CORE
Metadata, citation and similar papers at core.ac.uk
Provided by SEDICI - Repositorio de la UNLP

Ontología para el Aprendizaje y Compartición de Conocimientos entre Sistemas Autónomos

Ierache, Jorge
Facultad de Informática, Ciencias de la Comunicación y Técnicas Especiales. UM
jierache@unimoron.edu.ar, 54-11-56274520
Laboratorio de Sistemas Inteligentes. Facultad de Ingeniería. UBA
y

Bruno, M. Marcela
Facultad de Informática, Ciencias de la Comunicación y Técnicas Especiales. UM
mabruno@unimoron.edu.ar, 54-11-56274520
y

García Martínez, Ramón
Centro de Ingeniería de Software e Ingeniería del Conocimiento. Escuela de Postgrado. ITBA
rgm@itba.edu.ar, 54-11- 6393-4844
Laboratorio de Sistemas Inteligentes. Facultad de Ingeniería. UBA

Abstract

This paper presents a proposal for a model of an Intelligent Autonomous System (AIS) by means of an ontology based on the architecture Lopes-LC (Learning by Observation in Planning Environments, Life Cycle) by adding a cycle life learning composed of different layers that go with learning and sharing knowledge between AIS. As from the theories of the creator, theories that are generated as a product of his training and those arising from the interaction with the environment of his world of action and other AISs. It was considered from the software engineering concepts lifecycle, architectures and models. From the perspective of knowledge engineering, knowledge modelling through a preliminary ontology representing the AIS for their action learning and sharing knowledge.

Keywords: Autonomous Systems, Architecture, Ontology, Life Cycle, Sharing knowledge, Learning

Resumen

En el presente trabajo se presenta una propuesta de modelo de un Sistema Autónomo Inteligente (AIS) a través de una ontología sobre la base de la arquitectura LOPE-LC (Learning by Observation in Planning Environments, Life Cycle) con la incorporación de un ciclo de vida de aprendizaje compuesto de distintas layers que acompañan el aprendizaje y compartición de conocimiento entre AISs. Desde las teorías del creador, las teorías que se generan como producto de su entrenamiento y las que surgen de la interacción entre con el ambiente de su mundo de actuación y otros AISs. Se consideran desde la Ingeniería del Software los conceptos de ciclo de vida, arquitecturas y modelos. Desde la perspectiva de la Ingeniería del Conocimiento, la modelización de conocimiento a través de una ontología preliminar que represente al AIS para su actuación aprendizaje y compartición de conocimientos.

Palabras claves: Sistemas Autónomos, Arquitectura, Ontología, Ciclo de vida, Compartición de Conocimientos, Aprendizaje

1 INTRODUCCIÓN

El desarrollo de software para sistemas autónomos inteligentes (AISs) representado por robots que deben interactuar entre sí, con el contexto de actuación y con los humanos, imponen una mirada desde la Ingeniería de Software e Ingeniería del Conocimiento. Para el desarrollo de sistemas autónomos inteligentes se requiere de las mejores prácticas, metodologías y técnicas de estas ingenierías. En este contexto podemos enunciar diversas situaciones que justifican su aplicación en una aproximación que integre diversos puntos como lo son los *requerimientos funcionales* que vinculan principalmente sus capacidades de actuación en ambientes desconocidos. Por lo cual se requiere el desarrollo de software sobre la base de las distintas técnicas de aprendizaje automático. Estos sistemas deben ser capaces de generar teorías de cómo sus ambientes reaccionan a sus acciones, y cómo las acciones afectan al ambiente. Usualmente estas teorías de aprendizaje son parciales, incompletas e incorrectas, pero pueden ser usadas para modificar esas teorías o para crear nuevas y se enriquecen con el aseguramiento de la interoperabilidad entre AISs para compartir sus teorías. Por otro lado los sistemas autónomos requieren de una arquitectura que asocia *requerimientos no funcionales* que se corresponden con las capacidades de sensorización y actuación del sistema autónomo.

Nuestro enfoque en este trabajo apunta a aplicar de la ingeniería del software los conceptos de *ciclo de vida, arquitecturas y modelos*. Desde la perspectiva de la ingeniería del conocimiento, la *modelización de conocimiento a través de una ontología* preliminar que represente al AIS en su actuación, aprendizaje y compartición de conocimientos.

Un rasgo comúnmente asociado con la inteligencia es la capacidad de adquirir nuevos conocimientos [1], [2]. Esto se manifiesta en los procesos de aprendizaje, que aceptan ser descriptos en términos de asimilación e incorporación de información extraída del contexto. De esta forma, un sistema inteligente autónomo puede definirse [1], [3] como aquél capaz de descubrir y registrar si una acción efectuada sobre una situación dada fue beneficiosa para lograr su objetivo.

Para aprender en un mundo real, un sistema necesita formular una teoría acerca de los efectos de las acciones sobre su entorno. Necesita construir planes, monitorizar la ejecución de esos planes para detectar expectativas violadas y diagnosticar y rectificar errores que los datos inconsistentes revelen [4], [19], [20]. El desarrollo de ontologías permite compartir el entendimiento común de las estructuras de información entre personas o agentes de software, la reutilización de conocimientos del dominio, explicitar suposiciones del dominio, separar el conocimiento del dominio del conocimiento, operacional, analizar el conocimiento de un dominio. Constituyen entornos de edición de ontologías herramientas tales como: Protégé [5], Ontolingua [6], Chimaera [7], Shamsfard y Barforoush [8], [9] introducen un framework de OL (Ontologies Learning), que facilita la clasificación y comparación de OL y proponen un pequeño Kernel primario para construcción automatiza de ontologías, este Kernel contiene los conceptos, relaciones y operadores para construir ontologías independientes del dominio.

