AGENTE GENERADOR DE EJERCICIOS INTERACTIVOS PARA LA PLATAFORMA EDUCATIVA INSTITUCIONAL e-ESCEN@RI$_{UIS}$ DE LA UNIVERSIDAD INDUSTRIAL DE SANTANDER

SHEILA PAOLA UHIA KANMERER

UNIVERSIDAD INDUSTRIAL DE SANTANDER
ESCUELA DE INGENIERÍAS ELÉCTRICA, ELECTRÓNICA Y TELECOMUNICACIONES
ESPECIALIZACIÓN EN TELECOMUNICACIONES
BUCARAMANGA
2007

AGENTE GENERADOR DE EJERCICIOS INTERACTIVOS PARA LA PLATAFORMA EDUCATIVA INSTITUCIONAL e-ESCEN@RI$_{UIS}$ DE LA UNIVERSIDAD INDUSTRIAL DE SANTANDER

SHEILA PAOLA UHIA KANMERER

Monografía para optar el título de
Especialista en Telecomunicaciones

Directora
CLARA INÉS PEÑA DE CARRILLO
Doctora en Ingeniería Informática Industrial

UNIVERSIDAD INDUSTRIAL DE SANTANDER
ESCUELA DE INGENIERÍAS ELÉCTRICA, ELECTRÓNICA Y TELECOMUNICACIONES
ESPECIALIZACIÓN EN TELECOMUNICACIONES
BUCARAMANGA
2007

A Dios por su infinita misericordia.

A mis padres Agustín y Sonia quienes son mi apoyo y fortaleza.

SHEILA PAOLA

AGRADECIMIENTOS

Mis más sinceros agradecimientos a:

La DIVISIÓN DE SERVICIOS DE INFORMACIÓN de la Universidad Industrial de Santander por su colaboración.

La ESCUELA DE INGENIERÍAS ELÉCTRICA, ELECTRÓNICA Y TELECOMUNICACIONES de la Universidad Industrial de Santander, por brindarme la oportunidad de realizar la Especialización en Telecomunicaciones, lo cual es una gran contribución para mi desarrollo profesional y personal.

La Dra. CLARA INÉS PEÑA DE CARRILLO por su dedicación y valiosa colaboración.

Mis compañeros del Laboratorio de Investigación y Desarrollo del CENTIC por su paciencia y valiosos aportes para el desarrollo de este proyecto.

TABLA DE CONTENIDO

LISTA DE FIGURAS

LISTA DE TABLAS

RESUMEN EN ESPAÑOL

1. TITULO: AGENTE GENERADOR DE EJERCICIOS INTERACTIVOS PARA LA PLATAFORMA EDUCATIVA INSTITUCIONAL e-ESCEN@RI$_{UIS}$ DE LA UNIVERSIDAD INDUSTRIAL DE SANTANDER [*]

2. AUTOR: UHIA KANMERER SHEILA PAOLA. [**]

3. PALABRAS CLAVES: Agentes inteligentes, Sistemas Tutores Inteligentes, Sistemas multiagente, JADE, e-ESCEN@RI$_{UIS}$, Tecnologías de la Información y Comunicación (TICs)

4. DESCRIPCIÓN

La plataforma educativa e-ESCEN@RI$_{UIS}$ (escenario electrónico de recursos de aprendizaje e investigación UIS) de la Universidad Industrial de Santander, hace parte del *Proyecto Soporte al Proceso Educativo UIS mediante Tecnologías de Información y Comunicación – ProSPETIC*, el cual define la política de uso de las TICs en los procesos educativos institucionales y las estrategias encaminadas a lograr el desarrollo sistemático y planificado de experiencias educativas mediadas por las TICs.

El e-ESCEN@RI$_{UIS}$ es un Sistema Tutorial Inteligente implementado con tecnología Web que a través de un sistema multiagente busca modelar al estudiante con el fin de ofrecer los contenidos didácticos, las herramientas de navegación y las estrategias pedagógicas según las características del estilo de aprendizaje y del nivel de conocimiento del estudiante. La plataforma proporciona un conjunto de herramientas para permitir a los profesores crear y editar materiales, transferir, organizar y gestionar los archivos de estos

[*] Monografía
[**] Facultad de Ingenierías Físico-Mecánicas, Especialización en Telecomunicaciones, Directora: Dra. Clara Inés Peña de Carrillo

materiales, generar y gestionar diferentes tipos de ejercicios interactivos y crear y gestionar los contenidos de las unidades docentes.

Con esta monografía se pretende implantar el agente generador de ejercicios interactivos de la arquitectura de agentes del nivel superior. Este agente le permitirá al estudiante configurar un ejercicio de acuerdo a sus preferencias (el estudiante le indica la temática, el nivel de dificultad y el tipo de competencia) para autoevaluarse, o le construirá automáticamente un ejercicio adaptado a su nivel de conocimiento (el sistema es quien selecciona la temática, el nivel de conocimiento y el tipo de competencia a autoevaluar previa consulta al historial de actividades existentes en el modelo del estudiante).

RESUMEN EN INGLÉS

1. TITLE: GENERATING AGENT OF INTERACTIVE EXERCISES FOR INSTITUTIONAL EDUCATIVE PLATFORM e-ESCEN@RI$_{UIS}$ OF INDUSTRIAL UNIVERSITY OF SANTANDER*

2. AUTHOR: UHIA KANMERER SHEILA PAOLA.**

3. KEY WORDS: Intelligent agents, Intelligent Tutorial Systems, Multiagent Systems, JADE, e-ESCEN@RI$_{UIS}$, Information and Communication Technologies (ICTs)

4. DESCRIPTION

The institutional educative platform e-ESCEN@RI$_{UIS}$ (electronic scene of learning resources and investigation UIS) of the Industrial University of Santander, is part of the Project Support to Educative Process UIS by means Information and Communication Technologies - ProSPETIC, which defines the policy of use of the ICTs in the institutional educative processes and the directed strategies to obtain the systematic and planned development of half-full educative experiences by the ICTs.

e-ESCEN@RI$_{UIS}$ is an Intelligent Tutorial System implemented with technology Web that through a multiagent system it looks for to model to the student with the purpose of offering the didactic contents, the pedagogical tools of navigation and strategies according to the characteristics of the style of learning and the level of knowledge of the student. The platform provides a set of tools to allow the professors to create and to publish materials, to transfer, to organize and to manage the archives of these materials, to generate and to

* Monograph
** Physique-Mechanics Engineering Department, Specialization in telecommunications, Dra. Clara Inés Peña de Carrillo.

manage different types from interactive exercises and to create and to manage the contents of the educational units.

With this monograph is tried to implant the generating agent of interactive exercises of the architecture of agents of the superior level. This agent will allow the student to form an exercise according to his preferences (the student indicates the thematic, the level of difficulty and the type of competition) to auto evaluate itself, or he will construct an adapted exercise automatically to him to its level of knowledge (the system is who selects the thematic, the level of knowledge and the type of competition to auto evaluate previous consultation to the file of existing activities in the model of the student).

INTRODUCCIÓN

El proyecto ProSPETIC (*Proyecto Soporte al Proceso Educativo UIS mediante Tecnologías de Información y Comunicación*), le ha permitido a la UIS definir políticas y estrategias que soporten los programas académicos a través del uso de las Tecnologías de Información y Comunicación. En ese sentido se orientan entonces los desarrollos tecnológicos para la construcción del escenario electrónico de recursos de aprendizaje e investigación UIS (e-ESCEN@RI_{UIS}); una plataforma educativa, con tecnología Web, para que el estudiante acceda a los recursos didácticos existentes en la red, de forma dinámica, adaptativa, asistida y personalizada, y pueda lograr un verdadero aprendizaje significativo.

A través de esta investigación se contribuye al desarrollo del e-ESCEN@RI_{UIS} con la implementación del agente generador de ejercicios interactivos; uno de los agentes personales digitales propuestos en la arquitectura multiagentes. Este agente le permite al estudiante configurar un ejercicio de acuerdo a sus preferencias o solicitar al sistema la construcción de un ejercicio adaptado a su nivel de conocimiento.

El documento está organizado en cinco capítulos con el siguiente contenido:

En el primer capítulo se presentan los aspectos generales de la investigación como: el planteamiento del problema, la justificación, los objetivos y el estado del arte.

En el segundo capítulo se realiza una revisión de los conceptos básicos de la tecnología de agentes inteligentes y su aplicación en sistemas Web.

En el tercer capítulo se introducen conceptos básicos relacionados con los sistemas de tutoría inteligente soportados en el hipermedia.

En el cuarto capítulo se consideran aspectos de la tecnología a utilizar para la implementación del agente generador de ejercicios interactivos (lenguajes de comunicación entre agentes, plataformas y metodologías para el desarrollo de sistemas multiagentes, etc).

El quinto capítulo presenta la arquitectura multiagentes del e-ESCEN@RI_{UIS} y el desarrollo del agente generador de ejercicios interactivos, objetivo de está investigación.

Finalmente, se cierra el documento con las conclusiones y recomendaciones de la investigación realizada.

