int	0-1000
	nombre_curso	1	Str	
	año_dictado	1	int	2000-2100
	docente_profesor	1	instance	Profesor
	docente_auxiliar_alumno	0-2	instance	Auxiliar_docente_alumno
	docente_auxiliar_graduado	1	instance	Auxiliar_docente_graduado
	alumnos_inscriptos	0- 30	instance	Alumno

PROTÉGÉ: ESTRUCTURA DE CLASES CON SUS SLOTS

A continuación se muestra una pantalla de la herramienta protege en donde a la izquierda pueden apreciarse la jerarquía de clases, y en la parte central la descripción de conceptos de la clases Curso junto con los spots y sus propiedades.

Definiremos ahora cada una de las clases en la herramienta

Clase: Curso

	Nombre del Slot	Documentación	Tipo	Cardinalidad
	alumnos_inscriptos		Alumno	0:30
	año_dictado		Integer	0:1
	codigo_curso	Numero identificatorio del curso	Integer	1:1
	docente_auxiliar_alumno		Auxiliar_docente_alumno	0:2
	docente_auxiliar_graduado		Auxiliar_docente_graduado, Profesor	0:1
	docente_profesor		Profesor	0:1
	nombre_curso		String	1:1

Clase Abstracta: Persona

	Nombre del Slot	Documentación	Tipo	Cardinalidad
	dni	Número de Documento de la persona	Integer	1:1

	edad	Es la edad de la persona	Integer	1:1
	nombre	Representará el nombre de la persona	String	1:1
	sexo	Sexo de la persona, admite valores M= Masculino F= femenino	{M, F}	1:1

Clase: Alumno

	Nombre del Slot	Documentación	Tipo	Cardinalidad
	año_ingreso_alumno	Año de ingreso del alumno	Integer	1:1
	cantidad_cursos_aprobados	Cantidad de cursos aprobados al finalizar el ultimo año	Integer	1:1
	dni	Número de Documento de la persona	Integer	1:1
	edad	Es la edad de la persona	Integer	1:1
	inscripto_a_cursado		Curso	0:10
	legajo_alumno	Legajo de una alumno	Integer	1:1
	nombre	Representará el nombre de la persona	String	1:1
	sexo	Sexo de la persona, admite valores M= Masculino F= femenino	{M, F}	1:1

Clase Abstracta: Docente

	Nombre del Slot	Documentación	Tipo	Cardinalidad
	año_ingreso_docente	Año de ingreso a la docencia	Integer	1:1
	cantidad_cursos_asignados		Integer	1:1
	dni	Número de Documento de la persona	Integer	1:1
	edad	Es la edad de la persona	Integer	1:1
	legajo	es el legajo del docente - el mismo se deberá asignar en la creación incrementado en uno al anterior otorgado	Integer	1:1
	nombre	Representará el nombre de la persona	String	1:1
	sexo	Sexo de la persona, admite valores M= Masculino F= femenino	{M, F}	1:1

Clase Abstracta: Auxiliar_docente

	Nombre del Slot	Documentación	Tipo	Cardinalidad
	año_ingreso_docente	Año de ingreso a la docencia	Integer	1:1
	cantidad_cursos_asignados		Integer	1:1
	categoria_auxiliar		{JTP, AYTE1RA, AYTE2DA}	1:1
	dni	Número de Documento de la persona	Integer	1:1
	edad	Es la edad de la persona	Integer	1:1
	legajo	es el legajo del docente - el mismo se deberá asignar en la creación incrementado en uno al anterior otorgado	Integer	1:1
	nombre	Representará el nombre de la persona	String	1:1
	sexo	Sexo de la persona, admite valores M= Masculino F= femenino	{M, F}	1:1

Clase: Auxiliar_docente_alumno

	Nombre del Slot	Documentación	Tipo	Cardinalidad
	año_ingreso_docente	Año de ingreso a la docencia	Integer	1:1
	cantidad_cursos_asignados		Integer	1:1
	cantidad_cursos_max_docente_alumno		Integer	1:1
	categoria_auxiliar		{JTP, AYTE1RA, AYTE2DA}	1:1
	dni	Número de Documento de la persona	Integer	1:1
	edad	Es la edad de la persona	Integer	1:1
	legajo	es el legajo del docente - el mismo se deberá asignar en la creación incrementado en uno al anterior otorgado	Integer	1:1
	legajo_docente_auxiliar_alumno		Integer	1:1
	nombre	Representará el nombre de la persona	String	1:1
	sexo	Sexo de la persona, admite valores M= Masculino F= femenino	{M, F}	1:1

Clase: Auxiliar_docente_graduado

	Nombre del Slot	Documentación	Tipo	Cardinalidad
	año_ingreso_docente	Año de ingreso a la docencia	Integer	1:1
	cantidad_cursos_asignados		Integer	1:1
	cantidad_cursos_max_docente_graduado		Integer	1:1
	categoria_auxiliar		{JTP, AYTE1RA, AYTE2DA}	1:1
	dni	Número de Documento de la persona	Integer	1:1
	edad	Es la edad de la persona	Integer	1:1
	legajo	es el legajo del docente - el mismo se deberá asignar en la creación incrementado en uno al anterior otorgado	Integer	1:1
	nombre	Representará el nombre de la persona	String	1:1
	sexo	Sexo de la persona, admite valores M= Masculino F= femenino	{M, F}	1:1

• 7.- Crear instancias.

Se crean instancias de las clases para posteriormente verificar las preguntas de competencia.

PROTÉGÉ: INSTANCIAS DE CLASES

Clase: Auxiliar_docente_graduado

Instancia: ModeloDocente_vers1_Instance_40

Nombre del Slot	Valor
año_ingreso_docente	1999

cantidad_cursos_asignados	2
cantidad_cursos_max_docente_graduado	3
categoria_auxiliar	AYTE1RA
dni	34657876
edad	35
legajo	76987
nombre	Juan
sexo	M

Instancia: ModeloDocente_vers1_Instance_42

Nombre del Slot	Valor
año_ingreso_docente	2001
cantidad_cursos_asignados	1
cantidad_cursos_max_docente_graduado	3
categoria_auxiliar	JTP
dni	67453232
edad	47
legajo	78987
nombre	Yo
sexo	M

Clase: Auxiliar_docente_alumno

Instancia: ModeloDocente_vers1_Instance_43

Nombre del Slot	Valor
año_ingreso_docente	2004
cantidad_cursos_asignados	1
cantidad_cursos_max_docente_alumno	1
categoria_auxiliar	AYTE2DA
dni	16273876
edad	46
legajo	654321
legajo_docente_auxiliar_alumno	123456
nombre	A1
sexo	M

Instancia: ModeloDocente_vers1_Instance_52

Nombre del Slot	Valor
año_ingreso_docente	2005

cantidad_cursos_asignados	1
cantidad_cursos_max_docente_alumno	1
categoria_auxiliar	AYTE2DA
dni	6756435
edad	19
legajo	67876
legajo_docente_auxiliar_alumno	
nombre	A5
sexo	M

Clase: Profesor

Instancia: ModeloDocente_vers1_Instance_39

Nombre del Slot	Valor
año_ingreso_docente	2005
cantidad_cursos_asignados	2
cantidad_cursos_max_profesor	3
categoría_profesor	TITULAR
dni	12736827
edad	45
legajo	87656
nombre	Pepe
sexo	M

Instancia: ModeloDocente_vers1_Instance_41

Nombre del Slot	Valor
año_ingreso_docente	1998
cantidad_cursos_asignados	1
cantidad_cursos_max_profesor	3
categoría_profesor	ASOCIADO
dni	78987456
edad	43
legajo	87654
nombre	Jose
sexo	M

Clase: Alumno

Instancia: ModeloDocente_vers1_Instance_44

Nombre del Slot	Valor
año_ingreso_alumno	2000

cantidad_cursos_aprobados	2
dni	16765656
edad	20
inscripto_a_cursado	ModeloDocente_vers1_Instance_48, ModeloDocente_vers1_Instance_45
legajo_alumno	654321
nombre	A1
sexo	M

Instancia: ModeloDocente_vers1_Instance_49

Nombre del Slot	Valor
año_ingreso_alumno	2001
cantidad_cursos_aprobados	10
dni	29093848
edad	18
inscripto_a_cursado	ModeloDocente_vers1_Instance_45, ModeloDocente_vers1_Instance_48
legajo_alumno	87654
nombre	A2
sexo	F

Instancia: ModeloDocente_vers1_Instance_50

Nombre del Slot	Valor
año_ingreso_alumno	2003
cantidad_cursos_aprobados	7
dni	28983444
edad	19
inscripto_a_cursado	ModeloDocente_vers1_Instance_48, ModeloDocente_vers1_Instance_45
legajo_alumno	87654
nombre	A3
sexo	M

Instancia: ModeloDocente_vers1_Instance_51

Nombre del Slot	Valor
año_ingreso_alumno	2000
cantidad_cursos_aprobados	5
dni	27876988
edad	5
inscripto_a_cursado	ModeloDocente_vers1_Instance_47
legajo_alumno	87654
nombre	A4

sexo	M

Instancia: ModeloDocente_vers1_Instance_53

Nombre del Slot	Valor
año_ingreso_alumno	2001
cantidad_cursos_aprobados	8
dni	123547
edad	19
inscripto_a_cursado	ModeloDocente_vers1_Instance_45
legajo_alumno	
nombre	A5
sexo	M

Clase: Curso

Instancia: ModeloDocente_vers1_Instance_45

Nombre del Slot	Valor
alumnos_inscriptos	ModeloDocente_vers1_Instance_44, ModeloDocente_vers1_Instance_49, ModeloDocente_vers1_Instance_50
año_dictado	2009
codigo_curso	1
docente_auxiliar_alumno	ModeloDocente_vers1_Instance_43
docente_auxiliar_graduado	ModeloDocente_vers1_Instance_41
docente_profesor	ModeloDocente_vers1_Instance_39
nombre_curso	1

Instancia: ModeloDocente_vers1_Instance_47

Nombre del Slot	Valor
alumnos_inscriptos	ModeloDocente_vers1_Instance_51
año_dictado	2009
codigo_curso	2
docente_auxiliar_alumno	
docente_auxiliar_graduado	ModeloDocente_vers1_Instance_40
docente_profesor	ModeloDocente_vers1_Instance_41
nombre_curso	2

Instancia: ModeloDocente_vers1_Instance_48

Nombre del Slot	Valor
alumnos_inscriptos	ModeloDocente_vers1_Instance_49, ModeloDocente_vers1_Instance_50, ModeloDocente_vers1_Instance_44

año_dictado	2009
codigo_curso	3
docente_auxiliar_alumno	
docente_auxiliar_graduado	ModeloDocente_vers1_Instance_42
docente_profesor	ModeloDocente_vers1_Instance_41
nombre_curso	3

• **8.- Ejecutar las preguntas de competencia**.

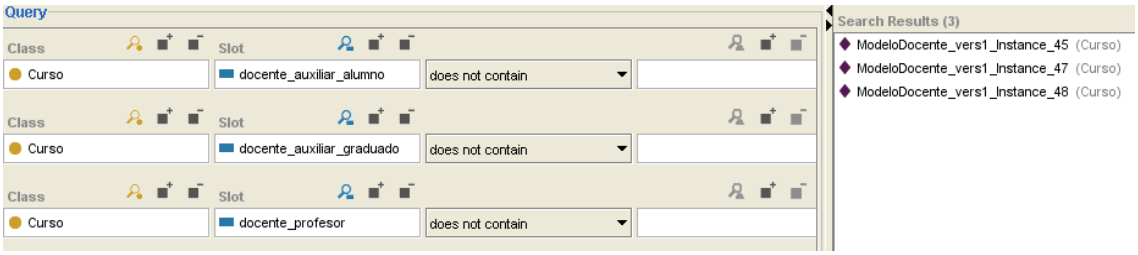
PROTÉGÉ: PREGUNTAS DE COMPETENCIA

- Que cursos se dictan en un determinado período

La pregunta ha sido satisfactoriamente resuelta.

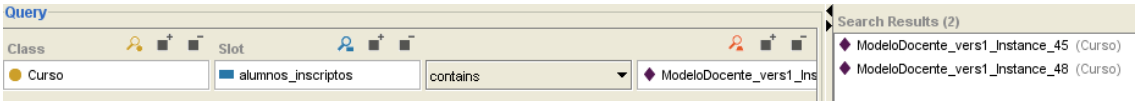
- Que docentes imparten un curso determinado

Solo se puede preguntar, si A docente dicta B curso. No se puede determinar cuales son los docentes del curso A. (pregunta cuales son las instancias que cumplen ciertas condiciones, por ejemplo, cuales son las instancias de la clase curso que son dictados por el profesor C). En este caso, se hizo algo un poco rebuscado y desprolijo para poder obtener la respuesta.



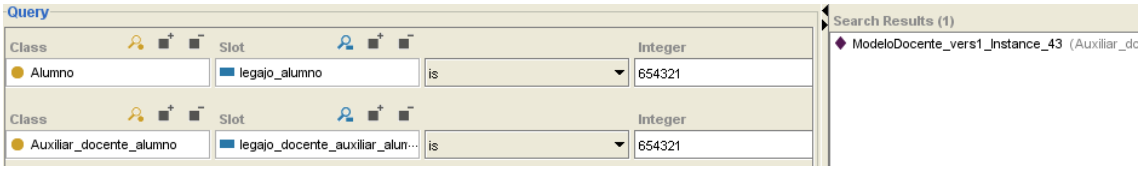
- Que Alumnos asisten a un curso determinado

Se puede determinar los cursos a los que asiste un determinado alumno pero no a la inversa.



- Que alumnos se desempeñan como auxiliares docentes

Se puede saber si un determinado alumno se desempeña como auxiliar docente, pero no se puede determinar cuales son los alumnos que también son auxiliares docentes.



- Cuales diferentes cursos, dicta un docente

La pregunta ha sido satisfactoriamente resuelta.

- Cuales diferentes cursos asiste un alumno

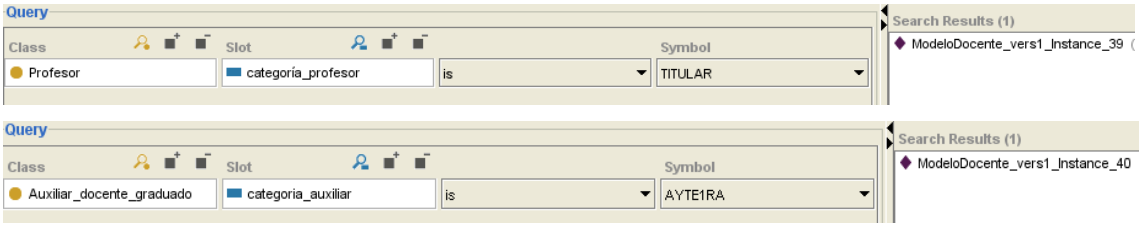
La pregunta ha sido satisfactoriamente resuelta.