Para la actuación de los AIS se requiere de un marco que integre el aprendizaje y la compartición de conocimiento sobre la base de un modelo común para los AISs, se presenta un problema crucial cuando se está tratando con sistemas que deben "autonómicamente" adaptarse a un ambiente dinámico y desconocido. Uno de los principales objetivos de cada sistema autónomo (AIS) es cómo los operadores aprenden automáticamente (modelos de acción) que predice los efectos de acciones en el ambiente por observación de las consecuencias de estas acciones. Para poder aprender estas descripciones, éste es capaz de lograr metas auto-definidas, ejecutar los planes, encontrar los comportamientos correctos o incorrectos, y aprender de la interacción con el ambiente y otros AISs.

LOPE (Learning by Observation in Planning Enviroments) es una arquitectura de AIS que integra planificación, aprendizaje y ejecución en un reiteración cerrada, mostrando cómo funciona el comportamiento de inteligencia autónoma [11], [4], [18]. Trabajos anteriores basados en

aprendizaje de máquinas eran aplicados a resolución de problemas que principalmente estaban enfocados en aprendizaje de conocimientos cuya meta era mejorar la eficiencia de la tarea de resolución de problemas [11].
Este trabajo se concentra en el desarrollo de un modelo para el aprendizaje y compartición de teorías entre múltiples AISs sobre la base de la arquitectura LOPE-LLC (LOPE-Learning Life Cycle) [12].

2 CICLO DE VIDA DEL APRENDIZAJE DEL AIS

Cada AIS recibe la percepción desde el ambiente, llamado situaciones y aprende desde su interacción con el mundo externo (el ambiente y otros AIS). Al principio, el AIS percibe la situación inicial, y selecciona una acción al azar para ejecutar en el ambiente. Entonces, entra en un *loop* para la ejecución de una acción, percibiendo las situaciones resultantes y la utilidad de la situación, aprendiendo de la observación los efectos de aplicar las acciones en el ambiente.
Cada uno de estos AISs recibe un *input*: percepciones desde el ambiente (situaciones y utilidades); un conjunto de acciones que pueden ejecutarse, operadores aprendidos propios y de otros AISs.
La salida de cada AIS es una secuencia de acciones sobre el tiempo (para el ambiente), regularmente ese conjunto de operadores que él aprende y comparte con otros AIS. El operador O es una tupla <C,A,F,P,K,U> ,[13] donde: C es la situación inicial (condiciones), A acción a ser ejecutada, F situación final (post-condiciones), P cantidad de veces que el operador O_i fue aplicado satisfactoriamente (la citación final F fue obtenida), K cantidad de veces que la acción A fue aplicada a la condición C, U nivel de utilidad del operador, obtenido de aplicar la acción A, a la situación C, se determina sobre la relación P/K.
El sistema inteligente autónomo (AIS) "nace" con los operadores O implantados por su programador. Estos representan a los operadores que conforman el conocimiento basal que permiten el comportamiento inicial de los AIS. La formación adquirida por los operadores facilita la evolución de éstos, mediante un mecanismo de refuerzo de los exitosos y de castigo sobre aquellos operadores que tuvieron mal funcionamiento.
Uno de los principales objetivos de la arquitectura propuesta en LOPE-LLC (LOPE-Learning Life Cycle) es que el AIS aprenda autónomamente sobre la base de los operadores (modelos de acción) que predicen los efectos de las acciones en el medio ambiente mediante la observación de las consecuencias de esas acciones, conocer el comportamiento correcto o incorrecto, y aprender de los operadores.
Sobre la base del ciclo de vida para el aprendizaje y compartición de conocimientos entre AISs LOPE LC [12], el que considera los estados de evolución del AIS como nacido, novato, entrenado, maduro los que se desarrollan a través de tres Layers: [a] La layer de BI (Built-In Operators) que se desarrolla desde el nacimiento del AIS hasta que éste alcanza su estado de novato, el aprendizaje del AIS que se desarrolla en esta layer es a partir de los operadores implantados por el programador o creador. [b] La layer TB (Trained Base Operators) que se desarrolla desde el AIS novato hasta que el AIS alcanza el estado de entrenado, esta es la capa de aprendizaje en los que los operadores evolucionan en el contexto de entrenamiento del AIS sobre la base inicial de los operadores del AIS novato. Y [c] la layer WI (World Interaction Operators) que se desarrolla desde que el AIS es entrenado hasta que alcanza el estado de maduro, es la capa de aprendizaje en la que los operadores se aprenden por la interacción con el mundo y el intercambio con otros AISs. La evolución de los estados (born, newbie, trained, mature) del AIS se desarrollan a través de las Layers (BIO, BTO, WIO), se muestra el ciclo de vida en Figura 1.
Una vista conceptual de la arquitectura LOPE LC que integra el ciclo de vida propuesto se presenta en la figura 2, los AISs pueden compartir sus operadores a través de sus layers (BIO, BTO, WIO), de acuerdo a su estado (born, newbie, trained, mature). En el contexto propuesto se imponen los

siguientes axiomas de compartición entre AISs, un AIS Mature puede compartir operadores con otros AIS entrenados, novatos y nacidos, un AIS entrenado comparte sus operadores con AIS entrenados y novatos, un AIS novato comparte sus operadores con AISs novatos y nacidos.