1. ASPECTOS GENERALES

1.1. PLANTEAMIENTO DEL PROBLEMA

La Universidad Industrial de Santander (UIS) en su propósito de formar personas integrales se enfrenta ante los nuevos retos tecnológicos para transformar los sistemas de enseñanza tradicional mediante el uso de las Tecnologías de Información y Comunicación y nuevas estrategias pedagógicas que fomenten el aprendizaje activo. En ese sentido se orientan los esfuerzos para la construcción del escenario electrónico de recursos de aprendizaje e investigación UIS (e-ESCEN@RI_{UIS}) considerado como una plataforma educativa abierta e interoperable que permita el acceso a los contenidos didácticos, de forma dinámica, adaptativa, asistida y personalizada tomando como base el estilo de aprendizaje del estudiante y su nivel de conocimiento.

La arquitectura que soporta el e-ESCEN@RI_{UIS} está constituida por un sistema multiagente de dos niveles (ver [30]): los agentes del nivel superior que actúan como asistentes personales digitales para aprender del entorno y ayudar a desarrollar las actividades de aprendizaje propuestas; y los agentes del nivel inferior que actúan como intermediarios entre los agentes del nivel superior y las bases de datos que constituyen el modelo del dominio, el modelo pedagógico y el modelo del estudiante para recomendar los materiales didácticos adaptados a las preferencias del estudiante, a su estilo de aprendizaje y a su nivel de conocimiento.

Con esta investigación se pretende implantar el agente generador de ejercicios interactivos de la arquitectura de agentes del nivel superior tomando como base el planteamiento de [30]. Este agente le permitirá al estudiante configurar un ejercicio de acuerdo a sus preferencias (el estudiante le indica la temática, el nivel de dificultad y el tipo de competencia) para autoevaluarse, o le construirá automáticamente un ejercicio adaptado a su nivel de conocimiento (el sistema es quien selecciona la temática, el nivel

de conocimiento y el tipo de competencia a autoevaluar previa consulta al historial de actividades existentes en el modelo del estudiante). La arquitectura y el flujo de comunicación entre unos agentes y otros se puede observar en la figura 1.

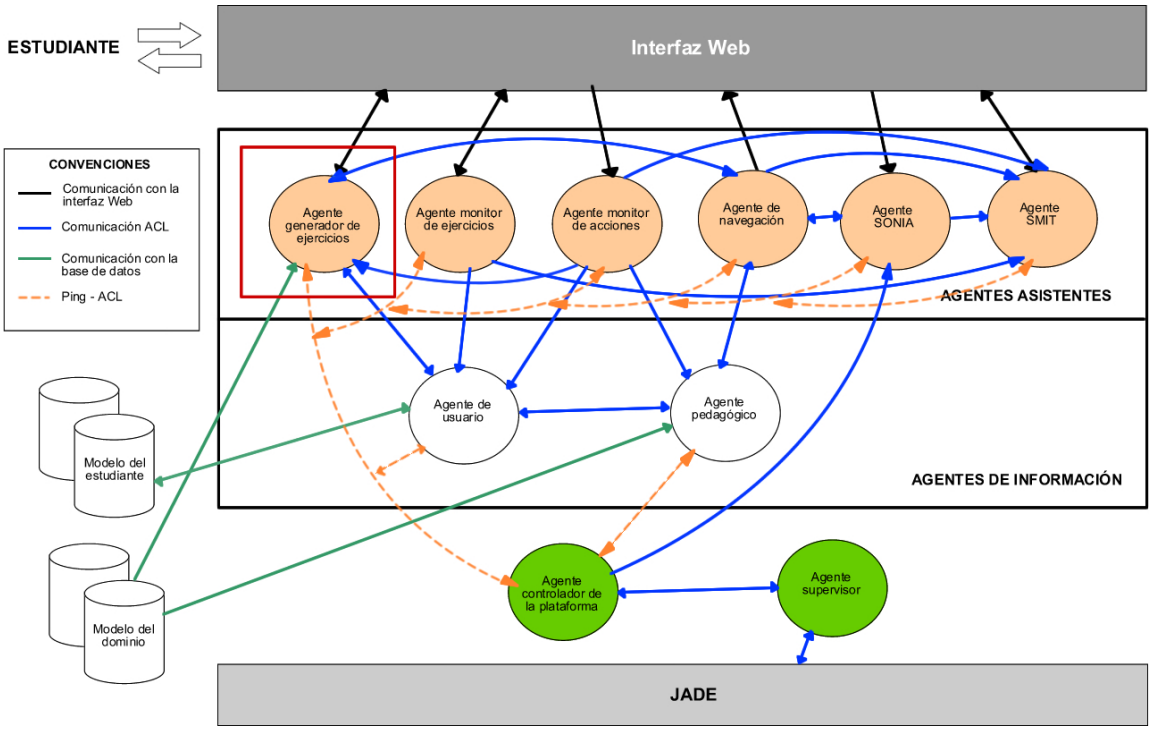


Figura 1. Arquitectura multiagente del e-ESCEN@RI$_{UIS}$ (tomado de [30])

1.2. JUSTIFICACIÓN

En la actualidad, como consecuencia de la globalización, se han venido dando grandes avances en el campo de las Tecnologías de la Información y Comunicación, lo que ha permitido que el área de la educación se proyecte y expanda a través de Internet, ya que es un medio eficaz para garantizar la comunicación, interacción y aprendizaje entre los diferentes actores que hacen parte del proceso de enseñanza/aprendizaje.

La incorporación de las Tecnologías de la Información y Comunicación en el ámbito académico ha traído consigo no sólo dar el soporte a las actividades curriculares y de

investigación, sino que ha propiciado el intercambio de información entre alumnos y docentes de una manera dinámica a través de la red, lo que ha dado origen al establecimiento de nuevos ambientes de aprendizaje basados en el uso de Internet como medio difusor de conocimientos.

El surgimiento de estos ambientes de aprendizaje se remonta a los años 50, donde aparecieron los primeros sistemas de enseñanza asistidos por computador, y que fueran los llamados *programas lineales*, porque mostraban el conocimiento de forma secuencial y no tenían en cuenta la aptitud del alumno. Luego vinieron los *sistemas ramificados*, los cuales se diferenciaban de los anteriores por su capacidad de actuar según la respuesta del alumno [34].

Entre finales de los años 60 y principios de los años 70 surgieron los *sistemas generativos* (o sistemas adaptativos), los cuales adaptaban la enseñanza de acuerdo a las necesidades del alumno. Estos sistemas eran capaces de generar ejercicios acorde con el nivel de conocimiento del alumno.

Posteriormente, estos sistemas evolucionaron y con la incorporación de la Inteligencia Artificial aparecieron los Sistemas Tutoriales Inteligentes (STI) que permiten emular el proceso de aprendizaje/enseñanza humano, adaptando el tipo y el contenido de la enseñanza a las necesidades específicas del alumno, decidiendo cuándo introducir nuevos conceptos o repasar los anteriores si éstos no han sido asimilados. Estos sistemas tienen en cuenta los conocimientos a enseñar (modelo del dominio), la forma de enseñarlo (modelo pedagógico), así como la información relevante del alumno (modelo del estudiante) para lograr una enseñanza personalizada.

Los Sistemas Tutoriales Inteligentes basados en Inteligencia Artificial permiten representar el conocimiento y dirigir una estrategia de enseñanza. Son capaces de comportarse como expertos, tanto en el manejo del dominio de conocimiento que se enseña (mostrando al alumno cómo avanzar en el conocimiento), como en el dominio pedagógico (diagnosticando la situación en la que se encuentra el estudiante y de acuerdo a ello ofrecer una acción o solución que le permita progresar en el aprendizaje) [20].

La plataforma educativa institucional e-ESCEN@RI$_{UIS}$ es un STI que a través de un sistema multiagente (SMA) busca modelar al estudiante con el fin de ofrecer contenidos didácticos, herramientas de navegación y estrategias pedagógicas adaptadas al estilo de aprendizaje y al nivel de conocimiento del estudiante.

El desarrollo del agente generador de ejercicios interactivos permitirá la construcción de ejercicios adaptativos para el estudiante (considerando su estado de conocimiento o sus preferencias). El agente puede escoger un nivel de dificultad conveniente para las preguntas de acuerdo al progreso del estudiante en el sistema. Este proceso se lleva a cabo teniendo en cuenta los siguientes aspectos:

- Las preferencias del usuario, en cuyo caso es el estudiante quien configura los temas y los tipos de preguntas que desea contestar (ejercicio configurado).
- El nivel de conocimiento del estudiante, en cuyo caso es el agente quien selecciona los temas y los tipos de preguntas que el estudiante puede responder en un momento dado (ejercicio adaptado al nivel de conocimiento del estudiante).

1.3. OBJETIVOS

1.3.1. Objetivo general

Desarrollar el agente generador de ejercicios interactivos para la plataforma educativa institucional e-ESCEN@RI$_{UIS}$.

1.3.2. Objetivos específicos

- Análisis de los conceptos básicos de la tecnología de agentes inteligentes y su aplicación en sistemas Web.
- Análisis de la estructura general de la arquitectura multiagente del e-ESCEN@RI$_{UIS}$ para identificar funcionalidades del agente a desarrollar.