- Cuales cursos y cuantos ha realizado un alumno

Se puede determinar cuáles son los cursos, pero no se puede determinar cuantos

- Informar la categoría Docente

No se puede determinar la categoría docente en general, si se puede determinar la categoría de un profesor o de un auxiliar determinado.



- Informar para las diferentes categorías que docentes existen

No se puede determinar para cada categoría, los docentes que existen, para ello se debería agregar una clase que sea de categorías.

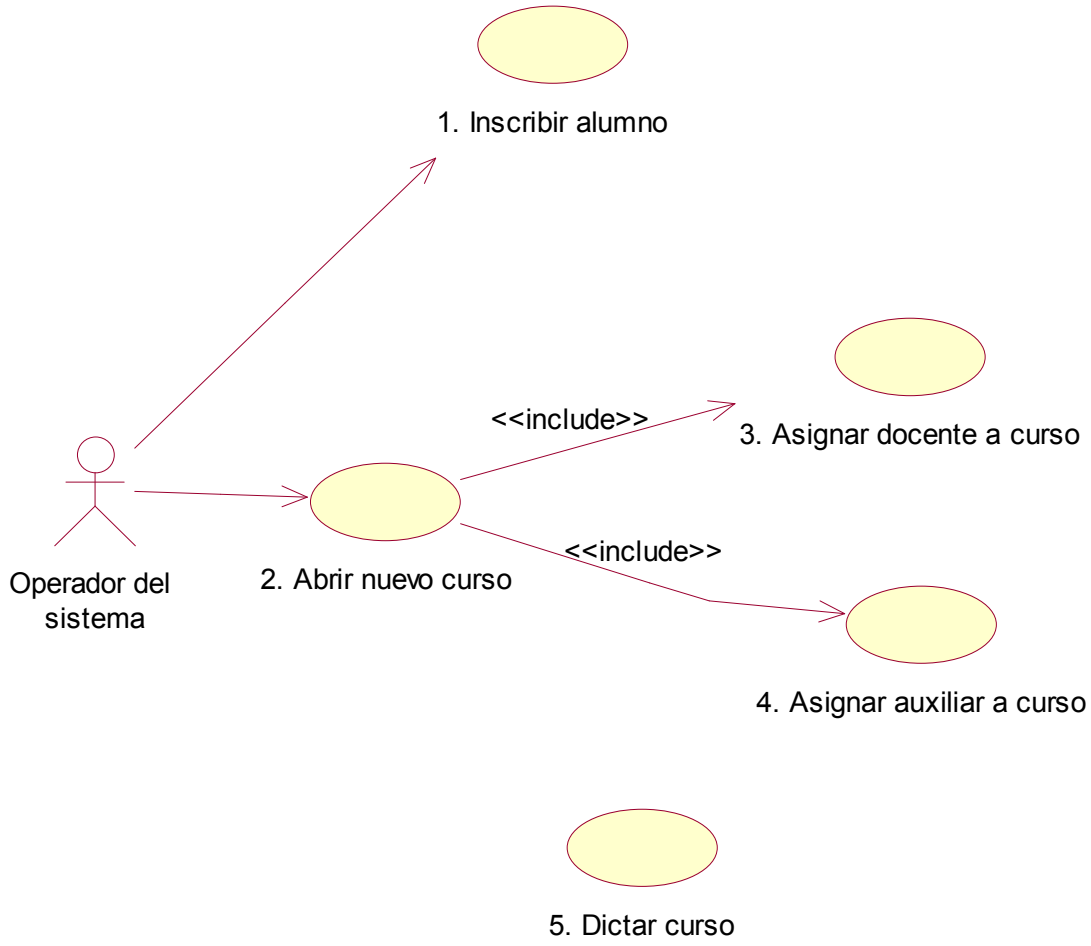
Solo se puede saber la categoría de un determinado profesor o auxiliar.

ANEXO IV: RUP/RATIONAL - APLICACIÓN AL MODELO DE DOCENTES

Modelos utilizados para la obtención del modelo conceptual:

- **1.- Modelo de Casos de Uso del Negocio**
- **2.- Descripción de Casos de Uso**
- **3.- Modelo de Casos de Uso del Sistema de Información**
- **4.- Diagrama de clases**

1. Modelo de casos de uso del negocio



2. Casos de uso

Nivel de Caso de Uso:	Negocio ■	Sistema de información □				**Orden: 1**
Nombre del Caso de Uso: *Asignar docente a curso*						
Objetivo: Realizar la asignación de un docente a un nuevo curso.						
Actor Principal: No aplica.		**Actor Secundario:** No aplica.				
Tipo de Caso de Uso :	Concreto ■	Abstracto □				
Prioridad :	Alta □	Media ■	Baja □			
Complejidad:	Alta □	Media ■	Baja □			
Pre-condiciones	No aplica.					
Post-condiciones	**Éxito:** Se ha registrado el docente en el nuevo curso.					
	Fracaso: No aplica.					
Curso Normal			**Curso Alternativo**			
1. El Caso de Uso (UC) comienza cuando se desea crear un nuevo curso, para lo cual se deberá asignar un nuevo docente al mismo.						
2. Se guardan los datos relacionados con el docente en el curso que se asignará (legajo, nombre, apellido).						
3. Fin del Use Case.						
Use Case que lo extiende: No aplica.						
Use Case incluido en: UC 3 “Abrir nuevo curso”.						
Use Case al cual incluye: No aplica.						
Use Case al cual extiende: No aplica.						
Caso de Uso de Generalización: No aplica.						
Autor: **MEC**		Fecha de creación: 06/04/09				Versión
Autor Ultima Modificación: **MEC**		Fecha Ultima Modificación: 08/04/09				1.0

Nivel de Caso de Uso:	Negocio ■	Sistema de información □				**Orden: 2**
Nombre del Caso de Uso: *Inscribir a alumno*						
Objetivo: Realizar la inscripción de un alumno a un curso determinado						
Actor Principal: No aplica.		**Actor Secundario:** No aplica.				
Tipo de Caso de Uso :	Concreto ■	Abstracto □				
Prioridad :	Alta ■	Media □	Baja □			
Complejidad:	Alta □	Media ■	Baja □			
Pre-condiciones	Solo un Alumno de la carrera puede inscribirse en un curso.					
Post-condiciones	**Éxito:** Se ha inscripto el alumno al curso.					
	Fracaso: No se ha podido inscribir al alumno al curso porque no hay cupo.					
Curso Normal			**Curso Alternativo**			
1. El Caso de Uso (UC) comienza cuando se desea inscribir un alumno a un curso.						
2. El Alumno informa el curso donde desea inscribirse.						
3. Se corrobora el cupo para el curso y hay disponibilidad.			3.A. Se corrobora el cupo para el curso y no hay disponibilidad.			
			3.A.1. No se puede inscribir en el curso solicitado.			
			3.A.2. Se cancela UC.			
4. Se solicitan los datos personales del Alumno (Legajo, Nombre, Apellido).						
5. Se realiza la inscripción del alumno al curso.						
6. Fin del Use Case.						
Use Case que lo extiende: No aplica.						
Use Case incluido en: No aplica.						
Use Case al cual incluye: No aplica.						
Use Case al cual extiende: No aplica.						
Caso de Uso de Generalización: No aplica.						
Autor: **MEC**		Fecha de creación: 06/04/09				Versión 1.0
Autor Ultima Modificación: **MEC**		Fecha Ultima Modificación: 08/04/09				

Nivel de Caso de Uso:	Negocio ■	Sistema de información □				**Orden: 3**
Nombre del Caso de Uso: *Abrir nuevo curso*						
Objetivo: Realizar la apertura de un nuevo curso.						
Actor Principal: No aplica.		**Actor Secundario:** No aplica.				
Tipo de Caso de Uso :	Concreto ■	Abstracto □				
Prioridad :	Alta ■	Media □		Baja □		
Complejidad:	Alta □	Media ■		Baja □		
Pre-condiciones	No aplica.					
Post-condiciones	**Éxito:** Se ha abierto el nuevo curso.					
	Fracaso: No aplica.					

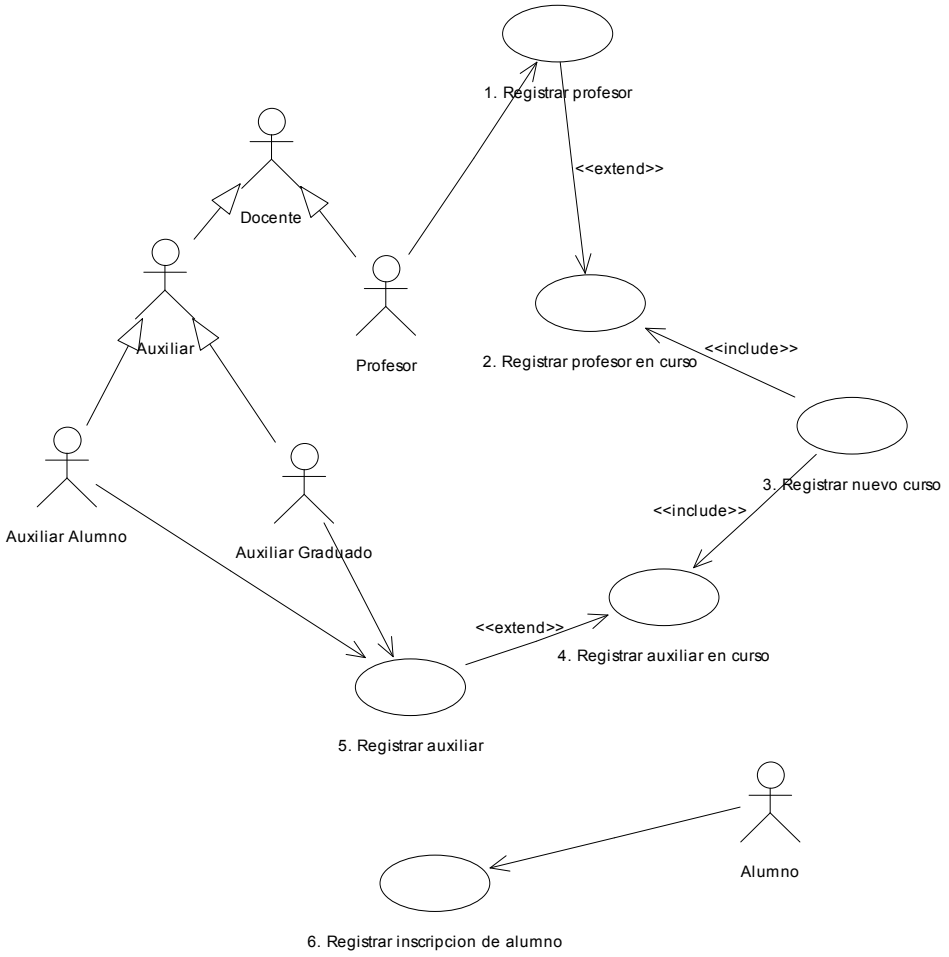
Curso Normal	**Curso Alternativo**
1. El Caso de Uso (UC) comienza cuando se desea crear un nuevo curso.	
2. Se desea asignar un docente al curso.	
3. Se llama al UC 1 "*Asignar docente a curso*".	
4. Se desea asignar un auxiliar al curso.	
5. Se llama al UC 4 "*Asignar auxiliar a curso*".	
6. No se desea asignar un ayudante al curso	6.A. Se desea asignar un ayudante al curso.
	6.A.1. Se asigna un ayudante al curso, guardando sus datos.
7. Se guardan los datos generales del curso (código de Identificación, nombre del curso, año de dictado).	
8. Fin del Use Case.	

Use Case que lo extiende: No aplica.		
Use Case incluido en: No aplica.		
Use Case al cual incluye: UC 1 "*Asignar docente a curso*", UC 4 "*Asignar auxiliar a curso*".		
Use Case al cual extiende: No aplica.		
Caso de Uso de Generalización: No aplica.		
Autor: **MEC**	Fecha de creación: 06/04/09	Versión 1.0
Autor Ultima Modificación: **MEC**	Fecha Ultima Modificación: 08/04/09	