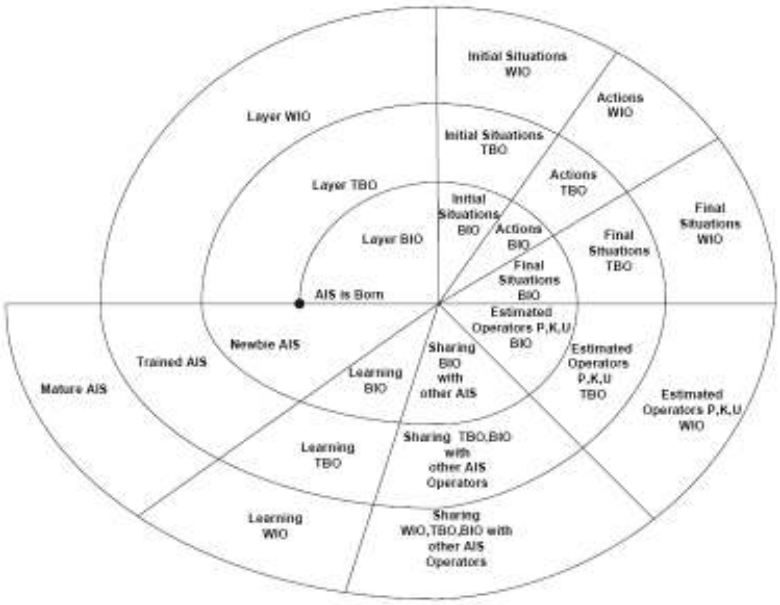

Figura 1. Modelo de Ciclo de vida del aprendizaje del AIS

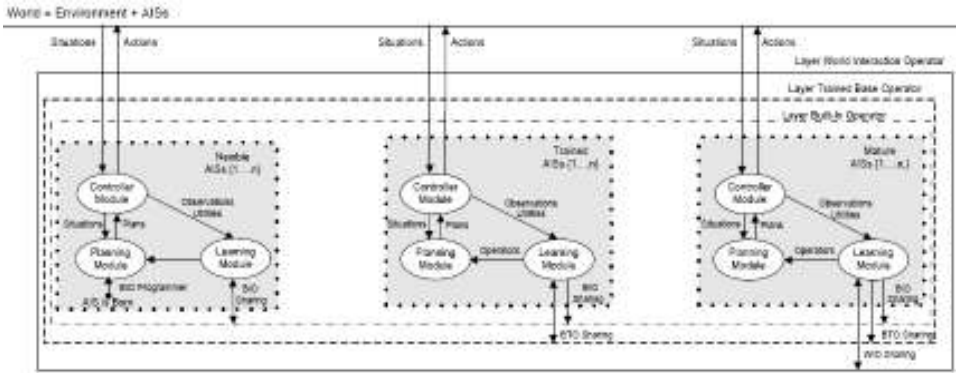

Figura. 2. Arquitectura de AIS LOPE–LLC

3 PROPUESTA DE MODELO DE AIS

Se representa en la figura 3, el modelo LOPE-LC AISs, basado sobre la arquitectura y ciclo de vida propuesto, este modelo incluye: [a] AIS, [b] World, [c] Environment especializado por Real Environment, Virtual Environment e Hybrid Environment ,[d] Knowledge Body compuesto por el grupo de operadores BIO,TBO,WIO, [e] Sensors Architecture compuesto a modo de ejemplo, por vision, light, sound, ultrasound, touch, microphone, [f] Efectors Architecture compuesto a modo de ejemplo por, por speaker, central engine, right engine, left engine. El mundo está compuesto uno o más AIS y por uno o más ambientes, un AIS opera en un ambiente el cual puede ser virtual, real, o hybrid, El AIS tiene un cuerpo de conocimiento el cual comparte con otros AIS e involucra cada una de las capas propuestas en el LLC con los siguientes estados: Born, Newbie, Trained y Mature.