- Estudio de la herramienta JADE (Java Agent DEvelopment Framework) para la implantación del agente.
- Definición de la ontología para el diseño del funcionamiento del agente generador de ejercicios interactivos.
- Implantación del agente en la plataforma e-ESCEN@RI$_{UIS}$.

1.4. ESTADO DEL ARTE

En la nueva sociedad del conocimiento, la incorporación de las nuevas tecnologías de información y comunicación, tales como el computador y la Internet, han generado nuevos enfoques educativos centrados en el aprendizaje del alumno, a través de entornos de aprendizaje en línea. Estos implican nuevas formas de enseñar y nuevas formas de aprender.

Es por ello, que se están desarrollando diferentes productos tecnológicos que incluyen agentes inteligentes, los cuales van desde poder entregar información más relacionada con los estudiantes, manejo de datos del usuario ya sea para reconocerlo y/o para tomar decisiones futuras, apoyarlo en el desarrollo de la sesión de trabajo e incluso ayudarlo como cuando un compañero enseña a otro, etc.

De esta manera, la tecnología está influenciando al menos en dos aspectos al mundo educacional. Uno relacionado con los intereses pedagógicos y la segunda con los cambios en las habilidades y competencias requeridas, para lograr transmitir el conocimiento.

El software educativo, juega un papel importante como apoyo al docente y al estudiante en la formación de este último. Si bien existen importantes y numerosos desarrollos de sistemas, los resultados no han sido lo esperado. En su mayoría corresponden a sistemas de práctica y ejercitación y su principal característica es entregar al alumno la posibilidad

de ejercitarse en una determinada tarea una vez obtenidos los conocimientos necesarios para el dominio de la misma.[1]

Los sistemas de ejercitación y práctica, no tienen una característica formativa. Sin embargo, existe una línea de sistemas desarrollados para apoyar la formación de los alumnos denominados *tutores inteligentes*. Por diferentes motivos, estos no han logrado los resultados esperados, dándole paso a sistemas que se basan en agentes inteligentes, los cuales son capaces de: comunicarse con el usuario en un lenguaje natural; simular el comportamiento humano; adaptarse a las necesidades de los alumnos; entre otros, lo que le ha permitido a estos sistemas apoyar la formación de los estudiantes, con la utilización de nuevos modelos y herramientas.[2]

[1] Gros, B. (1997) Diseños y programas educativos Pautas pedagógicas para la elaboración de software (enero 1997). Barcelona: Ariel Educación.
[2] VILLARREAL, G. Agentes Inteligentes en Educación. Centro Comenius Universidad de Santiago de Chile

2. TECNOLOGÍA DE AGENTES INTELIGENTES

A continuación se presenta el concepto de agente desde el punto de vista general, luego se define inteligencia para así llegar a la definición de Agente Inteligente. También se indican las características, tipologías, aplicaciones, estructura de los agentes inteligentes, ontología, definición de sistemas multiagente y su aplicación en sistemas Web para la enseñanza en línea.

2.1. DEFINICIÓN DE AGENTE

Según [31], un agente es cualquier entidad que percibe su entorno a través de sensores y actúa sobre ese entorno mediante efectores. Un agente es racional cuando realiza la mejor acción posible a partir de los datos percibidos. Ver figura 2.

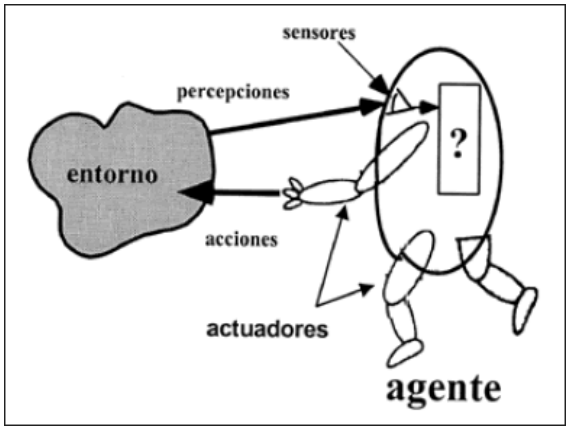


Figura 2. Visión esquemática de un agente (tomado de [36])

Según la enciclopedia Wikipedia, un agente es por lo general visto como una entidad *inteligente*, equivalente en términos computacionales a un proceso del sistema operativo, que existen dentro de cierto contexto o ambiente, y que se pueden comunicar a través de un mecanismo de comunicación inter-proceso, usualmente un sistema de red, utilizando protocolos de comunicación.

Los seres humanos, los robots y el software pueden ser considerados como agentes. A continuación se define cada uno de estos agentes:

- **Agentes humanos***:* estos poseen ojos, oídos y otros órganos que les sirven de sensores, así como: piernas, manos, boca y otras partes del cuerpo que le sirven de efectores, los cuales utiliza para alcanzar algún objetivo en particular en un ambiente dado.
- **Agentes Robóticos***:* los sensores son sustituidos por cámaras de vídeo y los efectores son reemplazados mediante servomecanismos[3].
- **Agentes de software:** son programas de computación que se ejecutan en un ambiente, y que son capaces de realizar acciones dentro de este, con el fin de alcanzar objetivos particulares para los cuales fue diseñado. Para un agente de software, sus percepciones y acciones vienen dadas por instrucciones de programas en algún lenguaje en particular.

2.2. DEFINICIÓN DE INTELIGENCIA

Según [1] es el grado de razonamiento y capacidad de aprendizaje. La habilidad del agente para aceptar las instrucciones de las metas del usuario y llevar a cabo la tarea delegada por él. Al menos deben existir algunas instrucciones de preferencias tal vez en forma de reglas con un motor de inferencia o algún otro mecanismo que actúe sobre estas preferencias. Altos niveles de inteligencia incluyen un modelo de usuario o alguna otra forma de entender y razonar acerca de lo que el usuario quiere hacer.

[3] Sistema electromecánico que se regula por sí mismo al detectar el error o la diferencia entre su propia actuación real y la deseada.

2.3. DEFINICIÓN DE AGENTE INTELIGENTE

Algunas definiciones dadas por diferentes autores a los agentes son las siguientes:

El agente MuBot: "El término agente se usa para representar dos conceptos ortogonales. El primero es la habilidad de un agente para la ejecución autónoma. El segundo es la habilidad que tiene un agente para hacer razonamientos orientados al dominio que se esté tratando."[4]

El Agente Maes [26]. "Los agentes autónomos son sistemas computacionales que habitan en entornos dinámicos complejos, percibiendo y actuando autónomamente en ese entorno, y realizan un conjunto de metas o tareas para las que han sido diseñados."

El Agente KidSim [33]. "Un agente es como una entidad software persistente dedicada a un propósito específico. La 'persistencia' distingue a los agentes de las subrutinas; los agentes tienen sus propias ideas sobre como ejecutar tareas, sobre sus agendas. Con 'propósito específico' se distinguen los agentes de las aplicaciones multifunción, que son típicamente más pequeños."

El Agente Hayes-Roth [15]. "Los agentes inteligentes realizan continuamente tres funciones: percibir condiciones dinámicas en el entorno, actuar afectando a las condiciones del entorno, y razonar para interpretar lo percibido; resuelven problemas, muestran interfaces y determinan acciones"

El Agente IBM. "Los agentes inteligentes son entidades software que llevan a cabo un conjunto de operaciones en beneficio de un usuario u otro programa con algún grado de independencia o autonomía, y haciendo esto, emplean algún conocimiento o representación de las metas y deseos del usuario"[5]

[4] http://ww.crystaliz.com/logicware/mubot.html

[5] http://activist.gpl.ibm.com:81/WhitePaper/ptc2.htm

El agente Brenner [5]. “Un agente software inteligente es un programa que puede realizar tareas específicas para un usuario y posee un grado de inteligencia suficiente para ejecutar parte de sus tareas de forma autónoma y para interactuar con su entorno de forma útil”.

El agente Wooldridge [35]. “Un agente inteligente es un sistema (hardware o software) situado en un determinado entorno, capaz de actuar de forma autónoma y flexible en dicho entorno para llevar a cabo unos objetivos predeterminados”.

Luego de definir agente inteligente, se puede hablar sobre *Sistemas basados en agentes,* los cuales utilizan varios agentes que colaboran juntos e interactúan para resolver un problema, a esto se le llama Sistema multiagente.

2.4. CARACTERÍSTICAS DE LOS AGENTES

Debido a la variedad de definición de agente, se ha optado por enumerar un conjunto de propiedades que lo caracterizan. Es importante aclarar que el agente no necesariamente debe poseer todas las características.

Un agente está caracterizado por una serie de calificativos que denotan algunas propiedades que deben ser cumplidas por el mismo. Según la definición dada anteriormente por [35], se debe entender que el agente es flexible si es:

- **Reactivo**, el agente es capaz de responder a cambios en el entorno en que se encuentra situado.
- **Pro-activo**, a su vez el agente debe ser capaz de intentar cumplir sus propios planes u objetivos.
- **Social**, debe ser capaz de comunicarse con otros agentes mediante algún tipo de lenguaje de comunicación de agentes.