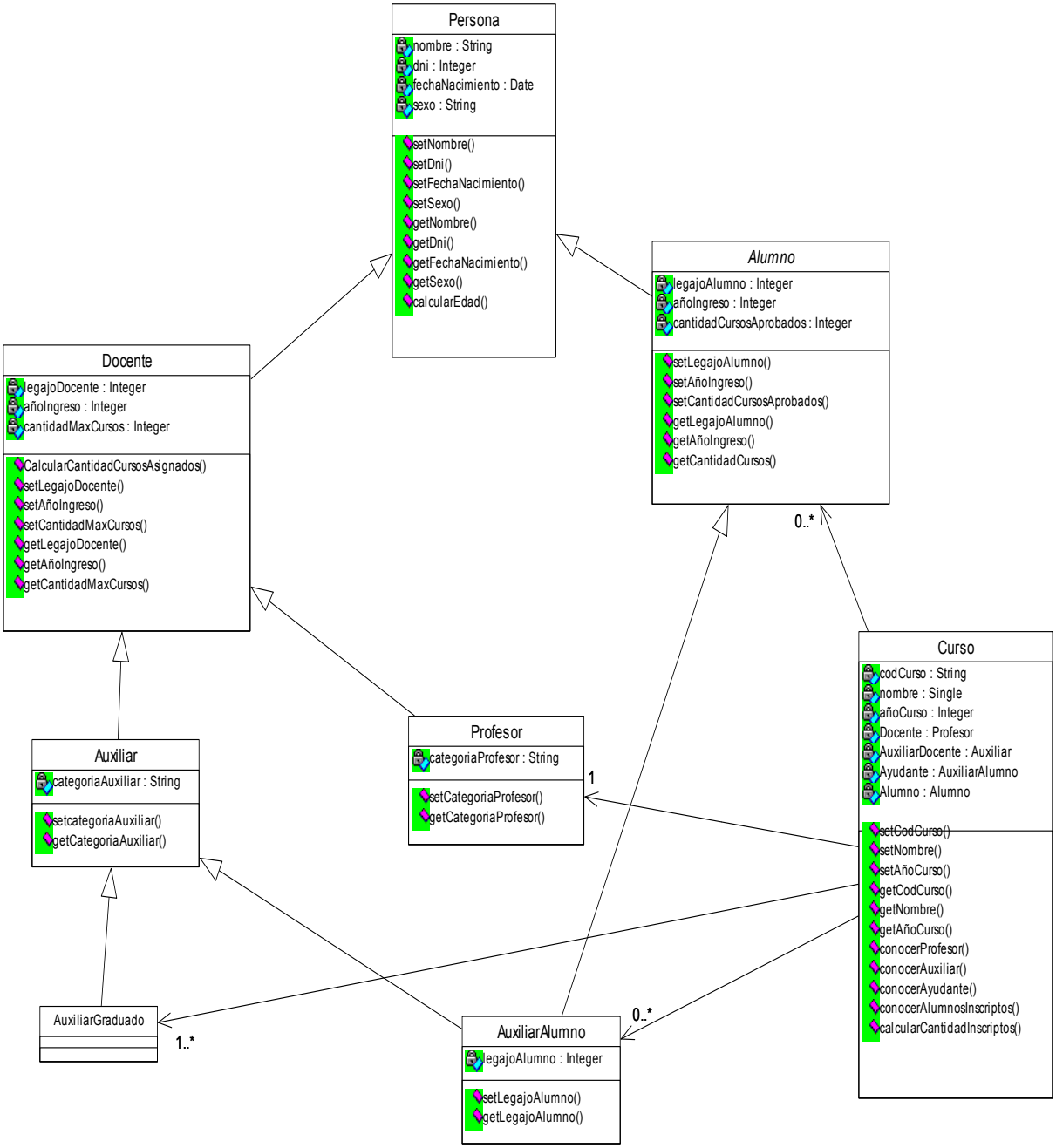
Nivel de Caso de Uso:	Negocio ■	Sistema de información □				**Orden: 4**
Nombre del Caso de Uso: *Asignar auxiliar a curso*						
Objetivo: Realizar la asignación de un auxiliar a un nuevo curso.						
Actor Principal: No aplica.		**Actor Secundario:** No aplica.				
Tipo de Caso de Uso :	Concreto ■	Abstracto □				
Prioridad :	Alta □	Media ■	Baja □			
Complejidad:	Alta □	Media ■	Baja □			
Pre-condiciones	No aplica.					
Post-condiciones	**Éxito:** Se ha asignado el auxiliar al nuevo curso.					
	Fracaso: No aplica.					
Curso Normal			**Curso Alternativo**			
1. El Caso de Uso (UC) comienza cuando se desea crear un nuevo curso, para lo cual se deberá asignar un nuevo auxiliar al mismo.						
2. Se guardan los datos relacionados con el auxiliar en el curso que se asignará (legajo, nombre, apellido).						
3. Fin del Use Case.						
Use Case que lo extiende: No aplica.						
Use Case incluido en: "Abrir nuevo curso".						
Use Case al cual incluye: No aplica.						
Use Case al cual extiende: No aplica.						
Caso de Uso de Generalización: No aplica.						
Autor: **MEC**		Fecha de creación: 07/04/09				Versión 1.0
Autor Ultima Modificación: **MEC**		Fecha Ultima Modificación: 08/04/09				

Nivel de Caso de Uso:	Negocio ■	Sistema de información □				**Orden: 5**
Nombre del Caso de Uso: *Dictar curso*						
Objetivo: Realizar la apertura de un nuevo curso.						
Actor Principal: No aplica.		**Actor Secundario:** No aplica.				
Tipo de Caso de Uso :	Concreto ■	Abstracto □				
Prioridad :	Alta ■	Media □	Baja □			
Complejidad:	Alta □	Media ■	Baja □			
Pre-condiciones	No aplica.					
Post-condiciones	**Éxito:** No aplica.					
	Fracaso: No aplica.					
Curso Normal			**Curso Alternativo**			
1. El Caso de Uso (UC) comienza cuando se han inscripto los alumnos al curso y es tiempo de inicio de dictado del curso.						
2. El docente comienza con el dictado del curso.						
3. Fin del Use Case.						
Use Case que lo extiende: No aplica.						
Use Case incluido en: No aplica.						
Use Case al cual incluye: No aplica.						
Use Case al cual extiende: No aplica.						
Caso de Uso de Generalización: No aplica.						
Autor: **MEC**		Fecha de creación: 08/04/09				Versión 1.0
Autor Ultima Modificación: **MEC**		Fecha Ultima Modificación: 08/04/09				

3. Modelo de casos de uso del sistema de información



4. Diagrama de clases



ANEXO V: LEL/ESCENARIOS/CRC - APLICACIÓN AL DOMINIO DE GRAMÁTICAS FORMALES Y MÁQUINAS ABSTRACTAS

1. Construcción del LEL.
2. Construcción de los Escenarios a partir del LEL.
3. A partir del LEL y escenarios se construyen las fichas CRC

1. Proceso de construcción del LEL

1.1. Generación de la lista de símbolos

Son los detectados al comienzo del proceso, los cuales son ampliados en las sucesivas iteraciones:

GRAMATICAS FORMALES	En AMBOS Modelos GF y MA	MAQUINAS ABSTRACTAS
Palabra/Tira/Cadena		
Palabra Generada	Palabra Generada / Palabra Aceptada	Palabra Aceptada
Lenguaje Generado / Lenguaje Formal	Lenguaje Generado/Lenguaje Aceptado / Lenguaje Formal	Lenguaje Aceptado / Lenguaje Formal
Isomorfismo/Equivalencia		
Conversión de una Gramática Regular a Autómata Finito.		
Conversión de un Autómata Finito a Gramática Regular.		
Conversión de una Gramática Independiente de Contexto a Autómata con Pila.		
Lenguaje Regular		
Lenguaje Independiente del Contexto		
Lenguaje Dependiente del Contexto		
Lenguaje Estructurado por Frases		
Gramática Formal		Máquinas Abstractas
Símbolos Terminales		Entradas
Símbolos No Terminales		Estados
Conjunto de Símbolos Terminales		Alfabeto de entrada
Conjunto de Símbolos No Terminales		Conjunto de estados
Lambda		
		Alfabeto de Pila
		Símbolo inicial de Pila
		Conjunto de Estados Finales de Aceptación
Axioma Inicial		Estado Inicial
		Estado final / Aceptación
Conjunto de Producciones		Función de Transición
Producción/ Reglas de Producción		Transición
Derivación Directa		Movimiento

Generación de Cadenas		Aceptación de Cadenas
GT0 - Estructurada por Frases		MT – Máquina de Turing
GT1 - Dependiente del contexto		ALA – Autómata Linealmente Acorado
GT2 – Gram. Independiente del Contexto		AP – Autómata con Pila
GT3 – Gramática Regular		AF – Autómata Finito

1.2. Clasificación de los símbolos

Símbolo	Clasificación
Palabra/Tira/Cadena	Objeto
Palabra Generada / Palabra Aceptada	Objeto
Lenguaje Generado / Lenguaje Formal	Objeto
Isomorfismo / Equivalencia	Sujeto
Conversión de una Gramática Regular a Autómata Finito.	Verbo
Conversión de un Autómata Finito a Gramática Regular.	Verbo
Conversión de una Gramática Independiente de Contexto a Autómata con Pila.	Verbo
Lenguaje Regular	Objeto
Lenguaje independiente del Contexto	Objeto
Lenguaje Dependiente del Contexto	Objeto
Lenguaje Estructurado por Frases	Objeto
Símbolos Terminales	Objeto
Símbolos No Terminales	Objeto
Gramática Formal	Objeto
Conjunto Símbolos Terminales	Objeto
Conjunto Símbolos No Terminales	Objeto
Lambda	Estado
Axioma Inicial	Objeto
Conjunto de Producciones	Objeto
Producción/ Reglas de Producción	Objeto
Derivación Directa	Verbo
Generación de Cadenas	Verbo
GT0 - Estructurada por Frases	Objeto
GT1 - Dependiente del contexto	Objeto
GT2 – Gram. Independiente del Contexto	Objeto
GT3 – Gramática Regular	Objeto
Máquinas Abstractas	Objeto
Entradas	Objeto
Estados	Objeto
Alfabeto de entrada	Objeto
Conjunto de estados	Objeto
Conjunto de Estados Finales	Objeto
Alfabeto de Pila	Objeto
Símbolo inicial de Pila	Objeto
Estado Inicial	Objeto
Estado final / Aceptación	Objeto
Función de Transición	Objeto
Transición	Verbo

1.3.

Movimiento	Objeto
Aceptación de Cadenas	Verbo
MT – Máquina de Turing	Objeto
ALA – Autómata Linealmente Acorado	Objeto
AP - Autómata con Pila	Objeto
AF – Autómata Finito	Objeto

Descripción de los símbolos

Sujeto	*Nociones:* describen quien es el sujeto.
	Impactos: registran acciones ejecutadas por el sujeto

Sujetos		
Entrada	Nociones	Impactos
Operador	Es quién creará y utilizará a las gramáticas Formales que Generará Cadenas de los Lenguajes Formales Es quién Creará máquinas abstractas y reconocerá cadenas de los Lenguajes Formales Realizará los procesos de conversión (isomorfismos)	Creación de una Gramática Formal a través de la definición del Conjunto de Símbolos Terminales , Conjunto de Símbolos No Terminales , axioma inicial y definición de las producciones del conjunto de reglas de producción Identificación de gramática formal , en el tipo de gramática formal que corresponda GT0, GT1, GT2, GT3 Generación de Cadenas de Lenguajes Formales Creación de máquinas Abstractas ya sean Máquinas de Turing , Autómatas Linealmente Acotados, Autómatas con Pila o Autómatas finitos. Definiendo sus componentes Alfabeto de entrada , Alfabeto de Pila , Conjunto de Estados , Estado Inicial , Símbolo inicial de Pila , Conjunto de Estados Finales , Función de Transición. Dada una Cadena de Entrada efectuar transiciones entre estados para Reconocer cadenas de entrada Conversión de una Gramática Regular a Autómata Finito. Conversión de un Autómata Finito a Gramática Regular.

		Conversión de una Gramática Independiente de Contexto a Autómata con Pila.

Objeto	*Nociones:* definen al objeto e identifica a otros términos con los cuales el objeto tiene algún tipo de relación.
	Impactos: describen las acciones que pueden ser aplicadas al objeto.

Objetos		
Entrada	Nociones	Impactos
Conjunto de Símbolos Terminales	Colección de los Símbolos Terminales de una Gramática Formal	Definirse a partir del Lenguaje Formal a construir
Conjunto de Símbolos No Terminales	Colección de Símbolos No Terminales de una Gramática Formal	Definirse a partir del conjunto de producciones de la Gramática Formal
Axioma Inicial	Símbolo No Terminal especial por donde comenzará la Generación de cadenas	Origen de toda Generación de cadenas de un Lenguaje Formal
Conjunto de Producciones	Colección de Producciones de una Gramática Formal	Utilizadas en las Derivaciones Directas
Símbolos Terminales	Símbolos que formarán parte del Conjunto de Símbolos Terminales Formarán parte de las Palabras Generadas por las Gramáticas Formales Formarán parte en la definición de las reglas de producción	Incluirse en conjunto de Símbolos Terminales
Símbolos No Terminales	Símbolos que formarán parte del Conjunto de Símbolos No Terminales Formarán parte en la definición de las reglas de producción Símbolos a sustituir en una Derivación Directa	Incluirse en conjunto de Símbolos No Terminales
Palabra / Tira / Cadena	Es una secuencia finita de símbolos Formada por Símbolos Terminales y/o Símbolos No Terminales	Pueden ser Generadas por las Gramáticas Formales Transformadas en otras aplicándoles reglas de producción
Palabras Generadas	Formada Solamente por Símbolos Terminales Generada a través de una Gramática Formal por la aplicación de derivaciones del conjunto de reglas de	Pueden ser Generadas por las Gramáticas Formales Formar parte de un Lenguaje Formal Reconocidas por las Máquinas Abstractas

	Producción a partir del axioma Inicial Formar parte del Lenguaje Generado por una Gramática Formal	
Gramática Formal	Notación para la descripción de Lenguajes Formales Queda definida por un cuádrupla a través de: Conjunto de Símbolos Terminales , Conjunto de Símbolos No Terminales, Axioma Inicial, Conjunto de Producciones	Generación de Lenguajes Formales
GT3 / Gramática Regular	Es una GT2 donde al menos a una producciones del conjunto de producciones se le agrega la restricción de que la Parte Derecha: Un Símbolo Terminal seguido de un Símbolo No Terminal o un Símbolo Terminal o Lambda si es una producción desde el axioma inicial	Generación de Lenguajes Regulares Posibilita la definición del Autómata Finito equivalente
GT2 / Gramática Independiente del Contexto	Es una GT1 donde al menos a una producciones del conjunto de producciones se le agrega la restricción de que la parte Izquierda tiene que estar formado por un único Símbolo No Terminal. La parte Derecha no tiene restricciones, salvo que si es Lambda solo si es una producción desde el axioma inicial	Generación de Lenguajes Independientes del Contexto Posibilita la definición Autómata Con Pila equivalente
GT1 / Gramática Dependiente del Contexto	Es una GT0 donde al menos a una producciones del conjunto de producciones se le agrega como restricción adicional no ser reglas compresoras, esto es que la longitud de las cadenas del lado izquierdo debe ser menor o igual que del lado derecho, salvo que el lado izquierdo sea el axioma inicial de la Gramática en cuyo caso se permite que el lado derecho sea la cadena vacía (Lambda)	Generación de Lenguajes Dependientes del Contexto
GT0 / Gramática estructurada por frases	Es una Gramática Formal Cada Producción del conjunto de producciones debe estar formado por: Lado Izquierdo: Cadenas de Símbolos Terminales y Símbolos	Generación de Cadenas de Lenguajes Estructurados por frases