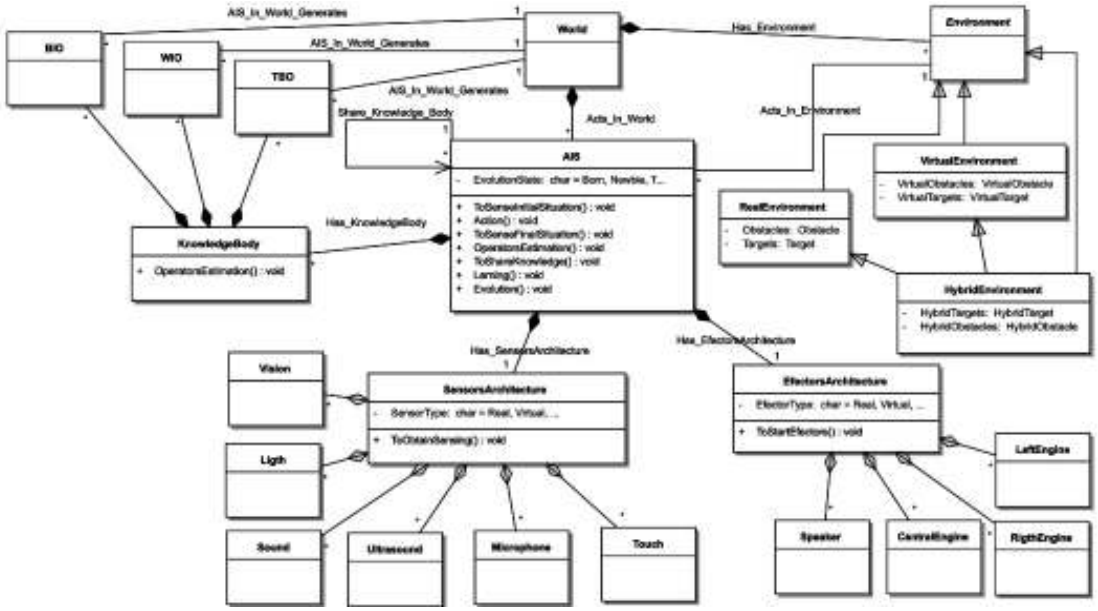

Figura 3. Modelo LOPE –LLC AISs

El AIS pasará por distintos estados de evolución (EvolutionState), según el nivel de conocimientos adquiridos a lo largo de su vida. Estos niveles van desde su nacimiento (born), pasando por un nivel básico (newbie), intermedio (trained), hasta alcanzar el nivel máximo de conocimiento (mature). Los AIS actúan en el mundo (world), y puede haber muchos de ellos dentro de ese mundo. A su vez, dentro del mundo existen diferentes ambientes (environments) que pueden ir a modo de ejemplo desde un desierto, hasta una selva. Los ambientes, se pueden ver como porciones individuales más pequeñas del mundo. Los AIS actuarán dentro de diferentes ambientes, pudiendo haber varios de ellos en cada uno, aunque solo pueden estar en un ambiente a la vez, en un momento determinado del tiempo. Sin embargo podría ocurrir que mientras un AIS, está recorriendo un ambiente, sus acciones lo lleven más allá de las fronteras de dicho ambiente, pasando a uno nuevo. De este modo, los AIS pueden ir recorriendo diversos ambientes, que están dentro de su mundo. En estos ambientes los AIS se encontrarán con diversos obstáculos (obstacles), que se interpondrán con su objetivo, y que deberá sortear. También se encontrarán con targets, que representan beneficios para el AIS, y por lo tanto debe tratar de alcanzarlos siempre que le sea posible.

Los AIS poseen, como todo AIS, una arquitectura de sensores, que le permite conocer el ambiente en el que está inmerso, y una arquitectura de efectores, con la que realiza acciones dentro del ambiente, en respuesta a lo que ocurre a su alrededor. Estas arquitecturas forman parte de la definición del AIS, y por lo tanto no pueden faltar. El AIS posee un cuerpo de conocimiento, que inicialmente será reducido, pero nunca nulo (lo que se representa el diagrama UML por medio de una composición). Los AIS pueden comportarse en forma colaborativa entre ellos compartiendo el conocimiento adquirido, contribuyendo de este modo al aprendizaje de otros AISs.

4 ONTOLOGÍA DEL AIS

Según Grüber [14], una ontología es la especificación formal de una conceptualización compartida y debe incluir un vocabulario de conceptos, relaciones y alguna especificación de su significado. En términos prácticos desarrollar una ontología incluye: definir las clases, organizar las clases en una jerarquía taxonómica (superclase/sub-clase), definir slots y describir valores permitidos para esos slots. Se pueden distinguir tres tipos fundamentales de ontologías: de un dominio, genéricas, representacionales o también denominadas meta-ontologías. En otro orden MITRE [10] presenta la integración semántica de sistemas de Comando y Control (C2), para alcanzar la interoperabilidad semántica se propone una aproximación de C2 a través de M2M (Machine to Machine) en ambientes dinámicos, en este ultimo caso es en el que radica nuestra propuesta de ontología para los AISs con la finalidad de brindar a estos la interoperabilidad Semántica que le facilite compartir conocimientos [15].
Desde el modelo propuesto en la sección anterior, para brindar un marco para la compartición de teorías entre los AISs es necesario obtener una conceptualización compartida entre los AISs, ésta se implementa con el editor de ontología Protégé [5]. Protégé permite crear y editar ontologías dentro de un entorno gráfico amigable. Por ser una herramienta específica para modelar ontologías posee un poder representativo mayor que otras herramientas, no específicas. Un diagrama UML por ejemplo, no muestra la misma información que un diagrama de una ontología. Los conceptos que ambos manejan aunque "visualmente" similares, son diferentes entre sí. Mientras que en modelado orientado a objetos, se trabaja con información, en una ontología se trabaja con conocimiento, un concepto que en sí mismo representa una evolución de la información. Permitiendo inferir nuevo conocimiento, a partir del conocimiento dado, así como posibles acciones futuras y sentido de intención. Además en un diagrama UML, no se pueden representar slots inversos, ni tampoco se pueden incluir los axiomas lógicos destinados a explicar el sentido intencionado del lenguaje, que se pueden incluir en una ontología, entre otros. Se describen en las subsecciones siguientes los principales conceptos que integran la Ontología de AIS.

4.1 World (AISs + Ambiente)

Representa el mundo completo en el que los AIS pueden existir. Este mundo tendrá a muchos AIS actuando dentro del él. A su vez el mundo se divide en porciones más pequeñas llamadas Ambientes (environment).

4.2 Environment

Son porciones individuales del mundo (ambientes). Cada ambiente tiene sus propias características que lo diferencian de los otros ambientes. Dentro de estos ambientes existirán elementos que representan obstáculos para los AIS, y otros que representan beneficios, o target que el AIS buscará alcanzar. Por otra parte, el ambiente está especializado en ambientes reales, virtuales e híbridos. Los ambientes reales tendrán obstáculos reales (como por ejemplo una pared, una piedra, etc) y targets reales (como por ejemplo una carga de batería, etc). Los ambientes virtuales tendrán obstáculos virtuales (como por ejemplo una montaña, un lago virtuales, etc) y targets virtuales (por ejemplo un sitio donde el AIS se siente "feliz"). Los ambientes híbridos son una especialización de los ambientes reales y los virtuales, y contienen características de ambos.