Las características según [12] y [27], que califican a un agente son las siguientes:

- **Continuidad Temporal**: se considera a un agente como un proceso sin fin, ejecutándose continuamente y desarrollando su función.
- **Autonomía**: un agente es completamente autónomo si es capaz de actuar basándose en su experiencia. El agente puede adaptarse aunque el entorno cambie severamente.
- **Sociabilidad**: este atributo permite a un agente comunicarse con otros agentes o incluso con otras entidades.
- **Racionalidad**: el agente siempre realiza *lo correcto* a partir de los datos que percibe del entorno.
- **Reactividad**: un agente actúa como resultado de cambios en su entorno. En este caso, un agente percibe el entorno y esos cambios dirigen el comportamiento del agente.
- **Pro-actividad**: un agente es pro-activo cuando es capaz de controlar sus propios objetivos a pesar de cambios en el entorno.
- **Adaptatividad**: está relacionado con el aprendizaje que un agente es capaz de realizar y si puede cambiar su comportamiento basándose en ese aprendizaje.
- **Movilidad**: capacidad de un agente de trasladarse a través de una red telemática.
- **Veracidad**: asunción de que un agente no comunica información falsa a propósito.
- **Benevolencia**: asunción de que un agente está dispuesto a ayudar a otros agentes si esto no entra en conflicto con sus propios objetivos.

2.5. ESTRUCTURA DE LOS AGENTES INTELIGENTES

Si se toma como base la definición de agentes dada por [31] y esquematizada en la (Figura 2), en el diseño de un programa de agentes hay que tener en cuenta el entorno en el que se moverá el agente (es decir, la estructura de su ambiente computacional) que le permitirá definir la forma de recibir las percepciones, realizar los diferentes análisis con la información recibida y producir las determinadas acciones para solucionar un problema específico. Esta estructura computacional recibe el nombre de *arquitectura* que [23] la representa según la (Figura 3) haciendo énfasis en las relaciones del agente con las componentes del sistema.

La relación entre agentes, arquitectura y programas se puede resumir con la siguiente fórmula:

Agente = **Programa agente** + **Arquitectura**

Donde, el *programa agente* será una función que implementará la transformación de secuencias de percepciones en acciones y la *arquitectura* será un computador que se ocupará de que las percepciones lleguen al programa y las acciones lleguen a los efectores.

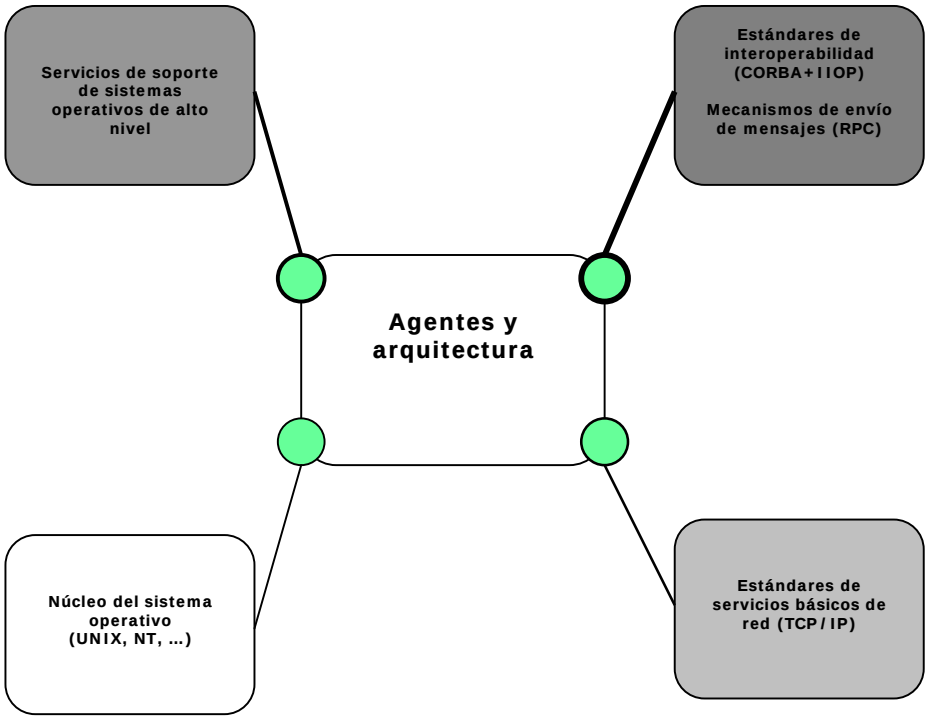


Figura 3. Relación de un agente con las componentes del sistema (tomado de [29])

Antes de realizar el diseño de un programa de agentes es necesario tener una idea clara y precisa de las posibles percepciones y acciones que intervendrán, las metas o medidas de desempeño que se deben cumplir y el tipo de entorno en el que operará el agente. Este conjunto de elementos según [31] se conoce con las siglas PAMA referentes a

Percepciones, Acciones, Metas y Ambiente. Ejemplos de tipos de agentes y elementos que los caracterizan se pueden apreciar en la tabla 1.

Tabla 1. Ejemplos de tipos de agente y elementos que los caracterizan

Tipo de agente	Percepciones	Acciones	Metas	Ambiente
Sistema de diagnóstico médico	Síntomas, hallazgos, respuestas del paciente	Preguntas, pruebas, tratamientos	Paciente saludable, costos mínimos	Paciente, hospital
Sistema de análisis de imagen satelital	Píxeles de intensidad y colores diversos	Imprimir una clasificación de escena	Clasificación correcta	Imágenes enviadas desde un satélite de orbita
Robot clasificador de partes	Píxeles de intensidad variable	Tomar las partes y ordenarlas en compartimientos	Poner las partes en los compartimientos que le correspondan	Banda transportadora sobre la que se encuentran las partes
Regulador de la refinería	Lecturas de temperatura y presión	Abrir y cerrar válvulas; ajustar la temperatura	Maximizar la pureza, producción y seguridad	Refinería
Tutor interactivo de Inglés	Palabras mecanografiadas	Impresión de ejercicios, sugerencias, correcciones	Maximizar el resultado del estudiante en las pruebas	Grupo de estudiantes

2.6. TIPOS DE AGENTES

Los agentes inteligentes se pueden clasificar en los siguientes dos grupos: basados en el mapeo de las percepciones a las acciones y basados en el tipo de aplicación.

2.6.1. Basados en el mapeo de las percepciones a las acciones

- **Agentes de reflejo simple:** Los agentes de reflejo simple operan siguiendo las instrucciones que le proporcionan un conjunto de reglas del tipo condición-acción, las cuales le permiten al agente establecer las conexiones entre las percepciones y las acciones. Mediante un chequeo entre el estado actual del ambiente, el cual es

captado por los sensores, el agente busca en el conjunto de reglas condición-acción, cuál es la regla que coincide con la percepción captada, para que se ejecute la acción. Este agente funcionará correctamente solo si se toma la decisión adecuada basándose en la percepción del ambiente en un instante dado. Este tipo de agente no contiene internamente estados. Ver figura 4.

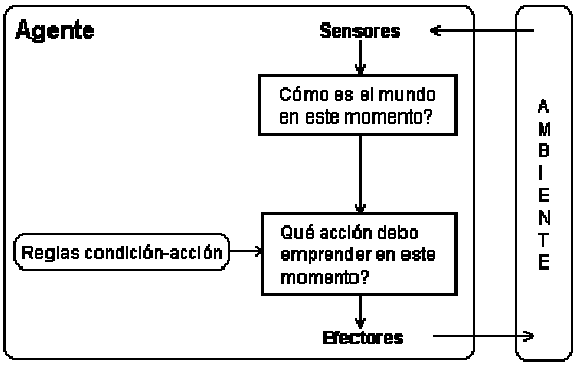


Figura 4. Esquema de un agente reflejo simple (tomada de [37])

- **Agentes informados de lo que pasa:** El agente tiene la capacidad de decidir la acción a ejecutar considerando y evaluando acciones y/o percepciones anteriores en función de una visión global del ambiente. Específicamente en estos casos el agente necesita actualizar la información del estado interno que le permita discernir entre estados del mundo que generan la misma entrada de percepciones pero que, sin embargo, para cada uno de los estados se necesitan acciones distintas. Para dotar al agente de capacidad de discernir se debe codificar conocimiento de dos clases: primero el conocimiento de cómo evoluciona el ambiente y segundo el conocimiento de cómo las acciones del agente afectan el ambiente. Ver figura 5.