	No Terminales con al menos un Símbolo No Terminal Lado derecho: Sin Restricciones	
Producción/ Producciones / Reglas de Producción	Tienen una parte derecha y una parte izquierda. De acuerdo a las restricciones que se les impongan a estas partes pueden clasificarse en diferentes tipos	Son utilizadas en una derivación
Lenguajes Formales / Lenguaje Generado	Es el conjunto de palabras generadas a partir de una Gramática Formal	Dependiendo el tipo de Gramática Formal es el tipo de lenguaje Formal generado
Lenguaje Regular	Es un Lenguaje Formal Generado por una Gramática Regular y reconocido por un Autómata Finito	Generado por una GT3 / Gramática Regular
Lenguaje independien-te del Contexto	Es un Lenguaje Formal Generado por una Gramática Independiente del Contexto y reconocido por un Autómata con Pila	Generado por una GT2 / Gramática Independiente del Contexto
Lenguaje Dependiente del Contexto	Es un Lenguaje Formal Generado por una Gramática Dependiente del Contexto y reconocido por un Autómata Linealmente Acotado	Generado por una GT1 / Gramática Dependiente del Contexto
Lenguaje Estructurado por Frases	Es un Lenguaje Formal Generado por una Gramática Estructurada por frases y reconocido por una Máquina de Turing	Generado por una GT0 / Gramática Estructurada por Frases
Entrada	Símbolos que formarán parte del Alfabeto de entrada Formarán parte de las Palabras Aceptadas por las Máquinas Abstractas Formarán parte en la definición de las Transiciones	Definirse a partir del Máquina Abstracta a construir Incluirse en el Alfabeto de Entrada
Estados	Símbolos perteneciente al conjunto de estados que define la situación en que se encuentra la Máquina Abstracta en un intervalo de tiempo determinado	Definirse a partir del Máquina Abstracta a construir Incluirse en el Conjunto de estados Incluirse en el Conjunto de Estados Finales Ser el Estado Inicial
Alfabeto de entrada	Colección de símbolos de Entrada	Definirse a partir de la Máquina Abstracta a construir
Conjunto de estados	Colección finita de Estados por los que puede transitar una Máquina Abstracta	Definirse a partir de la Máquina Abstracta a construir
Conjunto de Estados	Subconjunto del Conjunto de Estados	Definirse a partir de la Máquina Abstracta a construir

Finales	Al finalizar de procesar una cadena en la Entrada si la Máquina Abstracta queda posicionado en un Estado perteneciente a este conjunto, tenemos una Aceptación de Cadenas	
Alfabeto de Pila	Colección de símbolos de que se pueden grabar en la Pila de una Autómata con Pila	Definirse a partir del Autómata con Pila a construir
Símbolo inicial de Pila	Símbolo Perteneciente al Alfabeto con Pila, que se utiliza como marca o fondo de la Pila	Definirse a partir del Autómata con Pila a construir
Estado Inicial	Estado por donde comenzará la Máquina abstracta para la Aceptación de cadena	Origen de toda Máquina Abstracta
Estado final de Aceptación	Estado incluido en el Conjunto de estados Finales Si al detenerse la Máquina Abstracta, queda posicionado en este Estado se produce la Aceptación de cadenas	Definirse a partir de la Máquina Abstracta a construir
Función de Transición	Es una función que permite la Transición de un Estado a otro dentro del Conjunto de estados	Son utilizadas en el procedimiento de Aceptación de cadenas Forman parte de las máquinas abstractas
Máquinas Abstractas	Tienen la capacidad de aceptar cadenas dispuestas en su entrada.	Aceptación de Cadenas de Lenguajes Formales
MT / Máquina de Turing	Es una Máquina Abstracta. Queda definida por una quíntupla a través de: Alfabeto de entrada , Conjunto de Estados , Estado Inicial, Conjunto de Estados Finales , Función de Transición . Las Transiciones son del tipo **f: Q x ∑ → Q x ∑ x M)** Estando en un estado y ante un valor leído en la cinta, cambia de estado, graba en la cinta y realiza un movimiento del cabezal. La cinta de entrada es infinita al menos en uno de los extremos	Aceptación de Cadenas de Lenguaje Estructurado por Frases
ALA – Autómata Linealmente Acorado	Es una Máquina Abstracta. Queda definida por una quíntupla a través de: Alfabeto de entrada , Conjunto de Estados , Estado Inicial, Conjunto de Estados Finales , Función de Transición . Las Transiciones son del tipo	Aceptación de Cadenas de Lenguajes Dependientes del Contexto

	f: Q x ∑ → Q x ∑ x M) Estando en un estado y ante un valor leído en la cinta, cambia de estado, graba en la cinta y realiza un movimiento del cabezal. La cinta de entrada es acotada, o sea que la función de transición, no puede realizar movimientos por fuera de las celdas asignadas	
AP / Autómata con Pila	Es una Máquina Abstracta. Queda definida por una séptupla a través de: Alfabeto de entrada , Alfabeto de Pila , Conjunto de Estados , Estado Inicial , Símbolo inicial de Pila , Conjunto de Estados Finales , Función de Transición. Las Transiciones son del tipo **f: Q x (∑ U { λ}) x A → P (Q xA)** Indica que una transición de estado, se dará: cuando estando en un estado, ante la presencia de un símbolo de entrada o la palabra vacía (λ), y habiendo un elemento en la cima de la pila, se produce la transición de estado y se realizará una operación de pila.	Aceptación de Cadenas de Lenguajes Independientes del Contexto
AF – Autómata Finito	Es una Máquina Abstracta. Queda definida por una quíntupla a través de: Alfabeto de entrada , Conjunto de Estados , Estado Inicial, Conjunto de Estados Finales , Función de Transición . Las Transiciones son del tipo $f: Q \times \Sigma \rightarrow Q$ Lo que significa que estando en un estado ante una entrada cambia de estado	Aceptación de Cadenas de Lenguajes Regulares Posibilita la definición de la Gramática Regular equivalente
Isomorfismo / Equivalencia	Es una función biyectiva que permite la conversión de Máquinas Abstractas a Gramáticas Formales y viceversa	Conversión de una Gramática Regular a Autómata Finito. Conversión de un Autómata Finito a Gramática Regular.

		Conversión de una Gramática Independiente de Contexto a Autómata con Pila.

Verbo	*Nociones:* describen quien ejecuta la acción, cuando ocurre, y cuáles son los procedimientos involucrados.
	Impactos: describen las restricciones sobre la acción, cuáles son las acciones desencadenadas en el ambiente y las nuevas situaciones que aparecen como resultado de la acción.

Verbos		
Entrada	Nociones	Impactos
Derivación Directa / Derivaciones Directas	Es realizada por el Operador cuando aplicando una producción se transforma una Cadena en Otra Los símbolos que se encuentran a la derecha de una reglas de producción son reemplazados en la tira por las que se encuentran a la Izquierda de la misma Producción	La producción a aplicar debe pertenecer al conjunto de Producciones de la Gramática Formal. La cadena resultante puede contener otros símbolos a reemplazar
Generación de Cadenas	Es realizada por el Operador Obtener una cadena de Símbolos terminales efectuando sucesivas derivaciones directas desde el axioma inicial de la Gramática Formal	Se parte desde el Axioma Inicial Se pueden aplicar una o mas Derivaciones Directas El resultado debe ser una cadena de Símbolos Terminales La cadena de símbolos Terminales obtenida es parte del Lenguaje Formal Generado por la Gramática Formal
Derivación	Es la aplicación en forma sucesiva de varias Derivaciones Directas	Los símbolos a reemplazar en la cadena deben existir a la derecha de una regla de producción
Transición	Es el paso de un Estado a otro dentro del Conjunto de estados	Permite cambiar de Estados dependiendo de una entrada Forman parte de la Función de transición
Aceptación de cadenas	Es realizada por el Operador Dada una entrada formada por Símbolos del Alfabeto de entrada , se realizan Transiciones dentro de la máquina Abstracta	Se parte desde el Estado Inicial Se aplican transiciones de la función de transición Para considerar que una cadena es aceptada la máquina abstracta debe quedar posicionada en un estado perteneciente al conjunto de estados de aceptación, y no restar leer nada en la entrada, o detenerse
Conversión de una Gramática Regular a Autómata	Es realizada por la existencia de Isomorfismo. Es realizada cuando se quiere obtener el Autómata Finito a partir de una Gramática	Debe partirse de una Gramática Regular lineal por derecha. Se obtiene un Autómata Finito no determinista

Finito.	Regular Procedimiento: El Conjunto de Símbolos Terminales de la Gramática Regular para a ser el Alfabeto de entrada del Autómata Finito. El Conjunto de Estados del Autómata Finito será el Conjunto de Símbolos No Terminales de la Gramática Regular mas un nuevo símbolo especial que además será el único estado del conjunto de estados finales de aceptación El Axioma Inicial de la Gramática Regular pasará a ser el estado Inicial del Autómata Finito Construcción de la Función de Transición. A partir del Conjunto de Producciones Si la producción tienen la forma: A:= aB entonces f(A , a) = B A:= a entonces f(A, a) = F S:=Lambda entonces f(A, a) = F	
Conversión de un Autómata Finito a Gramática Regular.	Es realizada por la existencia de Isomorfismo. Es realizada cuando se quiere obtener una Gramática Regular a partir de un Autómata Finito Procedimiento: El Alfabeto de entrada del Autómata Finito. Pasa a ser el Conjunto de Símbolos Terminales de la Gramática Regular El Conjunto de Estados del Autómata Finito pasa a ser el Conjunto de Símbolos No Terminales de la Gramática Regular El estado Inicial del Autómata Finito pasa a ser el Axioma	Debe partirse de un Autómata Finito. Se construye una Gramática Regular lineal por derecha.

	Inicial de la Gramática Regular Construcción del Conjunto de Producciones a partir de la Función de Transición. Si la Transición tiene la forma: f(q , a) = p entonces q := ap Si p es además perteneciente al conjunto de estados finales de aceptación entonces q := a Si el estado inicial q0 pertenece al conjunto de estados finales de aceptación entonces el axioma inicial de la Gramática Regular S := Lambda	
Conversión de una Gramática Independiente de Contexto a Autómata con Pila.	Es realizada por la existencia de Isomorfismo. Es realizada cuando se quiere obtener el Autómata a pila a partir de una Gramática Independiente del Contexto Procedimiento: El Conjunto de Símbolos Terminales de la Gramática Independiente del Contexto pasa a ser el Alfabeto de entrada del Autómata a Pila. El Alfabeto de Pila del Autómata a Pila se obtiene con la Unión de los Conjunto de Símbolos Terminales , Conjunto de Símbolos No Terminales y un símbolo especial de pila que marca el fondo de la pila El Conjunto de estados del Autómata a Pila tendrá 4 estados: Un estado Inicial (t), un estado final de aceptación (r) y dos estados intermedios (p,q) Se definen transiciones: Transición inicial marca el fondo de pila f(t,λ,λ) --> (p, #) Transición al estado operativo,	Debe partirse de una Gramática Independiente del Contexto Se obtiene un Autómata a Pila Por medio del método de análisis descendente

	con inserción del Axioma Inicial De la Gramática Formal f(p,λ,#) --> (q, #S) Por cada producción C := w se define una transición f(q,λ,C) --> (q, w) por cada Símbolo terminal del conjunto de símbolos terminales de la GT2 se define una transición para el vaciado de pila f(q,x,x) --> (q, λ) Se define una transición que vacía la pila y pasa al estado final de aceptación f(q, λ, #) --> (r, λ)	

Estado	*Nociones:* describen que significa y que acciones pueden desencadenarse como consecuencia de ese estado.
	Impactos : describen otras situaciones y acciones relacionadas

Estados		
Entrada	Nociones	Impactos
Lambda	Es una cadena especial cuya longitud es igual a cero	Pueden utilizarse en las producciones para la Generación de Cadenas

Notas Aclaratorias:

- Rojo Corresponde a Gramáticas
- Azul Corresponde a Máquinas
- Verde Corresponde a los Lenguajes e isomorfismos
- Las entradas al LEL marcadas en color Naranja corresponden a la primer versión de trabajo en la Herramienta BMW sobre Gramáticas y Lenguajes
- Las entradas al LEL marcadas en color Fucsia corresponden a la segunda versión de trabajo en la Herramienta BMW adicionando las máquinas Abstractas

- Las entradas al LEL marcadas en color Turquesa corresponden a la Tercer versión de trabajo en la Herramienta BMW adicionando los Isomorfismos entre Máquinas Abstractas y Gramáticas Formales

2. Proceso de construcción de escenarios

En la 1ra versión del BMW Gramáticas y Lenguajes
En la Segunda versión del BMW se agregan las máquinas Abstractas
En la Tercer versión del BMW se agregan los Isomorfismos y conversiones

2.1. Identificación de los actores de la aplicación.

Actores: sujeto en el LEL

Actores Primarios

Operador

2.2. Generación de la lista de escenarios candidatos, a partir de los actores Principales.

Escenarios Candidatos:

Creación de Gramática Formal

Generación de Cadenas
Identificación de Tipo de Gramática Formal

Generación de Lenguajes Formales

Generación de Lenguaje estructurado por frase.

Generación de Lenguaje dependiente del contexto.
Generación de Lenguaje Independiente del contexto.

Generación de Lenguaje Regular.

Creación de máquinas Abstractas

Creación de Máquina de Turing

Creación de Autómata Linealmente Acotado

Creación de Autómata con Pila

Creación de Autómata Finito

Reconocimiento de cadenas de entrada

Conversión de una Gramática Regular a Autómata Finito.

Conversión de un Autómata Finito a Gramática Regular.

Conversión de una Gramática Independiente de Contexto a Autómata con Pila.

Nota: Rojo GF – Azul MA – Verde Isomorfismo

2.3. Descripción de los escenarios candidatos, provenientes de los actores principales.

Descripción de Escenarios:

Gramáticas Formales

Título	Creación de Gramáticas Formales
Objetivo	Creación de una Gramática Formal
Contexto	Lenguajes Formales
Recursos	Conjunto de Símbolos Terminales, Conjunto de Símbolos No Terminales, Axioma Inicial , Conjunto de Producciones
Actores	Operador
Episodios	El Operador define el Conjunto de Símbolos Terminales El Operador define el Conjunto de Símbolos No Terminales El Operador define el Axioma Inicial El Operador define el Conjunto de reglas de producciones

Título	Generación de Cadenas
Objetivo	Obtención de Cadenas Generadas de un Lenguaje Formal a través de una Gramática Formal
Contexto	Lenguajes Formales
Recursos	Gramáticas Formales, producciones, derivaciones directas, Conjunto de Símbolos Terminales, Conjunto de Símbolos No Terminales, Axioma Inicial , Conjunto de Producciones
Actores	Operador
Episodios	El operador toma en axioma inicial de la gramática formal Selecciona una producción del conjunto de reglas de producción de la Gramática Formal y comienza a realizar derivaciones directas

	El proceso finaliza cuando la cadena no tiene mas Símbolos No terminales a reemplazar y se convierte en una de las Cadenas generadas

Título	Identificación del tipo de Gramática Formal
Objetivo	Determinar el tipo de Gramática Formal
Contexto	Lenguajes Formales
Recursos	Gramáticas Formales, producciones, , Conjunto de Producciones
Actores	Operador
Episodios	El operador toma el Conjunto de producciones de la Gramática formal Selecciona una a una cada producción del conjunto de reglas de producción de la Gramática Formal y verifica el lado izquierdo y derecho de las mismas para determinar el tipo de gramática formal que se trata De acuerdo a la mayor restricción encontrada entre todas las producciones del conjunto de reglas de producción será el tipo de gramática formal

Título	Generación de Lenguajes Formales
Objetivo	Generar el Lenguaje Formal
Contexto	Lenguajes Formales
Recursos	Gramáticas Formales , GT0 , GT1 , GT2 , GT3
Actores	Operador
Episodios	El operador realiza la Identificación del tipo de gramática Formal IF Gramática Formal es GTO THEN Generación de Lenguaje estructurado por frase . IF Gramática Formal es GT1 THEN Generación de Lenguaje dependiente del contexto. IF Gramática Formal es GT2 THEN Generación de Lenguaje Independiente del contexto. IF Gramática Formal es GT3 THEN Generación de Lenguaje Regular.