4.3 KnowledgeBody

Es el cuerpo de conocimiento que formará parte del AIS, y el que se irá enriqueciendo, a medida que el AIS actúa en el ambiente, y con otros AIS. Hay tres tipos de operadores que forman parte del cuerpo de conocimiento y son BIO, TBO y WIO como lo detallamos en la sección anterior

4.4 SensorsArchitecture

La arquitectura de sensores compone al AIS y es la que le permite percibir el ambiente en el que se encuentra, ejecutando el método ToObtainSensing. Existen tres tipos de arquitecturas de sensores: reales, virtuales e híbridos. A modo de ejemplo sensores de visión, de luz, de sonido, ultrasonido, tacto y un micrófono. Como se dijo anteriormente, estos sensores pueden ser reales (cuando se tiene por ejemplo un sensor de luz real), virtuales (por ejemplo un AIS simulando por medio de un sistema de cómputo externo, un sensor de ultrasonido) o híbridos (por ejemplo mediante un sensor de luz real y un sensor de temperatura simulado)

4.5 EfectorsArchitecture

La arquitectura de efectores es la que le permite al AIS realizar acciones concretas en el Ambiente en el que se encuentra, mediante el método ToStartEfectors. Del mismo modo que ocurría con la arquitectura de sensores, existen tres clases de efectores: reales, virtuales e híbridos. Según el tipo de acción que realizan pueden ser: motores (derecho, central e izquierdo), speaker.

4.6. Modelización de la Ontología del AIS

Se detalla a continuación una breve descripción de las distintas herramientas en el contexto de protege que se aplicaron para la modelización de la ontología del AIS, se destacan:

4.6.1 El Plug-in Jambalaya:

Jambalaya representa clases (conceptos) como nodos, y slots (o propiedades) como arcos. En la vista principal se pueden establecer distintos filtros para agregar/remover nodos o tipos de arcos. Se puede navegar en el diagrama generado por Jambalaya haciendo un doble clic sobre una clase, para verla en detalle, maximizándola, y luego alejarse con un clic simple en otra parte del diagrama. Existen distintas vistas, según los aspectos que se quieren mostrar, o los modos en que se quiere visualizar la ontología. En las vistas seleccionadas lo que se muestra es lo siguiente:
En la Nested View (que es la vista por default) en la figura 4, se muestran las clases anidadas con sus subclases (donde la clase más general es la clase del sistema Thing, la contiene a todas las demás).

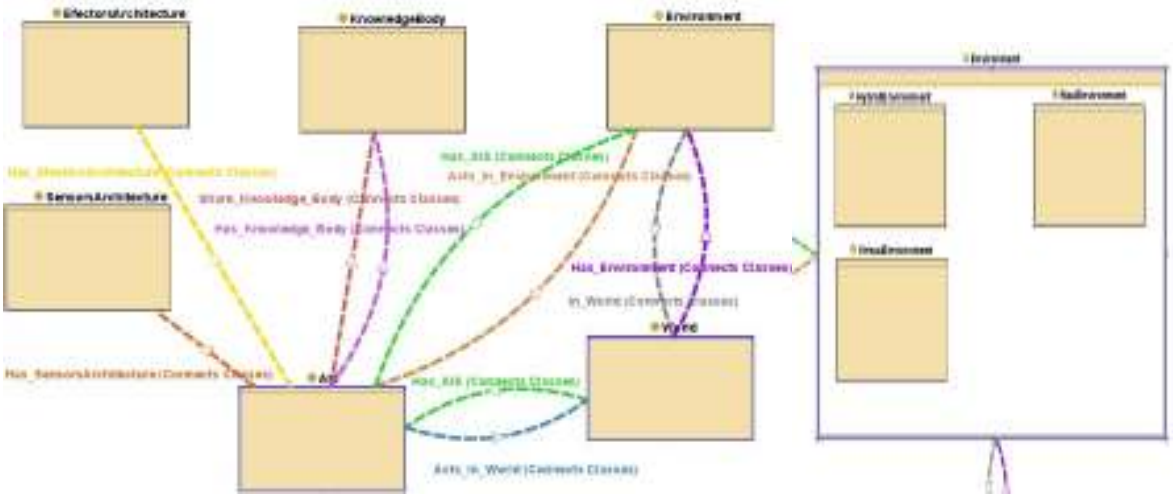

Figura 4. (Izq) vista Plugin Jambalaya. Nested view. (Der) Vista de subclases anidadas

En este caso al hacer doble click en una superclase (luego de haber seleccionado la lupa para maximizar de la barra de herramientas del plug-in), se podrán ver en su interior a sus subclases (figura 4. Der). En el caso de que la clase no tenga subclases, se verán los slots que ésta posee (figura 5). También se muestran los arcos correspondientes a las relaciones entre clases (slots inversos, etc), y no se muestran los arcos correspondientes a relaciones estructurales (del tipo has-subclass), ya que como se dijo anteriormente, la forma de mostrar estas relaciones es como clases anidadas.

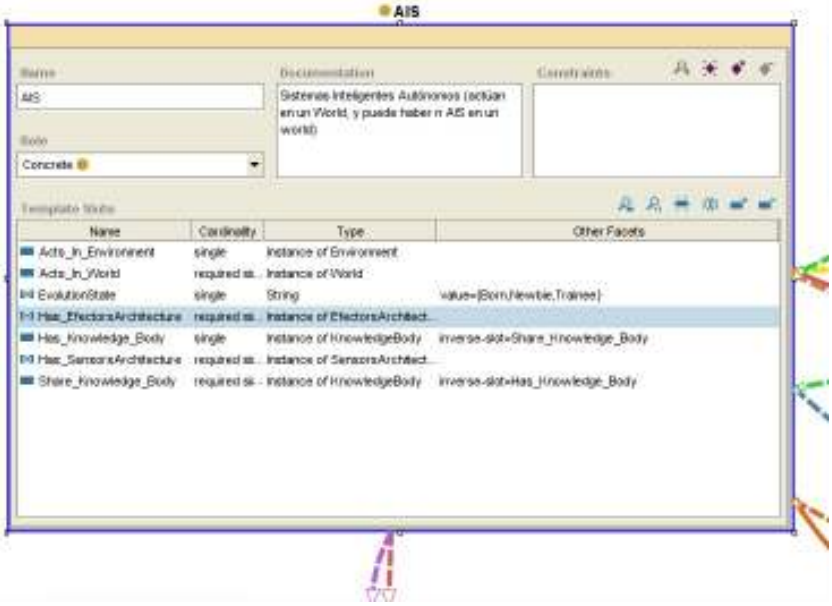

Figura 5. Nested view. Vista de los detalles de una clase (que no tiene subclases)