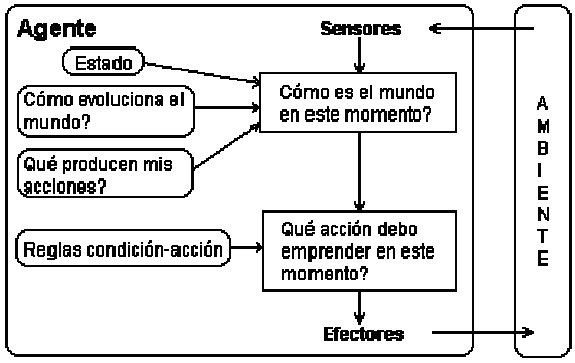


Figura 5. Esquema de un agente informado de lo que pasa (tomada de [37])

- **Agentes basados en metas:** Para que el agente pueda discernir no es suficiente tener información acerca del estado que prevalece en el ambiente, es necesario también disponer de datos acerca de las metas que se pretendan alcanzar. Al tener esta información, se pueden combinar con el resultado que producirán las posibles acciones que se emprendan y así elegir aquellas que permitan alcanzar las metas de una manera eficiente y eficaz. Es posible que el agente deba ejecutar operaciones de búsqueda y planificación de las metas para seleccionar la acción a seguir.

  Es importante diferenciar el comportamiento basado en reglas del tipo condición-acción con el de este tipo de agente, ya que las decisiones se deben tomar en función de satisfacer las metas trazadas. Ver figura 6.

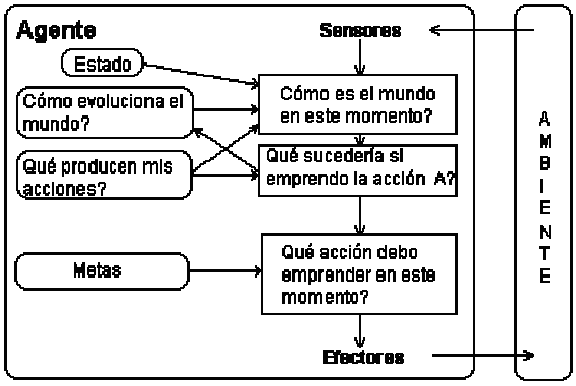


Figura 6. Esquema de un agente basado en metas (tomada de [37])

- **Agentes basados en utilidad:** Las metas por sí solas no garantizan la obtención de una conducta de alta calidad, es por ello que el agente basado en la utilidad usa un criterio para estimar el grado de satisfacción de un estado con el objeto que le sirva para escoger entre distintas acciones válidas. Ver figura 7

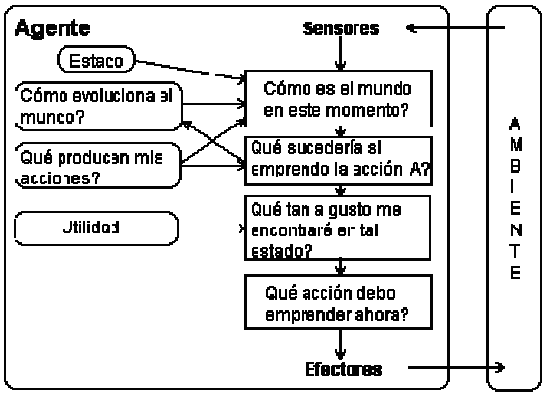


Figura 7. Esquema de un agente basado en utilidad (tomada de [37])

2.6.2. Basados en el tipo de aplicación

- **Agentes para aplicaciones industriales**
  - ***Control de procesos:*** Esta es una aplicación natural para los agentes inteligentes y sistemas multiagente, ya que son en sí mismos sistemas autónomos reactivos. ARCHON (*Architecture for Cooperative Heterogeneous Online Systems*) es una plataforma de software para la construcción de sistemas multiagente, junto con una metodología asociada para la construcción de aplicaciones utilizando la plataforma, según [19]. Se aplica a casos de dirección, control del transporte de electricidad y en el control de un acelerador de partículas.
  - ***Manufacturación:*** [28] YAMS (*Yet Another Manufacturing Systems*), es un sistema que se utiliza en el área de la manufactura, tiene como objetivo principal administrar eficientemente los procesos de producción de diversas fábricas. Utiliza un esquema multiagente donde cada fábrica y sus componentes son representados por agentes que interactúan entre sí.

- ***Control de tráfico aéreo:*** [24] describe un sofisticado agente para la realización de un sistema de control aéreo, conocido como OASIS (*Optimal Aircraft Sequencing using Intelligent Scheduling*). Se representan diversos aeropuertos y el tráfico entre ellos. Es un simulador, que recrea las actividades entre aeropuertos reales.

- **Agentes para aplicaciones comerciales**
  - ***Administración de Información:*** Los agentes son importantes para reducir la sobrecarga de información. Estos agentes necesitan comprender los diferentes formatos de los datos para registrar las bases de datos en busca de información. Los agentes avanzados de administración de información filtran, condensan y presentan la información siguiendo una serie de reglas que les especifique el usuario.
  - ***Comercio Electrónico:*** Los agentes dedicados al comercio electrónico pueden ayudar a los compradores a localizar vendedores y viceversa. Los agentes especializados en compras pueden recorrer la red buscando productos que reúnan una serie de características impuestas por el usuario. Una vez recopilada la información, la revisa y le aplica las tareas para las que esté designado, como sugerir al usuario determinados productos. Un ejemplo es el agente Kasbah [8], el cual está constituido por agentes vendedores y compradores para cada bien a ser vendido o comprado respectivamente. Las transacciones comerciales se dan a través de interacciones de los agentes.
  - ***Correo Electrónico:*** El agente Maxims [25] es un agente que trabaja como un filtro electrónico de correo, que aprende a darle prioridad, borrar, reenviar, buscar y archivar mensajes, observando cómo interactúa el usuario con el sistema de correo.
  - ***Equipo de trabajo:*** Hoover de *Sandpoint Corporation* (Cambridge, MA) es un agente que proporciona una interfaz de usuario para múltiples medios de información; organiza automáticamente información seleccionada de acuerdo a las necesidades del usuario; y pueden trabajar con los grupos de trabajo dirigiendo documentos, mandando notas, concertando reuniones y automatizando en definitiva los procesos estándar de una compañía.

- **Agentes en Aplicaciones Médicas**
  - ***Monitorización de pacientes:*** El sistema Guardián descrito por [16], está diseñado para ayudar a monitorear a los pacientes en la UCI (Unidad de Cuidados Intensivos). La tarea de monitorización está distribuida entre varios agentes de diferentes tipos:
    - Agentes de percepción/acción: responsables de la interfaz entre el sistema y el mundo exterior, captan información mediante sensores y actúan adecuadamente mediante efectores.
    - Agentes de razonamiento: responsables de los procesos de decisión.
    - Agentes de control: es único en el sistema y se encarga de las tareas de control entre los demás agentes.

    Estos agentes se organizan jerárquicamente y todos cooperan a través del conocimiento compartido en una estructura de datos común.

- **Aplicaciones para el entretenimiento**

Los juegos y las aplicaciones de entretenimiento se pueden beneficiar de los agentes, pues tienden a estar llenos de entes animados semi-autónomos.

- ***Juegos:*** Existe una versión del popular juego Tetris en el que el agente se pone de parte del jugador cuando ve que éste no puede colocar bien los bloques. Para esto se utiliza un agente reactivo.
- ***Teatro y Cine interactivo:*** Por teatro y cine interactivo se designa a un sistema que permite al usuario desempeñar un rol análogo a los roles de la vida real, o actores humanos en películas, e interactuar con los actores de la película o la obra.

2.7. ONTOLOGÍA

Una ontología es una descripción formal de los conceptos y relaciones que pueden existir en una determinada comunidad de agentes. A través de la ontología se busca compartir conocimiento utilizando un mismo vocabulario de forma coherente y consistente.

Los elementos de una ontología son [14]:

- Conceptos: cualquier cosa sobre la que se pueda emitir un juicio o comentario.
- Relaciones: representan diferentes tipos de interacción entre conceptos del dominio.
- Funciones: son un caso especial de relaciones.
- Instancias: representan cada uno de los elementos.
- Axiomas: sentencias que son siempre verdaderas en ese dominio.

Las dos principales fuentes de categorías ontológicas son:

- La observación, que proporciona conocimiento del mundo físico.
- El razonamiento, que da sentido a las observaciones generando un marco de abstracción llamado metafísica.

La ontología define las clases de cosas que existen en el dominio de aplicación, evitando que los términos y símbolos estén mal definidos o sean confusos. La elección de las categorías ontológicas es el primer paso en el diseño de una base de conocimiento. Esta selección de categorías determinará todas las cosas que pueden ser representadas en nuestro sistema de agentes. Un agente debe representar sus conocimientos en el vocabulario de una ontología específica. El creador del agente debe utilizar una ontología específica para representar el conocimiento del agente.

El vocabulario del lenguaje empleado para comunicarse entre agentes consiste en un diccionario de palabras apropiado para áreas de aplicación comunes. Cada palabra en el diccionario tiene una descripción (escritura en lenguaje natural) que es usada por las personas para entender su significado y una anotación formal que es usada por los programas. El diccionario es abierto, es decir, es posible añadir nuevas palabras dentro de áreas existentes y en nuevas áreas de aplicación.

La existencia de este diccionario no significa que solamente hay una manera de describir un área de aplicación. Un diccionario puede contener múltiples ontologías para un área dada y un agente puede utilizar la ontología que le sea más conveniente. Las definiciones formales asociadas con cualquiera de estas ontologías pueden ser utilizadas por los agentes para traducir mensajes que usan una ontología en específico a mensajes que usan otras ontologías.