Subescenarios GF

Título	Generación de Lenguaje estructurado por frase.
Objetivo	Generar el Lenguaje Estructurado por frases
Contexto	Lenguajes Formales
Recursos	GT0 , derivaciones directas
Actores	Operador
Episodios	El operador realiza la generación de cadenas de todas las cadenas de la GT0

Título	Generación de Lenguaje Dependiente del Contexto
Objetivo	Generar el Lenguaje Dependiente de contexto
Contexto	Lenguajes Formales
Recursos	GT1 , derivaciones directas
Actores	Operador
Episodios	El operador realiza la generación de cadenas de todas las cadenas de la GT1

Título	Generación de Lenguaje Independiente del contexto
Objetivo	Generar el Lenguaje Independiente de contexto
Contexto	Lenguajes Formales
Recursos	GT2 , derivaciones directas
Actores	Operador
Episodios	El operador realiza la generación de cadenas de todas las cadenas de la GT2

Título	Generación de Lenguaje Regular
Objetivo	Generar el Lenguaje Regular
Contexto	Lenguajes Formales
Recursos	GT3 , derivaciones directas
Actores	Operador
Episodios	El operador realiza la generación de cadenas de todas las cadenas de la GT3

Máquinas Abstractas

Título	Creación de Máquinas Abstractas
Objetivo	Crear la Máquina Abstracta
Contexto	Lenguajes Formales
Recursos	Máquinas Abstractas Máquina de Turing

	Autómata Linealmente Acotado Autómata con Pila Autómata Finito
Actores	Operador
Episodios	El operador realiza identifica el tipo de Máquina Abstracta a construir IF Máquina Abstracta es una Máquina de Turing THEN Creación de Máquina de Turing. IF Máquina Abstracta es un Autómata linealmente Acotado THEN Creación de Autómata Linealmente Acotado. IF Máquina Abstracta es un Autómata con Pila THEN Creación de Autómata con Pila. IF Máquina Abstracta es un Autómata Finito THEN Creación de Autómata Finito.

Título	Reconocimiento de Cadenas
Objetivo	Determinar si una Cadena es reconocida por una Máquina Abstracta
Contexto	Lenguajes Formales
Recursos	Cadenas Máquinas Abstractas Máquina de Turing Autómata Linealmente Acotado Autómata con Pila Autómata Finito
Actores	Operador
Episodios	El operador toma la cadena a procesar. Para cada uno de los símbolos de entrada realiza las transiciones de estados que correspondan de acuerdo a la función de transición de la Máquina Abstracta Cuando no resten símbolos de entrada a procesar en la cadenas de entrada si la máquina abstracta que corresponda queda posicionada en un estado perteneciente al conjunto de estados Finales, la cadena es aceptada

Subescenarios MA

Título	Creación de Máquina de Turing.
Objetivo	Construir la Máquina de Turing
Contexto	Lenguajes Formales
Recursos	Alfabeto de entrada , Conjunto de Estados , Estado Inicial, Conjunto de Estados Finales , Función de Transición .

Actores	Operador
Episodios	El operador define el Alfabeto de entrada El operador define el Conjunto de Estados El operador define el Estado Inicial El operador define el Conjunto de Estados Finales El operador define la Función de Transición

Título	Creación de Autómata Linealmente acotado
Objetivo	Construir el Autómata Linealmente Acotado
Contexto	Lenguajes Formales
Recursos	Alfabeto de entrada , Conjunto de Estados , Estado Inicial, Conjunto de Estados Finales , Función de Transición .
Actores	Operador
Episodios	El operador define el Alfabeto de entrada El operador define el Conjunto de Estados El operador define el Estado Inicial El operador define el Conjunto de Estados Finales El operador define la Función de Transición

Título	Creación de Autómata con Pila
Objetivo	Construir el Autómata con Pila
Contexto	Lenguajes Formales
Recursos	Alfabeto de entrada , Alfabeto de Pila, Conjunto de Estados , Estado Inicial, Símbolo Inicial de Pila, Conjunto de Estados Finales , Función de Transición .
Actores	Operador
Episodios	El operador define el Alfabeto de entrada El operador define el Alfabeto de Pila El operador define el Conjunto de Estados El operador define el Estado Inicial El operador define el Símbolo inicial de Pila El operador define el Conjunto de Estados Finales El operador define la Función de Transición

Título	Creación de Autómata Finito
Objetivo	Construir el Autómata Finito
Contexto	Lenguajes Formales
Recursos	Alfabeto de entrada , Conjunto de Estados , Estado Inicial, Conjunto de Estados Finales , Función de Transición .
Actores	Operador

Episodios	El operador define el Alfabeto de entrada El operador define el Conjunto de Estados El operador define el Estado Inicial El operador define el Conjunto de Estados Finales El operador define la Función de Transición

Isomorfismos

Título	Conversión de una Gramática Regular a Autómata Finito
Objetivo	Obtención de un Autómata Finito a partir de una Gramática regular
Contexto	Lenguajes Formales
Recursos	Gramática Regular , Autómata Finito
Actores	Operador
Episodios	El operador toma la gramática Formal y realiza el proceso de conversión a Autómata Finito en donde: El Conjunto de Símbolos Terminales de la Gramática Regular para a ser el Alfabeto de entrada del Autómata Finito. El Conjunto de Estados del Autómata Finito será el Conjunto de Símbolos No Terminales de la Gramática Regular más un nuevo símbolo especial que además será el único estado del conjunto de estados finales de aceptación El Axioma Inicial de la Gramática Regular pasará a ser el estado Inicial del Autómata Finito Construcción de la Función de Transición. A partir del Conjunto de Producciones Si la producción tienen la forma: A:= aB entonces f(A , a) = B A:= a entonces f(A, a) = F S:=Lambda entonces f(A, a) = F

Título	Conversión de un Autómata Finito a Gramática Regular
Objetivo	Obtención de una Gramática Regular a partir de un Autómata Finito
Contexto	Lenguajes Formales
Recursos	Gramática Regular , Autómata Finito
Actores	Operador
Episodios	El operador toma el Autómata Finito y realiza el proceso de conversión a gramática Formal en donde:

	El Alfabeto de entrada del Autómata Finito. Pasa a ser el Conjunto de Símbolos Terminales de la Gramática Regular El Conjunto de Estados del Autómata Finito pasa a ser el Conjunto de Símbolos No Terminales de la Gramática Regular El estado Inicial del Autómata Finito pasa a ser el Axioma Inicial de la Gramática Regular Construcción del Conjunto de Producciones a partir de la Función de Transición. Si la Transición tiene la forma: f(q , a) = p entonces q := ap Si p es además perteneciente al conjunto de estados finales de aceptación entonces q := a Si el estado inicial q0 pertenece al conjunto de estados finales de aceptación entonces el axioma inicial de la Gramática Regular S := Lambda

Título	Conversión de una Gramática Independiente de Contexto a Autómata con Pila.
Objetivo	Obtención de un Autómata con Pila a partir de una Gramática Independiente del Contexto
Contexto	Lenguajes Formales
Recursos	Gramática Independiente del Contexto , Autómata con Pila
Actores	Operador
Episodios	El operador toma la gramática Independiente del Contexto y realiza el proceso de conversión a Autómata con Pila en donde: El Conjunto de Símbolos Terminales de la Gramática Independiente del Contexto pasa a ser el Alfabeto de entrada del Autómata a Pila. El Alfabeto de Pila del Autómata a Pila se obtiene con la Unión de los Conjunto de Símbolos Terminales , Conjunto de Símbolos No Terminales y un símbolo especial de pila que marca el fondo de la pila El Conjunto de estados del Autómata a Pila tendrá 4 estados: Un estado Inicial (t), un estado final de aceptación (r) y dos estados intermedios (p,q) Se definen transiciones: Transición inicial marca el fondo de pila f(t,λ,λ) --> (p, #)

	Transición al estado operativo, con inserción del Axioma Inicial De la Gramática Formal f(p,λ,#) --> (q, #S) Por cada producción C := w se define una transición f(q,λ,C) --> (q, w) por cada Símbolo terminal del conjunto de símbolos terminales de la GT2 se define una transición para el vaciado de pila f(q,x,x) --> (q, λ) Se define una transición que vacía la pila y pasa al estado final de aceptación f(q, λ, #) --> (r, λ)

2.4. Ampliación de la lista de escenarios candidatos, a partir de los actores secundarios.

No generan nuevos escenarios

2.5. Descripción de los escenarios candidatos, provenientes de actores secundarios.

No Hay

3. Proceso de construcción de tarjetas CRC

Las Tarjetas CRC se obtienen a través de la derivación del LEL y escenarios, pudiendo dividirse en tres partes

3.1. Encontrar CRC primarias.

Los actores de los escenarios son candidatos a convertirse en CRC primarias

CRC Primaria	Operador
Responsabilidades	Creación de una Gramática Formal a través de la definición del Conjunto de Símbolos Terminales , Conjunto de Símbolos No Terminales , axioma inicial y definición de las producciones del conjunto de reglas de producción Identificación de gramática formal , en el tipo de gramática formal que corresponda GT0, GT1, GT2, GT3 Generación de Cadenas de Lenguajes Formales

	Creación de máquinas Abstractas ya sean Máquinas de Turing , Autómatas Linealmente Acotados, Autómatas con Pila o Autómatas finitos. Definiendo sus componentes Alfabeto de entrada , Alfabeto de Pila , Conjunto de Estados , Estado Inicial , Símbolo inicial de Pila , Conjunto de Estados Finales , Función de Transición. Dada una Cadena de Entrada efectuar transiciones entre estados para Reconocer cadenas de entrada Conversión de una Gramática Regular a Autómata Finito. Conversión de un Autómata Finito a Gramática Regular. Conversión de una Gramática Independiente de Contexto a Autómata con Pila.
Colaboraciones	Gramática Formal , Conjunto de Símbolos Terminales , Conjunto de Símbolos No Terminales , axioma inicial ,producciones , conjunto de reglas de producción , GT0, GT1, GT2, GT3 Generación de Cadenas , Lenguaje Formal Máquinas Abstractas , Máquinas de Turing , Autómatas Linealmente Acotados, Autómatas con Pila , Autómatas finitos. , Alfabeto de entrada , Alfabeto de Pila , Conjunto de Estados , Estado Inicial , Símbolo inicial de Pila , Conjunto de Estados Finales , Función de Transición. Cadena de Entrada, transiciones , estados Reconocer cadenas de entrada

3.2.a. Encontrar CRC secundarias.

Son aquellos colaboradores de las CRC primarias que las ayudan a cumplir con sus responsabilidades

Gramáticas Formales

CRC Secundaria	Gramática Formal
Responsabilidades	Generar Lenguaje Formal
Colaboraciones	Lenguaje Formal

CRC Secundaria	Conjunto de Símbolos Terminales
Responsabilidades	Definir los Símbolos terminales que constituirán el conjunto a ser utilizado dentro de la Gramática Formal para generar los Lenguajes Formales
Colaboraciones	Símbolos terminales , Gramática Formal , Lenguajes Formales

CRC Secundaria	Conjunto de Símbolos No Terminales
Responsabilidades	Definir los Símbolos No terminales que constituirán el conjunto a ser utilizado dentro de la Gramática Formal para generar los Lenguajes Formales
Colaboraciones	Símbolos No terminales , Gramática Formal , Lenguajes Formales

CRC Secundaria	Axioma Inicial
Responsabilidades	Origen de toda Generación de cadenas de los Lenguajes Formales
Colaboraciones	Generación de Cadenas , Lenguajes Formales

CRC Secundaria	Producciones
Responsabilidades	Son Utilizadas en una derivación
Colaboraciones	Derivación

CRC Secundaria	Conjunto de Reglas de Producciones
Responsabilidades	Formarán parte de la definición de una Gramática Formal
Colaboraciones	Gramática Formal

CRC Secundaria	GT0
Responsabilidades	Generación de Cadenas de Lenguaje estructurado por frases
Colaboraciones	Generación de Cadenas , Lenguaje estructurado por frases

CRC Secundaria	GT1
Responsabilidades	Generación de Cadenas de Lenguaje Dependiente del Contexto
Colaboraciones	Generación de Cadenas , Lenguaje Dependiente del Contexto

CRC Secundaria	GT2
Responsabilidades	Generación de Cadenas de Lenguaje Independiente del Contexto Posibilita el isomorfismo con la definición del Autómata con Pila equivalente
Colaboraciones	Generación de Cadenas , Lenguaje Independiente del Contexto , isomorfismo , Autómata con Pila

CRC Secundaria	GT3
Responsabilidades	Generación de Cadenas de Lenguaje Regulares Posibilita el isomorfismo con la definición del Autómata Finito

	equivalente
Colaboraciones	Generación de Cadenas , Lenguaje Regular, isomorfismo , Autómata Finito

CRC Secundaria	Generación de Cadenas
Responsabilidades	Se parte del axioma Inicial y se aplican una o mas derivaciones directas El resultado será una Cadena de símbolos terminales que es parte del Lenguaje formal generado por la Gramática Formal
Colaboraciones	axioma Inicial, derivaciones directas, Cadena , símbolos terminales , Lenguaje formal , Gramática Formal

CRC Secundaria	Lenguaje Formal
Responsabilidades	Conjunto de Cadenas Generadas a través de una Gramática Formal
Colaboraciones	Cadenas, Gramática Formal