En la Class Tree View, se muestra la estructura jerárquica completa de la ontología. Todas las clases con sus correspondientes subclases, con la clase del sistema Thing como clase raíz. Los arcos que se muestran son los correspondientes a relaciones estructurales, del tipo has-subclass (figura 6).

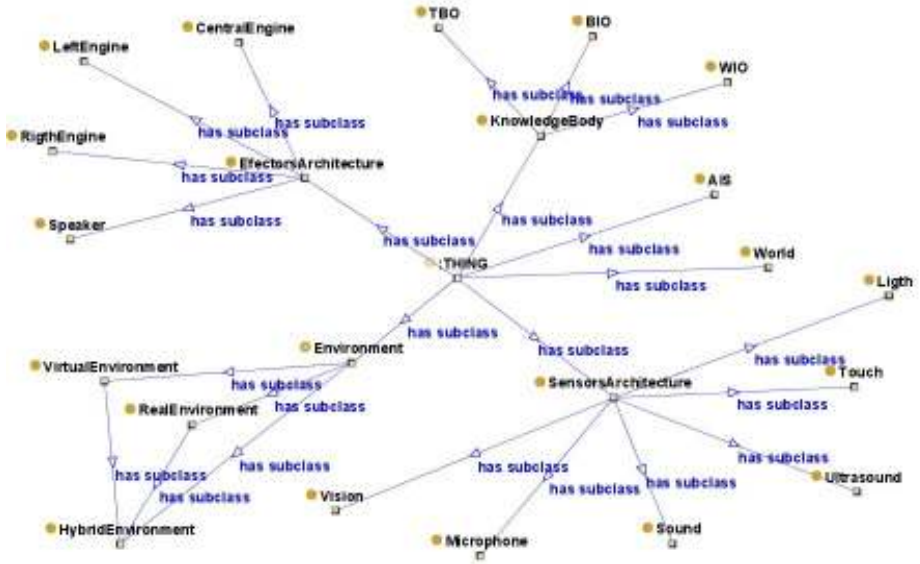

Figura 6. Vista Plugin Jambalaya. Class tree view

Además hay otras vistas que permiten ver otros aspectos de la ontología. Por ejemplo la vista Class & Instante Tree, es similar a la Vista Class Tree, pero además muestra las instancias correspondientes a las clases. Además de los arcos has-subclass, se muestran arcos correspondientes a las relaciones estructurales del tipo has-instance.

4.6.2. El plug-in TGViz

Se puede generar un gráfico de la ontología donde se ven los conceptos (solo sus nombres) y las relaciones existentes entre ellos, representadas por arcos (figura 7). Se muestran tanto relaciones estructurales, como las determinadas por los slots, slots inversos, etc. Luego de seleccionar un marco, se puede generar un gráfico que muestre los arcos que salen de y van hacia, ese marco. En este caso se seleccionó un marco clase. También es posible seleccionar marcos instancia. Se pueden establecer filtros para los arcos y clases mostrados, se pueden mostrar o no los nombres de los arcos, etc. En este caso, se muestran los conceptos relacionados con el concepto AIS.

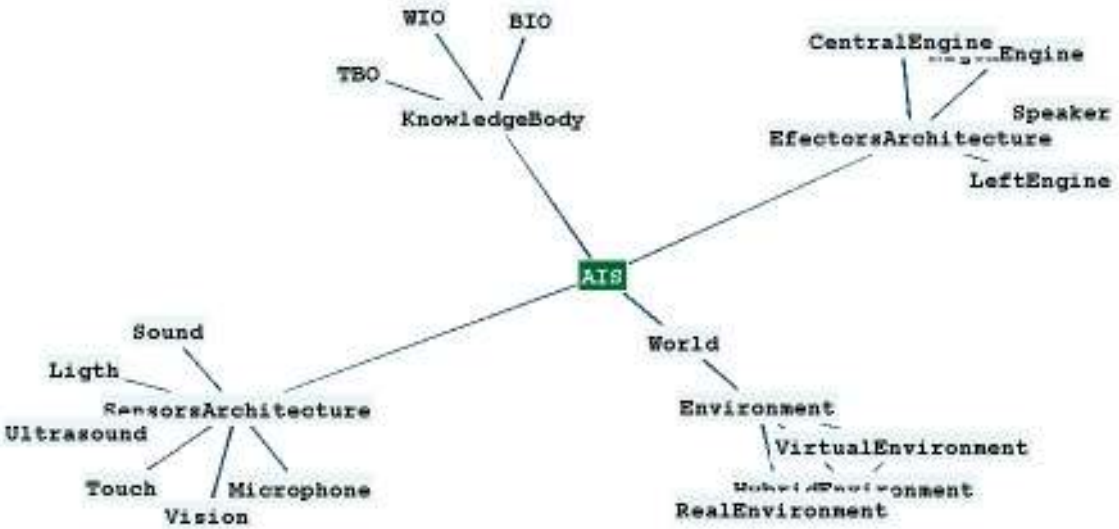

Figura 7. Vista Plugin TGViz. Selección de la clase AIS

También se pueden seleccionar otros conceptos y mostrar las relaciones de los mismos en forma individual. En este caso (figura 8), se seleccionaron, (de izquierda a derecha) y se generaron gráficos individuales para los conceptos: Environment, Efectors Architecture y SensorsArchitecture.