2.8. SISTEMA MULTIAGENTE (SMA)

Según [10], un SMA es un sistema que reúne los siguientes elementos:

1. Un entorno
2. Un conjunto de objetos. Estos objetos se encuentran integrados con el entorno, por ejemplo, es posible en un momento dado asociar uno de estos objetos con un lugar en el entorno. Estos objetos son pasivos, pueden ser percibidos, creados, destruidos y modificados por agentes.
3. Un conjunto de agentes que se consideran como objetos especiales que representan las entidades activas del sistema.
4. Un conjunto de relaciones que unen objetos, y por lo tanto, agentes.
5. Un conjunto de operaciones, que hacen posible que los agentes perciban, produzcan, consuman, transformen y manipulen objetos.
6. Operadores que representan la aplicación de operaciones sobre el mundo y la reacción de éste al ser alterado. Estos operadores se pueden entender como *las leyes del universo*.

Las características de los SMA son:

- Cada agente tiene información incompleta, y no posee todas las capacidades para resolver el problema por si mismo, así cada agente tiene un punto de vista limitado.
- No existe un sistema de control global.
- Los datos son descentralizados.
- La computación es asincrónica.

Las razones del interés por los SMA abarcan:

- Proveer robustez y eficiencia en sistemas distribuidos.
- La habilidad para coexistir con sistemas actuales.
- Las ventajas del enfoque para resolver problemas en los cuales los datos, la experiencia y/o el control es distribuido.

Sin embargo, los SMA presentan también una serie de inconvenientes:

- Cómo formular, describir, descomponer y distribuir problemas, y cómo sintetizar los resultados entre un grupo de agentes.
- Cómo permitir que los agentes se comuniquen e interactúen. Qué lenguaje y protocolo usar.
- Cómo asegurar que los agentes realicen tareas coherentemente.
- Cómo balancear la computación local y la comunicación.
- Cómo diseñar SMA.

Los SMA proponen ayudas metodológicas de ingeniería de software, en este caso metodologías de *ingeniería del software orientada a agentes* y notaciones para resolver los inconvenientes anteriores. Es decir, herramientas de desarrollo que son específicamente concebidas para crear sistemas basados en agentes.

Ejemplos de metodologías y notaciones de ingeniería de software orientada a agentes, son:

- **Vocales** (*Voyelles*) de Yves Demazeau es una de las primeras propuestas en el área, y considera la concepción de sistemas multiagente desde varios puntos de vista, correspondientes a las vocales: Agente, Entorno, Interacciones, y Organización.
- **GAIA** de Michael Wooldridge y Nick Jennings de la Universidad de Southampton, propone cómo realizar un análisis basado en roles del sistema multiagente.
- **MASE** de Scott A. Deloach propone agentes como extensiones de objetos y proporciona la herramienta *AgentTool* para análisis, diseño e implementación.
- **AgentUML** de James Odell, propone una notación, extendiendo UML, para especificar protocolos de comunicación entre agentes.
- **MADKiT** es una herramienta de desarrollo, propuesta por Jacques Ferber, basada en el paradigma Agente-Rol-Organización de la metodología *Aalaadin*.
- **ADELFE** del grupo IRIT de la Universidad de Toulouse, trata especialmente los temas de cooperación entre agentes.
- **INGENIAS** del grupo GRASIA de la Universidad Complutense de Madrid, extiende la metodología *MESSAGE (Methodology for Engineering Systems of Software AGEnts*) y

proporciona un conjunto de herramientas para modelar y generar código de sistemas multiagente.

- **Mas-CommonKADS** de Carlos Iglesias en la Universidad Politécnica de Madrid extiende la metodología CommonKADS, para sistemas expertos, a agentes, utilizando estructuración orientada a objetos y lenguajes de especificación de protocolos como SDL.

3. SISTEMAS DE TUTORÍA INTELIGENTE SOPORTADOS POR EL HIPERMEDIA

3.1. HIPERMEDIA

La hipermedia surge como resultado de la fusión de dos tecnologías, el hipertexto y la multimedia. El hipertexto es la organización de una determinada información en diferentes nodos, conectados entre sí a través de enlaces. Los nodos pueden contener sub-elementos con entidad propia. Un hiperdocumento estaría formado por un conjunto de nodos conectados y relacionados temática y estructuralmente.

La tecnología multimedia es la que permite integrar diferentes medios (sonido, imágenes, videos, etc.) en una misma presentación.

La hipermedia, por tanto, es la tecnología que permite estructurar la información de una manera no-secuencial, a través de nodos interconectados por enlaces. La información presentada en estos nodos podrá integrar diferentes medios (texto, sonido, gráficos, videos, etc.).

El diseño de sistemas hipermedia o hiperdocumentos puede ser abarcado desde una doble vertiente: el diseño de la información y el diseño de la navegación.

3.1.1. Diseño de la información

El usuario, ante un nodo (por ejemplo, una página Web), realiza una revisión de la información presentada, seleccionando aquella que es de su interés. Un buen diseño de la información, desde el punto de vista organizativo y de su utilidad, es aquel que ayude al

usuario a encontrar la información que busca de la forma más fácil, rápida y cómoda posible.

Uno de los aspectos más importantes en el diseño de la información es evitar la sobrecarga informativa: demasiada información (textual, visual, etc.) en un mismo nodo, porque confunde y agota al usuario. Asimismo, la legibilidad del texto (tipo y tamaño de fuente, contraste entre el color de la fuente y el fondo, etc.) es un factor muy importante para tener en cuenta.

La redacción de los contenidos debe realizarse en un lenguaje entendible fácilmente por los potenciales usuarios del sistema, evitando tecnicismos complejos, abreviaturas innecesarias o acrónimos poco comunes.

Para facilitar la exploración de la información por parte del usuario se debe jerarquizar:

- Aumentando el tamaño de los textos de mayor importancia (títulos, subtítulos, etc.).
- Agrupando la información que esté relacionada.
- Utilizando efectos tipográficos (negrita, cursiva, etc.) para enfatizar contenidos.
- Utilizando el contraste en el color para discriminar y distribuir informaciones.
- Ubicando la información más relevante en zonas visuales superiores. Si el usuario no se ve obligado a utilizar la barra de desplazamiento para encontrar la información que busca, ahorrará tiempo en su búsqueda y tendrá más probabilidades de encontrarla.

3.1.2. Diseño de la navegación

El diseño de la navegación consiste en definir la arquitectura del hipermedia: elementos de interacción entre el usuario y el sistema, enlaces y tipos de enlaces entre los nodos, agrupación de los nodos por categorías o propiedades, y respuestas del sistema ante peticiones del usuario.

Para diseñar la navegación se puede usar el vocabulario gráfico propuesto por Jesse James Garrett[6] para la descripción de la arquitectura de la información y el diseño de la interacción:

Mediante diagramas se puede documentar el diseño de la navegación: organización de la información en nodos, los enlaces y sus tipos, acciones permitidas al usuario, etc.

Una vez definida la arquitectura, se debe implementar los elementos de interacción en el hipermedia: enlaces, opciones o menús de navegación, componentes de interacción (botones, cajas de texto, etc.), etc.

La interacción usuario-hipermedia debe poder realizarse con la menor carga cognitiva para el usuario, por lo que se recomienda:

- Evitar la sobrecarga memorística: los menús o barras de navegación deben contener un máximo de 7 opciones diferentes.
- El usuario debe poder predecir la respuesta del sistema ante su acción, para ello el nombre de los enlaces y componentes de interacción debe ser significativo y preciso. Los globos de texto pueden ser de mucha utilidad en este sentido.
- Se debe ofrecer ayudas al usuario en procesos de interacción complejos (formularios, etc.).
- Los mensajes de error deben ser explicados de forma clara y no alarmista, indicando al usuario vías alternativas para resolver el problema.

3.2. EL e-LEARNING

La educación en línea (e-Learning) implica una nueva forma de aprender. No quiere decir que es totalmente diferente a cómo se aprende en la educación tradicional. Más bien quiere decir, que es una forma enriquecida de aprender, ya que integra nuevas fuentes y formas de adquirir conocimiento.

[6] http://www.jjg.net/ia/visvocab/spanish.html

El e-learning está referido al uso de las tecnologías para dar una variedad de soluciones que mejoren la enseñanza, la obtención del conocimiento y el rendimiento de los estudiantes. El e-learning, permite construir y poner a disposición del usuario, cursos educativos y de entrenamiento de alta calidad. Para que sea exitoso, es necesario plantear estrategias pedagógicas a través de la combinación de las nuevas tecnologías en la red asociadas a los nuevos enfoques de cómo aprenden las personas.