Máquinas Abstractas

CRC Secundaria	Máquinas Abstractas
Responsabilidades	Aceptación de Cadenas de Lenguajes Formales
Colaboraciones	Aceptación de Cadenas , Lenguajes Formales

CRC Secundaria	Máquinas de Turing
Responsabilidades	Aceptación de Cadenas de Lenguaje Estructurado por Frases
Colaboraciones	Aceptación de Cadenas , Lenguaje Estructurado por Frases

CRC Secundaria	Autómatas Linealmente Acotados
Responsabilidades	Aceptación de Cadenas de Lenguajes Dependientes del Contexto
Colaboraciones	Aceptación de Cadenas , Lenguajes Dependientes del Contexto

CRC Secundaria	Autómatas con Pila
Responsabilidades	Aceptación de Cadenas de Lenguajes Independientes del Contexto
Colaboraciones	Aceptación de Cadenas , Lenguajes Independientes del Contexto

CRC Secundaria	Autómatas finitos
Responsabilidades	Aceptación de Cadenas de Lenguajes Regulares Posibilita la definición de la Gramática Regular equivalente
Colaboraciones	Aceptación de Cadenas , Lenguajes Regulares Gramática Regular , equivalente

CRC Secundaria	Alfabeto de entrada
Responsabilidades	Definirse a partir de la Máquina Abstracta a construir
Colaboraciones	Máquina Abstracta

CRC Secundaria	Alfabeto de Pila
Responsabilidades	Definirse a partir del Autómata con Pila a construir
Colaboraciones	Autómata con Pila

CRC Secundaria	Conjunto de Estados
Responsabilidades	Definirse a partir de la Máquina Abstracta a construir
Colaboraciones	Máquina Abstracta

CRC Secundaria	Estado Inicial
Responsabilidades	Origen de toda Máquina Abstracta
Colaboraciones	Máquina Abstracta

CRC Secundaria	Estado Final de Aceptación
Responsabilidades	Definirse a partir de la Máquina Abstracta a construir
Colaboraciones	Máquina Abstracta

CRC Secundaria	Símbolo inicial de Pila
Responsabilidades	Definirse a partir del Autómata con Pila a construir
Colaboraciones	Autómata con Pila

CRC Secundaria	Función de Transición
Responsabilidades	Son utilizadas en el procedimiento de Aceptación de cadenas Forman parte de las máquinas abstractas
Colaboraciones	Aceptación de cadenas máquinas abstractas

CRC Secundaria	Entrada
Responsabilidades	Definirse a partir del Máquina Abstracta a construir Incluirse en el Alfabeto de Entrada
Colaboraciones	Alfabeto de entrada máquinas abstractas

CRC Secundaria	Transiciones
Responsabilidades	Permite cambiar de Estados dependiendo de una entrada Forman parte de la Función de transición
Colaboraciones	Estados Función de transición

CRC Secundaria	Estados
Responsabilidades	Definirse a partir del Máquina Abstracta a construir Incluirse en el Conjunto de estados Incluirse en el Conjunto de Estados Finales Ser estado inicial
Colaboraciones	Máquina Abstracta Conjunto de estados Conjunto de Estados Finales , Estado Inicial

ANEXO VI: ONTOLOGÍAS/PROTÉGÉ -: APLICACIÓN DE LA METODOLOGÍA SELECCIONADA AL MODELO DE GRAMÁTICAS FORMALES Y MAQUINAS ABSTRACTAS

Pasos:

• **1.- Determinar el dominio y alcance de la ontología, además del propósito u objetivo de la misma**,

- ¿Qué dominio cubrirá la ontología?

  Máquinas Abstractas y Gramáticas Formales

- ¿Para qué se va a emplear la ontología?

  Se utilizará para diseñar la implementación de las Máquinas Abstractas y Gramáticas Formales.

- ¿Qué preguntas debería contestar la ontología?

  // definir las preguntas

- ¿Quién usará y mantendrá la ontología?

  Uso: El operador

  Mantenimiento: El operador

• **2.- Considerar la reutilización de ontologías existentes**.

No es pertinente, ya que lo que se pretende evaluar la construcción de un modelo conceptual a partir de la definición de la misma.

• **3.- Enumerar términos importantes de la ontología**.

PALABRA, LENGUAJE, GRAMÁTICA, AUTÓMATA FINITO, SÍMBOLOS TERMINALES, SÍMBOLOS NO TERMINALES, MÁQUINAS ABSTRACTAS, ESTADO INICIAL, ESTADO FINAL, SIMBOLO INICIAL, AUTÓMATA FINITO LINEALMENTE ACOTADO, AUTÓMATA FINITO DETERMINISTA, AUTÓMATA FINITO NO DETERMINISTA, AUTÓMATA FINITO BIDIRECCIONAL, AUTÓMATA FINITO DETERMINISTA, MAQUINA DE TURING, MAQUINA DE TURING BIDIRECCIONAL, MAQUINA DE TURING NO DETERMINISTA, MAQUINA DE TURING UNIVERSAL, REGLA DE PRODUCCION, GRAMATICA DE TIPO 0, GRAMATICA DE TIPO 1, GRAMATICA DE TIPO 2, GRAMATICA DE TIPO 3.

• **4.- Definir las clases y la jerarquía de clases**.

En este punto se comienza con la enumeración de los conceptos o clases más destacadas y posteriormente se generalizan y especializan apropiadamente.

Se comienza identificando de la enumeración de términos, seleccionando aquellos que describan objetos con existencia independiente para constituir los conceptos o clases, en detrimento de aquellos términos que describan cómo son esos objetos.

A continuación se muestran algunas de las clases identificadas:

Gramática Tipo 0
- Gramática Tipo 1
  - Gramática Tipo 2
    - Gramática Tipo 3

Alfabeto
- Alfabeto de Símbolos Terminales
- Alfabeto de Símbolos No Terminales

Maquina Abstracta
- Autómata Finito Determinista
- Autómata Finito No Determinista
- Autómata Linealmente Acotado
- Maquina de Turing
- Autómata a Pila

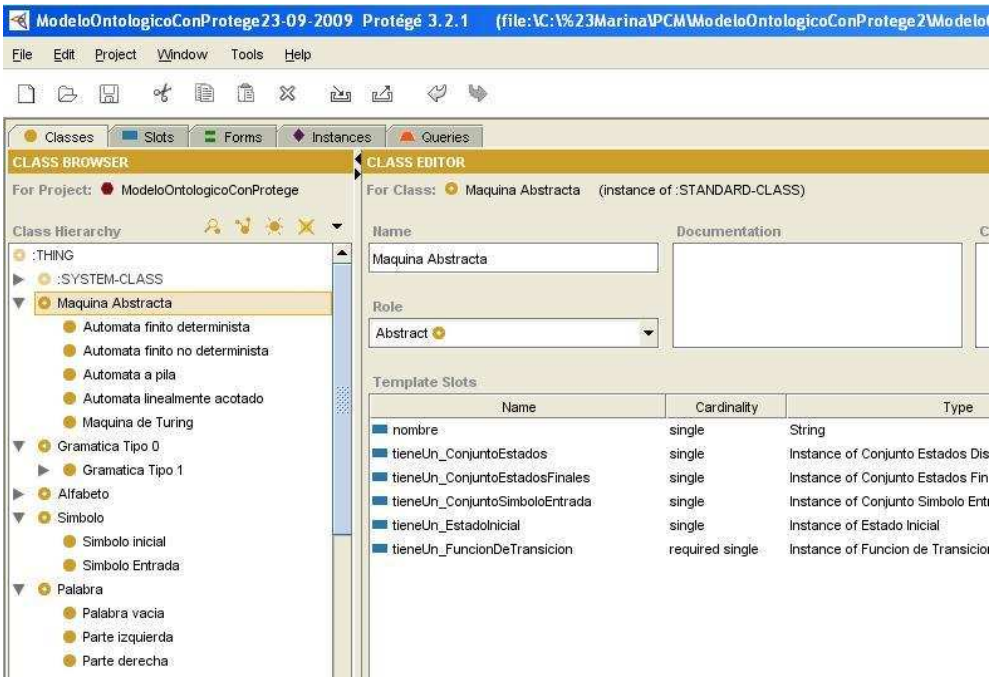
• **5.- Definir las propiedades/slots de las clases**.

// Este paso, es realizado en forma conjunta con las definiciones de las clases. Describimos sus características y estructura. Al definir las clases y sus propiedades, no deberían quedarnos términos entre la lista de términos sin utilizar.

• **6.- Definir las facetas o restricciones de las propiedades o slots**.

PROTÉGÉ: ESTRUCTURA DE CLASES CON SUS SLOTS

A continuación se muestra una pantalla de la herramienta protege en donde a la izquierda pueden apreciarse la jerarquía de clases, y en la parte central la descripción de conceptos de la clases Curso junto con los slots y sus propiedades.



Definiremos ahora cada una de las clases en la herramienta:

Clase: Alfabeto de símbolos no terminales

	Nombre del Slot	Documentación	Tipo	Cardinalidad
	componente_Alfabeto		Palabra, Símbolo	0:1

Clase: Alfabeto de símbolos terminales

	Nombre del Slot	Documentación	Tipo	Cardinalidad
	componente_Alfabeto		Palabra, Símbolo	0:1

Clase: Alfabeto

	Nombre del Slot	Documentación	Tipo	Cardinalidad
	componente_Alfabeto		Palabra, Símbolo	0:1

Clase: Autómata a pila

	Nombre del Slot	Documentación	Tipo	Cardinalidad
	nombre		String	0:1
	reconoce_AAP			0:*
	tieneUn_ConjuntoEstados		Conjunto Estados Disponibles	0:1
	tieneUn_ConjuntoEstadosFinales		Conjunto Estados Finales	0:1
	tieneUn_ConjuntoSimboloEntrada		Conjunto Simbolo Entrada	0:1
	tieneUn_EstadoInicial		Estado Inicial	0:1
	tieneUn_FuncionDeTransicion		Funcion de Transicion	1:1

Clase: Autómata finito determinista

	Nombre del Slot	Documentación	Tipo	Cardinalidad
	nombre		String	0:1
	tieneUn_ConjuntoEstados		Conjunto Estados Disponibles	0:1
	tieneUn_ConjuntoEstadosFinales		Conjunto Estados Finales	0:1
	tieneUn_ConjuntoSimboloEntrada		Conjunto Simbolo Entrada	0:1
	tieneUn_EstadoInicial		Estado Inicial	0:1
	tieneUn_FuncionDeTransicion		Funcion de Transicion	1:1

Clase: Autómata finito no determinista

	Nombre del Slot	Documentación	Tipo	Cardinalidad
	nombre		String	0:1
	tieneUn_ConjuntoEstados		Conjunto Estados Disponibles	0:1
	tieneUn_ConjuntoEstadosFinales		Conjunto Estados Finales	0:1
	tieneUn_ConjuntoSimboloEntrada		Conjunto Simbolo Entrada	0:1
	tieneUn_EstadoInicial		Estado Inicial	0:1
	tieneUn_FuncionDeTransicion		Funcion de Transicion	1:1

Clase: Autómata linealmente acotado

	Nombre del Slot	Documentación	Tipo	Cardinalidad
	nombre		String	0:1
	reconoce_ALA			0:*
	tieneUn_ConjuntoEstados		Conjunto Estados Disponibles	0:1
	tieneUn_ConjuntoEstadosFinales		Conjunto Estados Finales	0:1
	tieneUn_ConjuntoSimboloEntrada		Conjunto Simbolo Entrada	0:1
	tieneUn_EstadoInicial		Estado Inicial	0:1
	tieneUn_FuncionDeTransicion		Funcion de Transicion	1:1

Clase: Conjunto Estados Disponibles

	Nombre del Slot	Documentación	Tipo	Cardinalidad
	tieneUn_EstadoDisponible		Estado Disponible	1:*

Clase: Conjunto Estados Finales

	Nombre del Slot	Documentación	Tipo	Cardinalidad
	tieneUn_Estado Final		Estado Final	1:*

Clase: Conjunto Simbolo Entrada

	Nombre del Slot	Documentación	Tipo	Cardinalidad
	tieneUn_SimboloEntrada		Simbolo Entrada	1:*

Clase: Estado Actual

	Nombre del Slot	Documentación	Tipo	Cardinalidad
	descripcion		String	0:1
	nombre		String	0:1

Clase: Estado Disponible

	Nombre del Slot	Documentación	Tipo	Cardinalidad
	descripcion		String	0:1
	nombre		String	0:1

Clase: Estado Final

	Nombre del Slot	Documentación	Tipo	Cardinalidad
	descripcion		String	0:1
	nombre		String	0:1

Clase: Estado Inicial

	Nombre del Slot	Documentación	Tipo	Cardinalidad
	descripcion		String	0:1
	nombre		String	0:1

Clase: Estado

	Nombre del Slot	Documentación	Tipo	Cardinalidad
	descripcion		String	0:1
	nombre		String	0:1

Clase: Función de Transicion

	Nombre del Slot	Documentación	Tipo	Cardinalidad
	TieneUn_Transicion		Transicion	1:*

Clase: Funcion Transicion Autómata Finito Determinista

	Nombre del Slot	Documentación	Tipo	Cardinalidad
	TieneUn_Transicion		Transicion	1:*

Clase: Funcion Transicion Autómata Finito No Determinista

	Nombre del Slot	Documentación	Tipo	Cardinalidad
	TieneUn_Transicion		Transicion	1:*

Clase: Funcion Transicion Autómata Linealmente Acotado

	Nombre del Slot	Documentación	Tipo	Cardinalidad
	TieneUn_Transicion		Transicion	1:*

Clase: Función Transicion Autómata Pila

	Nombre del Slot	Documentación	Tipo	Cardinalidad
	TieneUn_Transicion		Transicion	1:*

Clase: Funcion Transicion Maquina Turing

	Nombre del Slot	Documentación	Tipo	Cardinalidad
	TieneUn_Transicion		Transicion	1:*

Clase: Gramática Tipo 0

	Nombre del Slot	Documentación	Tipo	Cardinalidad
	componente_Gramatica		Producción, Simbolo inicial, Alfabeto de símbolos terminales, Alfabeto de símbolos no terminales	0:*
	describeEstructura_Gramatica		Frase	0:1
	descripcion		String	0:1
	genera_Gramatica			0:1
	representacionGrafica			0:*

Clase: Gramática Tipo 1

	Nombre del Slot	Documentación	Tipo	Cardinalidad
	componente_Gramatica		Producción, Simbolo inicial, Alfabeto de símbolos terminales, Alfabeto de símbolos no terminales	0:*
	describeEstructura_Gramatica		Frase	0:1
	descripcion		String	0:1
	genera_Gramatica			0:1
	generaA_Tipo0			0:*
	representacionGrafica			0:*