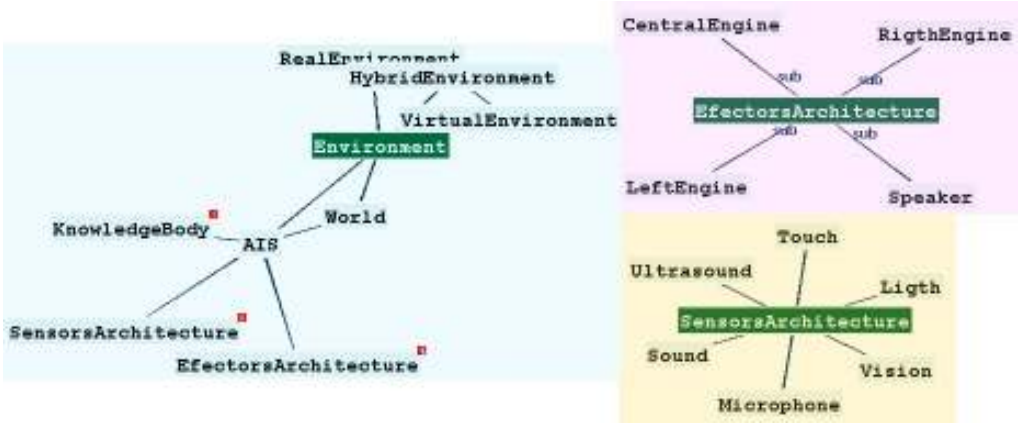

Figura 8. Vista Plugin TGViz. Selección de las clases: Environment (izq), EfectorsArchitecture (der. Arriba) y Sensors Architecture (der. Abajo)

Por default la visualización de los nombres de los arcos está desactivada, aunque se pueden ver al pasar el mouse sobre los arcos, o el concepto que relacionan (como en el caso de EfectorsArchitecture, en el gráfico anterior).También se puede establecer un "radio" para la visualización de conceptos, en cuyo caso, los conceptos que están por encima de ese radio se ocultan, y se muestra un número en un cuadrado rojo, en el concepto principal, que indica que hay n conceptos ocultos. En el gráfico anterior (figura 8 Izq) se puede ver que hay tres conceptos ocultos para KnowledgeBody, seis para SensorsArchitecture y cuatro para EfectorsArchitecture.

4.6.3. Classes Tab

Classes Tab Protégé, tiene distintos Tabs o pestañas, para facilitar el trabajo en la ontología. En el Classes tab, se pueden crear clases, editarlas, borrarlas etc. Además se puede ver la jerarquía completa conforme se la va armando, en la parte izquierda del tab (figura 9).

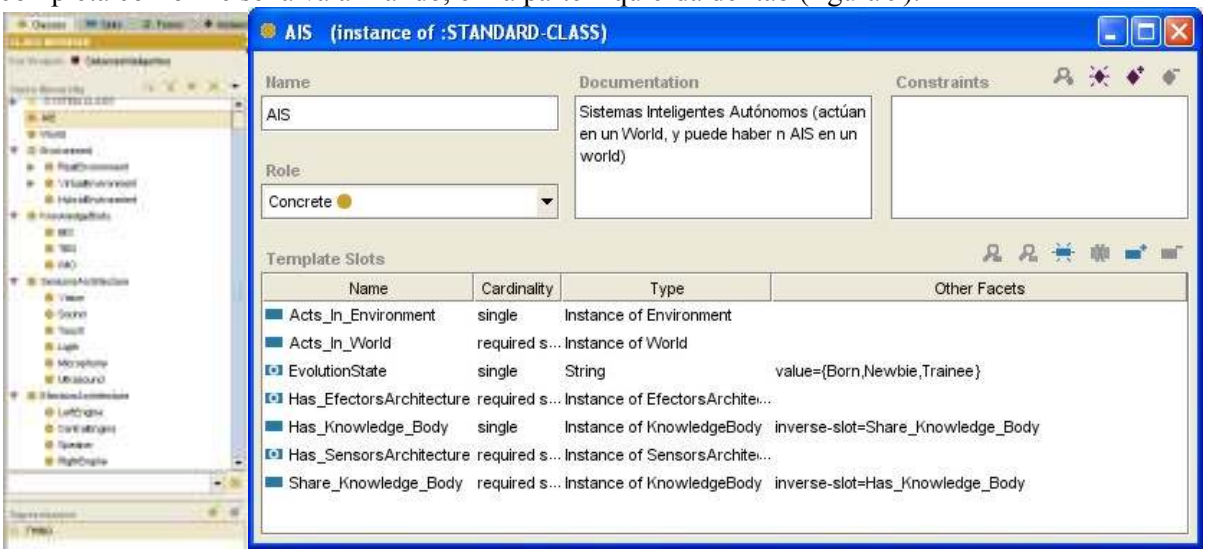

Figura 9. (Izq) Classes Tab Vista de la estructura jerárquica de la Ontología. (Der) Marco clase AIS

Cuando se selecciona una clase de la jerarquía se muestran sus slots correspondientes en la parte derecha de la ventana. En la parte inferior de la jerarquía de clases se muestran las superclases de esa clase seleccionada. También se puede hacer doble click sobre una clase, y en ese caso se desplegará una ventana conteniendo información similar a la que se presenta en el tab Classes (figura 9. Der).