3.2.1. Beneficios

Dentro de los beneficios del e-Learning se encuentran los siguientes:

- **Bajos costos del *e-learning:*** es la forma más económica de hacer llegar la instrucción o información. Elimina los costos de viajes, reduce el tiempo de entrenamiento de las personas y reduce significativamente las necesidades de infraestructura.
- ***El e-Learning* mejora la capacidad de respuesta de los negocios:** puede llegar a un número ilimitado de personas simultáneamente. Esto puede ser crítico cuando las prácticas y capacidades de negocios deben cambiar rápidamente debido al elevado nivel de competencia.
- **Mensajes consistentes y adaptados a las necesidades:** las personas pueden acceder a los mismos programas educativos presentados en diversas formas. Aún los programas pueden ser adaptados a diferentes necesidades o a diferentes grupos de personas.
- **El contenido es más oportuno y más confiable:** el contenido temático de los programas educativos disponible en la red, puede ser actualizado instantáneamente, haciendo que sea más exacto y usable para largos periodos de tiempo. La habilidad para actualizar los contenidos de manera fácil y rápida, para luego distribuirla de una manera ágil y dinámica, hacia un gran número de trabajadores y clientes, es una oportunidad para relacionarse con las personas en un proceso de cambios acelerados.
- **El aprendizaje es 24/7:** Las personas pueden acceder al proceso de enseñanza/aprendizaje en cualquier lugar y tiempo. Este enfoque “justo a tiempo – en

cualquier tiempo" hace que las operaciones de aprendizaje sean verdaderamente globales.

- **Reducción del tiempo empleado por el usuario:** Millones de personas sienten comodidad con la tecnología de Internet; aprender a acceder los recursos es un proceso fácil y rápido.
- **Universalidad:** está disponible en la red y toma las ventajas de los protocolos universales de Internet. Las incompatibilidades entre diferentes plataformas y sistemas operativos, están disminuyendo en forma sustancial.
- **Comunidades de desarrollo:** La red permite construir comunidades que pueden compartir el conocimiento, que perdura hasta después de finalizados los cursos. Esto puede ser motivador para el aprendizaje organizacional.
- **Influencia de las inversiones de la corporación en la red:** Los ejecutivos están incrementando sus expectativas en relación a las inversiones en intranets corporativas. El e-Learning está emergiendo como una de sus aplicaciones.

3.2.2. Ventajas

Algunas de las ventajas en la utilización de plataformas de e-learning como instrumento de formación personal son las siguientes:

- Diversificación y ampliación de la oferta de los programas educativos.
- Desarrollo de habilidades en el uso de la tecnología, brindando la posibilidad de acceso a información actualizada a través de Internet.
- Eficaz combinación de estudio y trabajo.
- Formación fuera del contexto del salón de clase.
- Ofrece alternativas para los diferentes ritmos de aprendizaje del estudiante o para diferentes niveles de profundidad dados por el docente.
- Comunicación bidireccional frecuente, garantizando un aprendizaje dinámico e innovador.
- Reducción de los gastos personales para entrenamientos presenciales (transporte, hospedaje, etc.)

- Aumenta el tiempo de dedicación para las actividades académicas, evitando la limitación de horarios, desplazamientos y canales limitados de comunicación.
- Permite generar verdaderos procesos de autoevaluación y diversas formas de evaluación, que convierten el proceso educativo en algo más dinámico, participativo e interactivo

3.3. SISTEMAS TUTORES INTELIGENTES

Un Sistema Tutor Inteligente (STI) es un sistema de enseñanza asistida por computadora, que utiliza técnicas de Inteligencia Artificial, principalmente para representar el conocimiento y dirigir una estrategia de enseñanza. Es capaz de comportarse como un experto, tanto en el manejo del modelo del dominio de conocimiento que se enseña (mostrando al alumno cómo avanzar en el conocimiento), como en el dominio pedagógico (diagnosticando la situación en la que se encuentra el estudiante y de acuerdo a ello ofrecer una acción o solución que le permita progresar en el aprendizaje). [20]

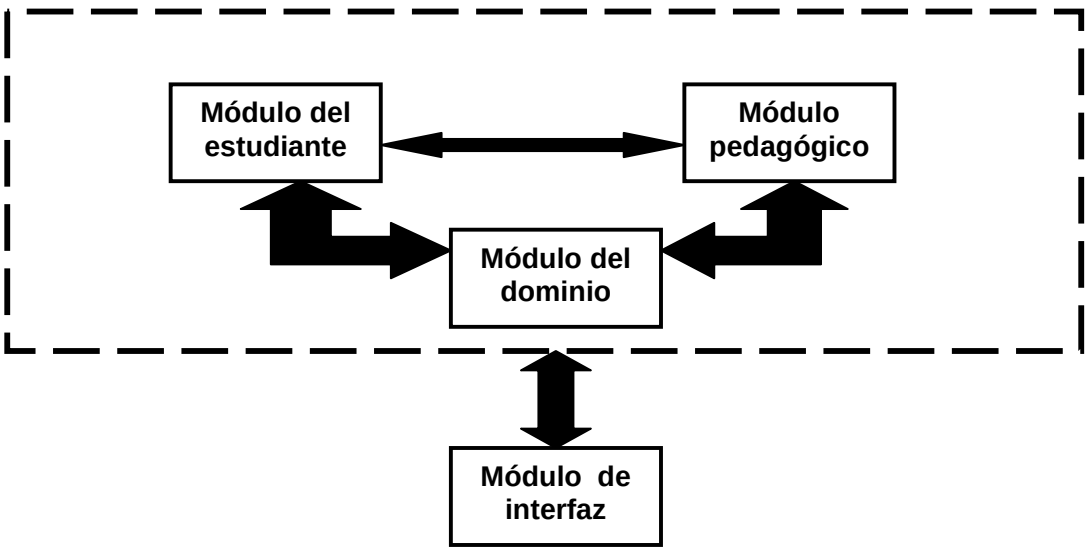


Figura 8. Arquitectura de un Sistema Tutor Inteligente (tomado de [21])

Un STI debe ser capaz de identificar la forma en que el estudiante está solucionando el problema y brindarle la ayuda necesaria cuando cometa errores, determinar el

conocimiento que necesita para resolver un problema y explicar ese conocimiento en el momento apropiado. La idea es producir sistemas expertos que puedan simular el comportamiento del estudiante, de manera que cuando el alumno dé una respuesta, el sistema pueda simular el comportamiento del tutor para poder guiarlo adecuadamente.

Arquitectura

Los STI poseen una estructura modular donde cada uno, además de desempeñar una función específica dentro de la arquitectura, está interrelacionado y en sincronía con los demás (Ver figura 8). [7]

- **Módulo del estudiante:** representa el conocimiento y las habilidades cognitivas del alumno en un momento dado. Contiene una representación del estado del conocimiento del alumno en el momento que interactúa con el STI. A partir de ese modelo y del contenido representado en la base del dominio, el sistema debe ser capaz de inferir la mejor estrategia de acción a ser utilizada para cada alumno. Este módulo almacena información específica para cada estudiante de forma individual. Como mínimo, debe mantener un histórico sobre cómo está trabajando el estudiante con el material en cuestión. Es interesante también mantener registro sobre las falencias del estudiante. [17]

- **Módulo pedagógico:** ofrece una metodología para el proceso de aprendizaje. Posee el conocimiento sobre las estrategias y tácticas para seleccionarlas en función de las características del alumno y determina la manera en que la información será representada. Ejecuta el diagnóstico del conocimiento del alumno [21]. Las entradas de este módulo son ofrecidas por el *Módulo del estudiante*.

- **Módulo del dominio:** almacena la información que el tutor está enseñando. El modelado del conocimiento a ser transferido es de gran importancia para el éxito del sistema como un todo. Se debe procurar una representación del conocimiento que esté preparada para el crecimiento incremental del dominio. Este modelo sirve como base para la construcción del *Modelo del estudiante* [21].

- **Módulo de interfaz:** permite la interacción entre el tutor y el alumno. Presenta el material apropiado al nivel de entendimiento del alumno y mantiene la coherencia en las explicaciones. Una interfaz puede tomar diferentes formas, por ejemplo, puede estar formada por preguntas y respuestas, menús, lenguaje natural, gráficos o una combinación de todas estas formas. [21].

4. METOLOGÍAS DE DESARROLLO DE SISTEMAS MULTIAGENTE (SMA)

Existen dos enfoques para el desarrollo de sistemas multiagente:

El *enfoque formal:* percibe el SMA como el resultado de utilizar un lenguaje de especificación de agentes. Para generar SMA de esta manera, se parte de principios basados en modelos operacionales y formales de SMA.

El *enfoque constructivista:* estudia el SMA como un sistema software que hay que construir. El desarrollo no parte de cero, sino que utiliza plataformas de desarrollo de agentes que proporcionan servicios básicos de comunicación, gestión de agentes y una arquitectura de agente.

En cualquiera de los dos casos, y más cuando el sistema a desarrollar es complejo, se necesitan metodologías que estructuren el desarrollo de acuerdo con las prácticas de ingeniería del software. Para ilustrar ambos enfoques, a lo largo de este capítulo se trataran los siguientes temas: lenguajes de agentes, plataformas de desarrollo y metodologías para el desarrollo de SMA.

Se hará énfasis en la plataforma de desarrollo JADE, ya que será utilizada para el desarrollo del agente generador de ejercicios interactivos del sistema multiagente de la plataforma educativa institucional e-ESCEN@RI_{UIS}. Para la documentación del agente se utilizara AgentUML.