Clase: Gramática Tipo 2

	Nombre del Slot	Documentación	Tipo	Cardinalidad
	componente_Gramatica		Producción, Simbolo inicial, Alfabeto de símbolos terminales, Alfabeto de símbolos no terminales	0:*
	describeEstructura_Gramatica		Frase	0:1
	descripcion		String	0:1
	genera_Gramatica			0:1
	generaA_Tipo0			0:*
	generaA_Tipo1			0:*
	representacionGrafica			0:*

Clase: Gramática Tipo 3

	Nombre del Slot	Documentación	Tipo	Cardinalidad
	caracterizante_Tipo2			0:1
	componente_Gramatica		Producción, Simbolo inicial, Alfabeto de símbolos terminales, Alfabeto de símbolos no terminales	0:*
	describeEstructura_Gramatica		Frase	0:1
	descripcion		String	0:1
	genera_Gramatica			0:1
	generaA_Tipo0			0:*
	generaA_Tipo1			0:*
	generaA_Tipo2			0:*
	representacionGrafica			0:*

Clase: Maquina Abstracta

	Nombre del Slot	Documentación	Tipo	Cardinalidad
	nombre		String	0:1
	tieneUn_ConjuntoEstados		Conjunto Estados Disponibles	0:1
	tieneUn_ConjuntoEstadosFinales		Conjunto Estados Finales	0:1
	tieneUn_ConjuntoSimboloEntrada		Conjunto Simbolo Entrada	0:1
	tieneUn_EstadoInicial		Estado Inicial	0:1
	tieneUn_FuncionDeTransicion		Funcion de Transicion	1:1

Clase: Maquina de Turing

	Nombre del Slot	Documentación	Tipo	Cardinalidad
	nombre		String	0:1
	reconoce_MaquinaDeTuring			0:*
	tieneUn_ConjuntoEstados		Conjunto Estados Disponibles	0:1
	tieneUn_ConjuntoEstadosFinales		Conjunto Estados Finales	0:1
	tieneUn_ConjuntoSimboloEntrada		Conjunto Simbolo Entrada	0:1
	tieneUn_EstadoInicial		Estado Inicial	0:1
	tieneUn_FuncionDeTransicion		Funcion de Transicion	1:1

Clase: Par ordenado

	Nombre del Slot	Documentación	Tipo	Cardinalidad
	componente_ParOrdenado		Parte izquierda, Parte derecha	0:1

Clase: Parte derecha

	Nombre del Slot	Documentación	Tipo	Cardinalidad
	componente_Palabra			0:1

	longitud		Integer	0:1
	nombre		String	0:1

Clase: Parte izquierda

	Nombre del Slot	Documentación	Tipo	Cardinalidad
	componente_Palabra			0:1
	longitud		Integer	0:1
	nombre		String	0:1

Clase: Regla de Producción

	Nombre del Slot	Documentación	Tipo	Cardinalidad
	componente_Produccion		Par ordenado	0:1

Clase: Simbolo Entrada

	Nombre del Slot	Documentación	Tipo	Cardinalidad
	descripcion		String	0:1
	nombre		String	0:1

Clase: Simbolo inicial

	Nombre del Slot	Documentación	Tipo	Cardinalidad
	descripcion		String	0:1
	nombre		String	0:1

Clase: Símbolo

	Nombre del Slot	Documentación	Tipo	Cardinalidad
	descripcion		String	0:1
	nombre		String	0:1

Clase: Transición

	Nombre del Slot	Documentación	Tipo	Cardinalidad
	Entrada en t		Simbolo Entrada	1:1
	Estado en t		Estado Actual	0:1
	Estado en t+1		Estado Disponible	0:1
	Tiempo		Integer	1:1

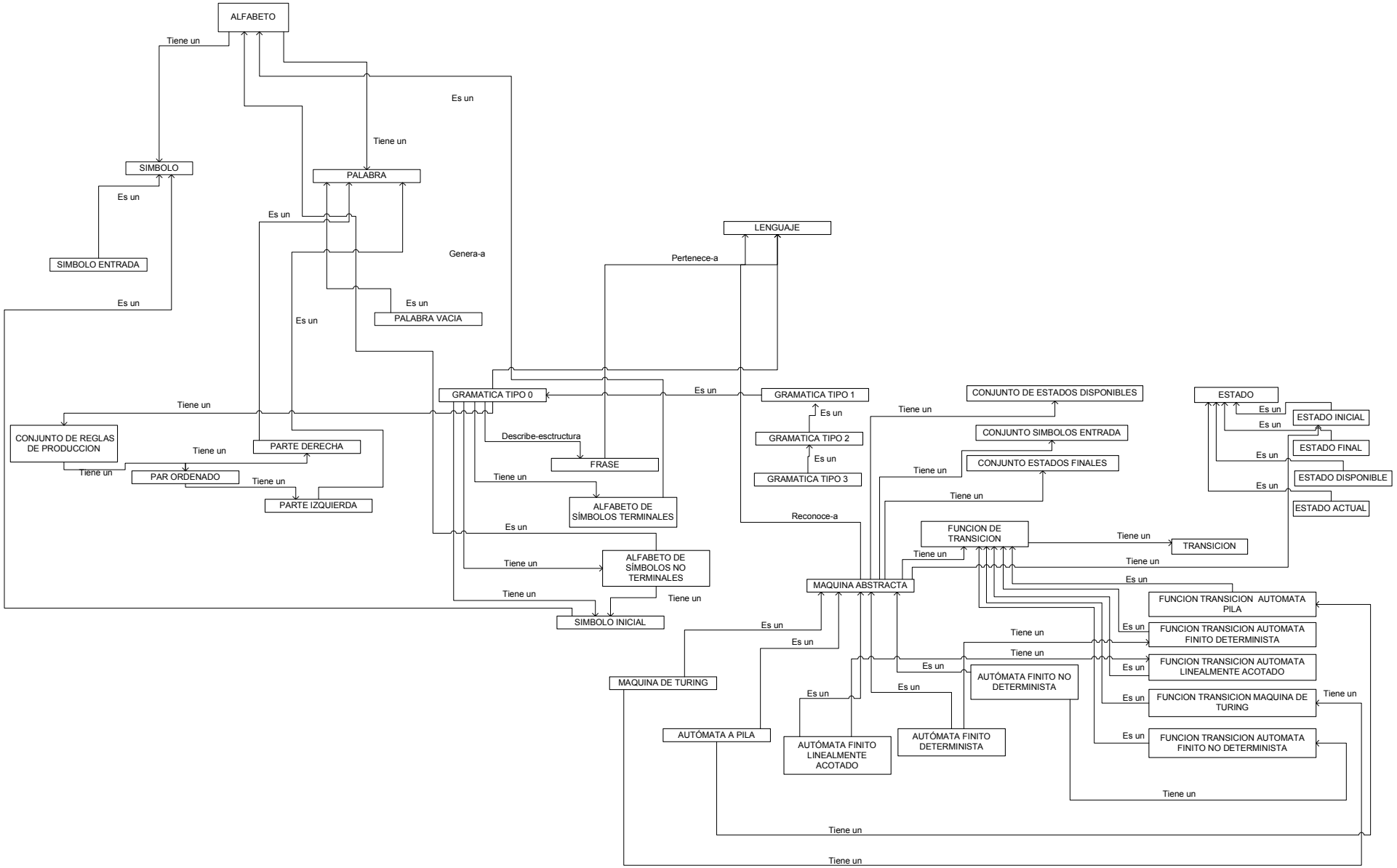
• **7.- Crear instancias**.

// Se crean instancias de las clases para posteriormente verificar las preguntas de competencia.

• **8.- Ejecutar las preguntas de competencia**.

PROTÉGÉ: PREGUNTAS DE COMPETENCIA

// Se debería poner aquí la aclaración acerca de las preguntas de competencia.

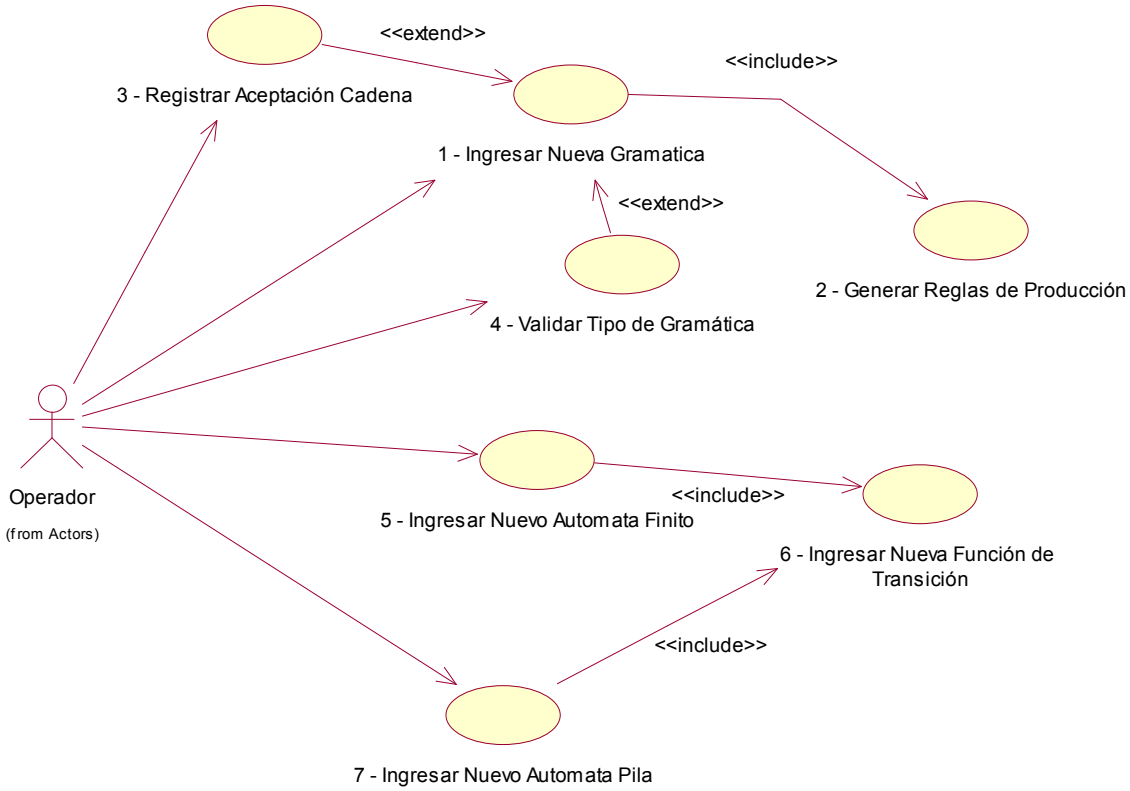

ALFABETO
Tiene un
Es un
Tiene un
SIMBOLO
PALABRA
Es un
Es un
SIMBOLO ENTRADA
Genera-a
LENGUAJE
Pertenece-a
Es un
Es un
Es un
PALABRA VACIA
GRAMATICA TIPO 0
Es un
GRAMATICA TIPO 1
Tiene un
CONJUNTO DE ESTADOS DISPONIBLES
ESTADO
Es un
ESTADO INICIAL
Es un
ESTADO FINAL
Es un
ESTADO DISPONIBLE
Es un
ESTADO ACTUAL
Tiene un
CONJUNTO DE REGLAS DE PRODUCCION
Tiene un
PARTE DERECHA
Describe-esctructura
FRASE
Es un
GRAMATICA TIPO 2
Es un
GRAMATICA TIPO 3
Tiene un
CONJUNTO SIMBOLOS ENTRADA
Tiene un
CONJUNTO ESTADOS FINALES
Tiene un
PAR ORDENADO
Tiene un
PARTE IZQUIERDA
Tiene un
ALFABETO DE SÍMBOLOS TERMINALES
Tiene un
Reconoce-a
Es un
FUNCION DE TRANSICION
Tiene un
TRANSICION
Tiene un
Tiene un
ALFABETO DE SÍMBOLOS NO TERMINALES
Tiene un
MAQUINA ABSTRACTA
Es un
FUNCION TRANSICION AUTOMATA PILA
Tiene un
SIMBOLO INICIAL
Es un
Es un
FUNCION TRANSICION AUTOMATA FINITO DETERMINISTA
Tiene un
Es un
FUNCION TRANSICION AUTOMATA LINEALMENTE ACOTADO
Es un
AUTÓMATA FINITO NO DETERMINISTA
Es un
Es un
Es un
FUNCION TRANSICION MAQUINA DE TURING
Tiene un
MAQUINA DE TURING
Es un
Es un
Es un
AUTÓMATA A PILA
AUTÓMATA FINITO LINEALMENTE ACOTADO
AUTÓMATA FINITO DETERMINISTA
Es un
FUNCION TRANSICION AUTOMATA FINITO NO DETERMINISTA
Tiene un
Tiene un
Tiene un

ANEXO VII: RUP/RATIONAL APLICACIÓN DE LA METODOLOGÍA SELECCIONADA AL MODELO DE GRAMÁTICAS FORMALES Y MAQUINAS ABSTRACTAS

Modelos utilizados para la obtención del modelo conceptual:

- **1.- Modelo de Casos de Uso del Sistema de Información**
- **2.- Trazo Fino Casos de Uso del Sistema de Información**
- **3.- Diagrama de clases**

1. Modelo de casos de uso del sistema de información



2. Trazo Fino casos de uso del sistema de información

Nivel de Caso de Uso:	Negocio ☐	Sistema de información ■				**Orden: 1**
Nombre del Caso de Uso: *Ingresar Nueva Gramática.*						
Objetivo: Ingresar una gramática junto con sus elementos.						
Actor Principal: Operador.		**Actor Secundario:** No aplica.				
Tipo de Caso de Uso :	Concreto ■	Abstracto ☐				
Prioridad :	Alta ☐	Media ■	Baja ☐			
Complejidad:	Alta ☐	Media ■	Baja ☐			
Pre-condiciones	No aplica.					
Post-condiciones	**Éxito:** Se ha ingresado correctamente una gramática.					
	Fracaso: No aplica.					