5. Conclusiones y futuras líneas de investigación

El presente trabajo presento la fusión de Ingeniería de Software, Ingeniería del Conocimiento en el contexto de Sistemas Autónomos Inteligentes, aportando un modelo de Ontología sobre la base del ciclo de vida de aprendizaje y la arquitectura LOPE LC. Para llevar adelante el proyecto se consolidara un framework basado en la propuesta para el aprendizaje y compartición de conocimiento entre sistemas inteligentes autónomos distribuidos [16], sobre la base de robots de bajo costo [17]. Los AISs distribuidos compartirán conocimiento a través de una conexión a Internet, en el contexto futuro de Ontology Web [21]. En futuras líneas se plantea la validación empírica en colaboración con el Instituto de Investigaciones en Informática LIDI de la UNLP.

6. Referencias

[1] Fritz, W., García Martínez, R., Rama, A., Blanqué, J., Adobatti, R, y Sarno, M. 1989. *The Autonomous Intelligent System*. Robotics and Autonomous Systems, 5(2): 109-125.

[2] García Martínez, R. 1997. *Sistemas Autónomos. Aprendizaje Automático*. Editorial Nueva Librería. ISBN 950-9088-84-6.

[3] García Martínez, R. & Borrajo Millán, D. 1996. *Unsupervised Machine Learning Embedded in Autonomous Intelligent Systems*. Proceedings of the XIV International Conference on Applied Informatics. Páginas 71-73. Innsbruck. Austria.

[4] García Martínez, R. y Borrajo Millán, D. 2000. *An Integrated Approach of Learning, Planning and Executing*. Journal of Intelligent and Robotic Systems 29(1):47-78. Jain, S., Osberson, D., Royer, J. y Sharma, A. 1999. *Systems That Learn*. MIT Press. ISBN 0-262-10077-0.

[5] PROTÉGÉ. 2000. *The Protégé Project*. [En línea] Diponible en Web: <http://protege.stanford.edu> Pagina vigente al 24/03/07.

[6] KSL. 2006 .*Ontolingua*. http://www.ksl.stanford.edu/software/ ontolingua/. Página vigente al 24/03/07

[7] Chimaera 2000 *Ontology Enviroment*. http://www.ksl.tanford.du/software/chimaera. Página vigente al 24/03/07. Conry, S. E., Meyer, R. A., & Lesser, V. R. 1988

[8] Shamsfard, M. y Barforoush, A. 2003. *The State of the Art in Ontology Learning: A Framework for Comparison*. Knowledge Engineering Review, 18(4): 293-316.

[9] Shamsfard, M. y Barforoush, A. 2004. *Learning Ontologies from Natural Language Texts*. International Journal of Human-Computer Studies 60(1): 17-63

Sierra-Araujo, B. 2006. *Aprendizaje Automático. Conceptos Básicos y Avanzados*. Perason/Prentice Hall. ISBN 84-8322-318-X.

[10] Pulvermacher M., Stoutenburg S., Semy, S. 2004. *Netcentric Semantic Linking: An Approach for Enterprise Semantic Interoperability*, MITRE Technical Report.

[11] Borrajo, D. and Veloso, M. (1997). Lazy incremental learning of control knowledge for efficiently obtaining quality plans. *AI Review Journal*, 11, 371-405.

[12] Ierache, J., García-Martínez, R., De Giusti, A. 2008. *Learning Life Cycle in Autonomous Intelligent Systems* .En Artificial Intelligence and Practice II, Max Bramer Ed. (Boston: Springer), IFIP International Federation for Information Processing, 276: 451-455

[13] García Martínez, R. y Borrajo, D. (2000*). An Integrated Approach of Learning, Planning and Executing*. Journal of Intelligent and Robotic Systems, 29, 47-78.

[14] Gruber T. R. (1993) "*A translation approach to portable ontology specifications*". Knowledge Acquisition, 5:199–220.

[15] Ierache, J., Naiouf, M., García Martínez, R., De Giusti A. "*Un modelo de Interoperabilidad para Sistemas autónomos en entornos distribuidos*". IX Workshop de Investigadores en Ciencias de la Computación (2007) Pág. 648-652

[16] Ierache, J., Naiouf, M., García Martínez, R., De Giusti A., "*Un modelo de arquitectura para el aprendizaje y compartición de conocimiento entre sistemas inteligentes autónomos distribuidos*" VII Jornadas Iberoamericanas de Ingeniería de Software e Ingeniería del Conocimiento JIISIC 2008 Guayaquil Ecuador. Proceedings VII Ibero-American Symposium on Software Engineering Pag 179-187

[17] Ierache, J., Bruno, M., Mazza, N., Dittler, M., "*Robots y Juguetes Autónomos una Oportunidad en el Contexto de las Nuevas Tecnologías en Educación*", VII Jornadas Iberoamericanas de Ingeniería de Software e Ingeniería del Conocimiento JIISIC Febrero 2008 Guayaquil Ecuador. Proceedings VII Ibero-American Symposium on Software Engineering. Pag 371-379.

[18] García-Martínez, R., Borrajo, D., Britos, P. y Maceri, P. *Learning by Knowledge Sharing in AutonomousIntelligentSystems.* Lecture Notes in Artificial Intelligence Vol. 4140. Pág. 128-137. Springer-Verlag. 2006

[19] García-Martínez, R. and Borrajo, D. (1997). Planning, learning, and executing in autonomous systems. *Lecture Notes in Artificial Intelligence*, 1348, 208-220.

[20] García Martínez, R. & Borrajo Millán, D. 1998. *Learning in Unknown Environments by Knowledge Sharing.* Proceedings of the Seventh European Workshop on Learning Robots. Páginas 22-32. Editado University of Edinburg Press.

[21] Ontology Web, http://www.w3.org/2001/sw/WebOnt/, Pagina vigente al 09/08/08