Curso Normal	**Curso Alternativo**
1. El Caso de Uso (CU) comienza cuando el operador selecciona la opción "Ingresar nueva gramática".	
2. El Sistema solicita que se le ingrese el conjunto de Símbolos No Terminales.	
3. El Operador ingresa el conjunto de Símbolos No Terminales	
4. El Sistema solicita que se le ingrese el conjunto de Símbolos Terminales	
5. El Operador ingresa el conjunto de Símbolos Terminales	
6. El Sistema solicita que se seleccione el Axioma del conjunto de Símbolos No Terminales.	
7. El Operador selecciona el Axioma del conjunto de Símbolos No Terminales.	
8. El Sistema solicita que se le ingrese Reglas de Producción. Se llama al CU "Generar Reglas de Producción".	
9. El sistema Visualiza las Reglas de Producción ingresadas por el Operador	
10. El Operador no selecciona la opción de "Comprobar Cadena".	10.A. El operador selecciona la opción de "Comprobar Cadena". 10.A.1. Se llama al CU "Registrar Aceptación Cadena"
11. El Operador no selecciona la opción "Validar tipo de Gramática"	11.A. El operador selecciona la opción "Validar tipo de Gramática". 11.A.1 Se llama al CU "Validar Tipo de Gramática".
12. Fin CU.	

Use Case que lo extiende: No aplica.		
Use Case incluido en: CU "Generar Reglas de Producción"		
Use Case al cual incluye: No aplica.		
Use Case al cual extiende: No aplica.		
Caso de Uso de Generalización: No aplica.		
Autor: **NF**	Fecha de creación: 07/09/09	Versión
Autor Ultima Modificación: **NF**	Fecha Ultima Modificación: 07/09/09	1.0

Nivel de Caso de Uso:	Negocio ☐	Sistema de información ■				**Orden: 3**
Nombre del Caso de Uso: *Registrar Aceptación de Cadena*						
Objetivo: Registrar la aceptación de una cadena ingresada en un Gramática.						
Actor Principal: Operador.		**Actor Secundario:** No aplica.				
Tipo de Caso de Uso :	Concreto ■	Abstracto ☐				
Prioridad :	Alta ☐	Media ■	Baja ☐			
Complejidad:	Alta ☐	Media ■	Baja ☐			
Pre-condiciones	No aplica.					
Post-condiciones	**Éxito:** Se registró la aceptación de la cadena.					
	Fracaso: No aplica.					
Curso Normal			**Curso Alternativo**			
1. El Caso de Uso (CU) comienza cuando el operador selecciona la opción "Comprobar Cadena"						
2. El Sistema Solicita que se seleccione una Gramática.						
3. El Operador selecciona una Gramática.						
4. El Sistema solicita que se ingrese la cadena a probar.						
5. El Operador selecciona la opción "Probar"						
6. El Sistema Prueba la cadena ingresada y determina si es aceptada o no.						
7. El Sistema informa si la cadena fue aceptada.						
8. Fin CU.						
Use Case que lo extiende: No aplica.						
Use Case incluido en: No aplica.						
Use Case al cual incluye: No aplica.						
Use Case al cual extiende: CU "Ingresar Nueva Gramática".						
Caso de Uso de Generalización: No aplica.						
Autor: **NF**		Fecha de creación: 07/09/09				Versión
Autor Ultima Modificación: **NF**		Fecha Ultima Modificación: 07/09/09				1.0

Nivel de Caso de Uso:	Negocio ☐	Sistema de información ■				**Orden: 4**
Nombre del Caso de Uso: *Validar Tipo de Gramática.*						
Objetivo: Determinar el tipo de Gramática de una ingresada.						
Actor Principal: Operador.		**Actor Secundario:** No aplica.				
Tipo de Caso de Uso :	Concreto ■	Abstracto ☐				
Prioridad :	Alta ☐	Media ■	Baja ☐			
Complejidad:	Alta ☐	Media ■	Baja ☐			
Pre-condiciones	No aplica.					
Post-condiciones	**Éxito:** Se determinó el tipo de Gramática.					
	Fracaso: No aplica.					

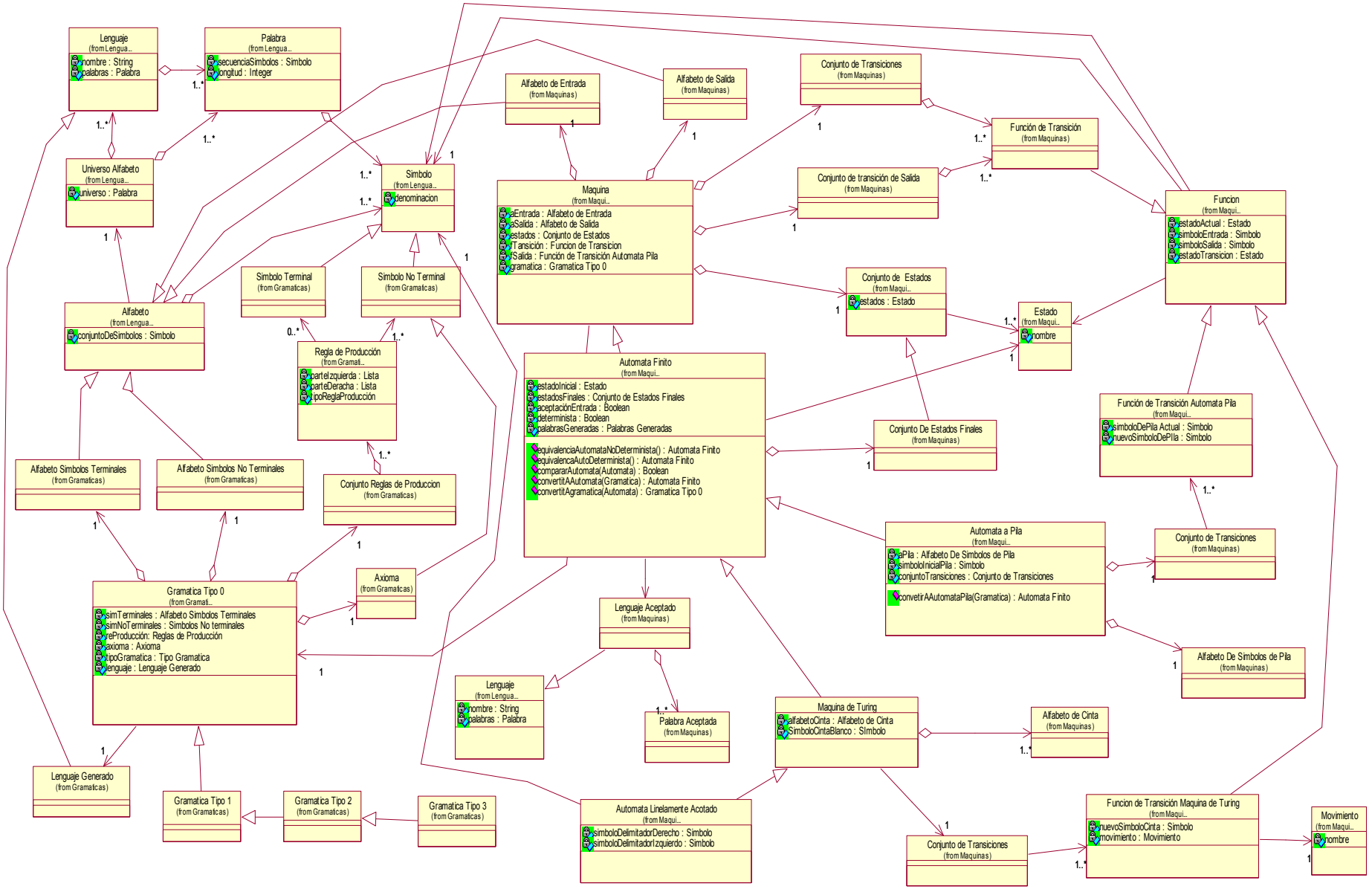
Curso Normal	**Curso Alternativo**
1. El Caso de Uso (CU) comienza cuando el operador selecciona la opción "Validar Gramática"	
2. El Sistema Solicita que se seleccione una Gramática.	
3. El Operador selecciona una Gramática.	
4. El Sistema solicita comprueba el tipo de la Gramática Ingresada.	
5. El Sistema informa el tipo de Gramática.	
6. Fin Cu	

Use Case que lo extiende: No aplica.		
Use Case incluido en: No aplica.		
Use Case al cual incluye: No aplica.		
Use Case al cual extiende: CU "Ingresar Nueva Gramática".		
Caso de Uso de Generalización: No aplica.		
Autor: **NF**	Fecha de creación: 07/09/09	Versión
Autor Ultima Modificación: **NF**	Fecha Ultima Modificación: 07/09/09	1.0

Nivel de Caso de Uso:	Negocio ☐	Sistema de información ■				**Orden: 5**
Nombre del Caso de Uso: Ingresar Nuevo Autómata Finito						
Objetivo: Ingresar un nuevo Autómata Finito junto con todos sus componentes.						
Actor Principal: Operador.		**Actor Secundario:** No aplica.				
Tipo de Caso de Uso :	Concreto ■	Abstracto ☐				
Prioridad :	Alta ☐	Media ■	Baja ☐			
Complejidad:	Alta ☐	Media ■	Baja ☐			
Pre-condiciones	No aplica.					
Post-condiciones	**Éxito:** Se ingreso el nuevo Autómata Finito.					
	Fracaso: No aplica.					

Curso Normal	**Curso Alternativo**
1. El Caso de Uso (CU) comienza cuando el operador selecciona la opción "Ingresar nuevo Autómata Finito"	
2. El Sistema Solicita que se le ingrese Alfabeto de Entrada.	
3. El Operado ingresa el Alfabeto de Entrada.	
4. El Sistema solicita que se ingrese el Alfabeto de Salida.	
5. El Operador ingresa el Alfabeto de Salida	
6. El Sistema solicita que se ingrese el Conjunto de Estados.	
7. El Operador ingresa el conjunto de estados.	
8. El Sistema Solicita que se seleccione el Estado Inicial del Conjunto de Estados.	
9. El Operador selecciona el Estado Inicial.	
10. El Sistema Solicita que se seleccione el conjunto de Estados Finales del Conjunto de Estados	
11. El Operador selecciona el conjunto de Estados Finales.	
12. El sistema Solicita que se ingrese la Función de Transición.	
13. Se llama al CU "Ingresar Nueva Función de Transición".	
14. El Sistema solicita que se ingrese el conjunto de Transiciones de Salida.	
15. Se llama al CU "Ingresar Nueva Función de Transición"	
16. Fin CU.	
Use Case que lo extiende: No aplica.	
Use Case incluido en: CU "Ingresar Nueva Función de Transición".	
Use Case al cual incluye: No aplica.	
Use Case al cual extiende: No aplica.	
Caso de Uso de Generalización: No aplica.	

Autor: **NF**	Fecha de creación: 07/09/09	Versión
Autor Ultima Modificación: **NF**	Fecha Ultima Modificación: 07/09/09	1.0


Lenguaje
(from Lengua...
nombre : String
palabras : Palabra
Palabra
(from Lengua...
secuenciaSimbolos : Simbolo
ongitud : Integer
Universo Alfabeto
(from Lengua...
universo : Palabra
Simbolo
(from Lengua...
denominacion
Alfabeto de Entrada
(from Maquinas)
Alfabeto de Salida
(from Maquinas)
Conjunto de Transiciones
(from Maquinas)
Función de Transición
(from Maquinas)
Conjunto de transición de Salida
(from Maquinas)
Maquina
(from Maqui...
aEntrada : Alfabeto de Entrada
aSalida : Alfabeto de Salida
estados : Conjunto de Estados
fTansición : Funcion de Transicion
fSalida : Función de Transición Automata Pila
gramatica : Gramatica Tipo 0
Funcion
(from Maqui...
estadoActual : Estado
simboloEntrada : Simbolo
simboloSalida : Simbolo
estadoTransicion : Estado
Simbolo Terminal
(from Gramaticas)
Simbolo No Terminal
(from Gramaticas)
Conjunto de Estados
(from Maqui...
estados : Estado
Estado
(from Maqui...
nombre
Alfabeto
(from Lengua...
conjuntoDeSimbolos : Simbolo
Regla de Producción
(from Gramati...
parteIzquierda : Lista
parteDeracha : Lista
tipoReglaProducción
Automata Finito
(from Maqui...
estadoInicial : Estado
estadosFinales : Conjunto de Estados Finales
aceptaciónEntrada : Boolean
determinista : Boolean
palabrasGeneradas : Palabras Generadas
equivalenciaAutomataNoDeterminista() : Automata Finito
equivalencaAutoDeterminista() : Automata Finito
compararAutomata(Automata) : Boolean
convertitAAutomata(Gramatica) : Automata Finito
convertitAgramatica(Automata) : Gramatica Tipo 0
Conjunto De Estados Finales
(from Maquinas)
Función de Transición Automata Pila
(from Maqui...
simboloDePila Actual : Simbolo
nuevoSimboloDePIla : Simbolo
Alfabeto Simbolos Terminales
(from Gramaticas)
Alfabeto Simbolos No Terminales
(from Gramaticas)
Conjunto Reglas de Produccion
(from Gramaticas)
Automata a Pila
(from Maqui...
aPila : Alfabeto De Simbolos de Pila
simboloInicialPila : Simbolo
conjuntoTransiciones : Conjunto de Transiciones
convertirAAutomataPila(Gramatica) : Automata Finito
Conjunto de Transiciones
(from Maquinas)
Axioma
(from Gramaticas)
Gramatica Tipo 0
(from Gramati...
simTerminales : Alfabeto Simbolos Terminales
simNoTerminales : Simbolos No terminales
reProducción: Reglas de Producción
axioma : Axioma
tipoGramatica : Tipo Gramatica
lenguaje : Lenguaje Generado
Lenguaje Aceptado
(from Maquinas)
Alfabeto De Simbolos de Pila
(from Maquinas)
Lenguaje
(from Lengua...
nombre : String
palabras : Palabra
Palabra Aceptada
(from Maquinas)
Maquina de Turing
alfabetoCinta : Alfabeto de Cinta
SimboloCintaBlanco : SImbolo
Alfabeto de Cinta
(from Maquinas)
Lenguaje Generado
(from Gramaticas)
Gramatica Tipo 1
(from Gramaticas)
Gramatica Tipo 2
(from Gramaticas)
Gramatica Tipo 3
(from Gramaticas)
Automata Linelamente Acotado
(from Maqui...
simboloDelimitadorDerecho : Simbolo
simboloDelimitadorIzquierdo : Simbolo
Conjunto de Transiciones
(from Maquinas)
Funcion de Transición Maquina de Turing
(from Maqui...
nuevoSimboloCinta : Simbolo
movimiento : Movimiento
Movimiento
(from Maqui...
nombre