4.1. LENGUAJES DE AGENTES

Lenguaje de agente se entiende como un sistema que permite programar sistemas de hardware o de software en términos de algunos de los conceptos desarrollados por las

teorías de agente. Al menos, se espera que tal lenguaje incluya alguna estructura correspondiente al agente. Sin embargo, se podría esperar que tuviera también otros atributos de agencia[7] (creencias, metas, u otras nociones mentales) utilizados para programar agentes. Algunos de los lenguajes que se presentarán contienen esta noción fuerte de agencia; otros no. Sin embargo, todos tienen propiedades que los hacen interesantes.

Se pueden distinguir dos tipos principales de lenguajes de programación [18]:

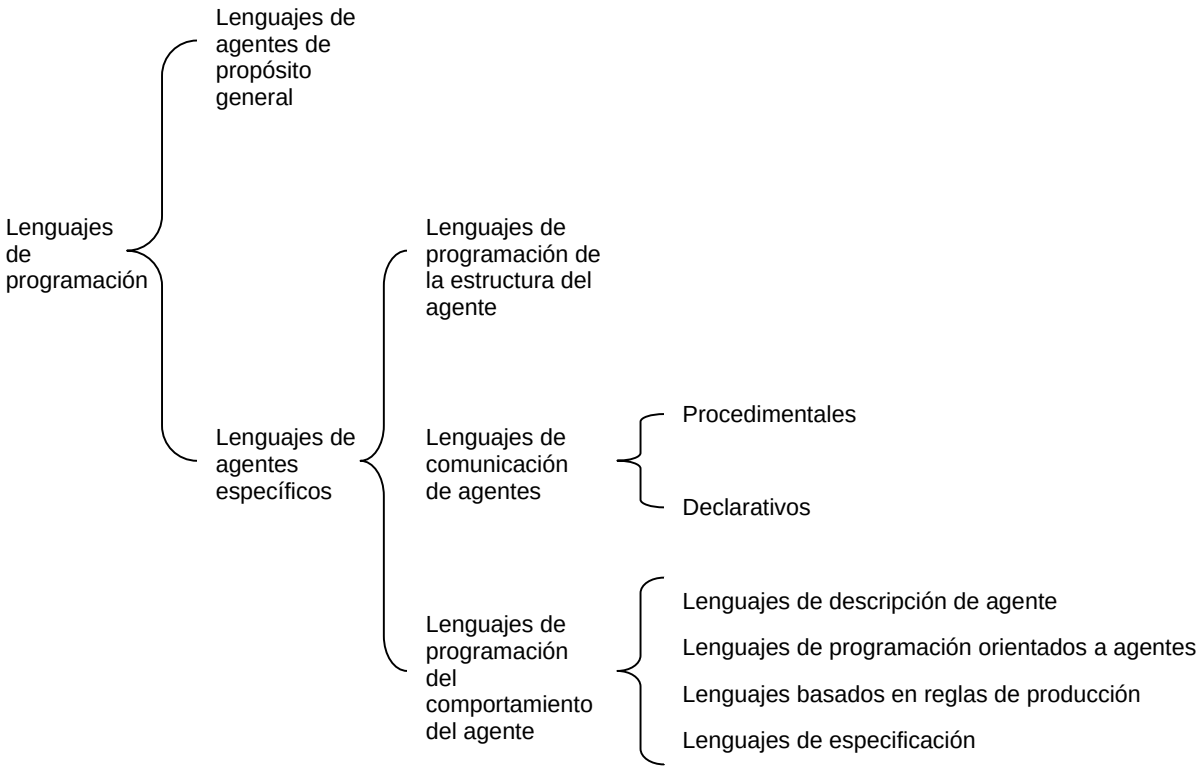


Figura 9. Tipos de lenguajes de programación

- **Lenguajes de agentes de propósito general:** lenguajes destinados a programar agentes genéricos utilizables en cualquier aplicación.

[7] Ensamblado de partes (agentes) con una tarea resultante de su unidad integrada, descartando lo que cada una de las partes realiza por sí mismas.

- **Lenguajes de agentes específicos:** lenguajes para un tipo de agentes específicos, por ejemplo los lenguajes para agentes móviles Telescript o Agent-Tcl.

  Se pueden distinguir los siguientes niveles en la programación de agentes:

  - ***Lenguajes de programación de la estructura del agente:*** permiten programar las funcionalidades básicas para definir a un agente: funciones de creación de procesos (creación del proceso agente y de los procesos concurrentes con él) y funciones de comunicación entre agentes (nivel de transporte).

    Este nivel de programación normalmente sólo es utilizado por los desarrolladores de una plataforma de desarrollo de agentes. Los lenguajes empleados suelen ser lenguajes de propósito general (C, C++, Java, Lisp, Prolog, etc.) o lenguajes específicos (por ejemplo: April dentro del proyecto IMAGINE, sobre Prolog/C y CUBL (*Concurrent Unit Based Language*) dentro del proyecto DAISY, sobre CLOS/C++).
  - ***Lenguajes de comunicación de agentes*:** definición del formato de los mensajes intercambiados, de las primitivas de comunicación y de los protocolos disponibles.

    Se destacan dos tipos de lenguajes de comunicación:
    - Procedimentales: se basan en el intercambio de directivas procedimentales, es decir, un agente recibe un mensaje que implica la ejecución de un procedimiento. Suelen emplear lenguajes de intérpretes de órdenes como Perl, Tcl, etc., permitiendo un rápido prototipado aunque no suelen ser fácilmente escalables ni reciclables. Son especialmente útiles para la construcción de agentes en aplicaciones finales como agentes de usuario o agentes móviles. Dentro de este enfoque se encuentran Sodabot o Telescript.
    - Declarativos: se basan en el intercambio de actos comunicativos, es decir, un agente recibe un mensaje con un acto comunicativo que le permite interpretar el contenido del mensaje. El ejemplo más extendido de este enfoque es KQML (*Knowledge Query Management Language*).
  - ***Lenguajes de programación del comportamiento del agente*:** permiten definir el conocimiento del agente: conocimiento inicial (modelo de entorno, creencias, deseos, objetivos), funciones de mantenimiento de dicho conocimiento (reglas, planes, etc.), funciones para alcanzar sus objetivos (planes, reglas, etc.) y funciones para desarrollar habilidades (programación de servicios).

Permiten la programación de los agentes: definición de su estructura, conocimiento y habilidades. Se destacan diferentes tipos:

- Lenguajes de descripción de agente: los agentes se derivan de una clase de agente genérica, permitiendo la definición de los elementos básicos del modelo de agente tales como base de conocimiento, grupos de agentes, habilidades del agente, servicios ofrecidos, planes para alcanzar objetivos, etc. La descripción se traduce a un lenguaje ya ejecutable. Hacen parte de este tipo de lenguaje: MACE ADL, AgentSpeak y MAST/ADL.
- Lenguajes de programación orientados a agentes: siguiendo el paradigma de Shoham [32] de la programación orientada a agentes (AOP), se han definido algunos lenguajes que tienen en cuenta el estado mental del agente para programar las funciones de transición entre estos estados mentales, que consisten en creencias, capacidades y obligaciones. Entre estos lenguajes se pueden citar AGENT0, PLACA y Agent-K.
- Lenguajes basados en reglas de producción: la programación de la base de conocimiento de los agentes se realiza en algunos sistemas empleando reglas de producción como, por ejemplo, MAGS (basado en OPS5), DYNACLIPS (arquitectura de pizarra basada en CLIPS) y RTA, que permite definir las conductas del agente con reglas.
- Lenguajes de especificación: emplean una especificación (normalmente lógica) del agente que se ejecuta directamente para generar su conducta, pudiendo verificar propiedades de la especificación. Se pueden citar como ejemplos a METATEM y DESIRE.

4.2. PLATAFORMAS DE DESARROLLO

Una plataforma de desarrollo es el entorno común en el cual se desenvuelve la programación de un grupo definido de aplicaciones. Comúnmente se encuentra relacionada directamente a un sistema operativo, sin embargo, también es posible encontrarlas ligadas a una familia de lenguajes de programación o a una Interfaz de

programación de aplicaciones o API por sus siglas en inglés.[8] A continuación se presentan algunas de las plataformas de desarrollo para agentes inteligentes analizadas.

4.2.1. Grasshopper

Grasshopper[9] es una plataforma para el desarrollo y la ejecución de agentes móviles escritos en lenguaje Java creada por la empresa alemana IKV++ y que cumple con las normas especificadas en MASIF (*Utilidades para la Interoperación entre Sistemas de Agentes Móviles*) por el O.M.G. (*Object Management Group*).

La arquitectura de esta plataforma consta de los siguientes elementos (ver figura 10):

- Agencia: entorno mínimo de ejecución para agentes móviles y estáticos.
- Núcleo de la agencia: contiene las funciones básicas de la agencia (comunicación, registro, seguridad y persistencia).
- Lugar: agrupación funcional lógica dentro de una agencia.
- Región: mantiene un registro de todos los componentes pertenecientes a una determinada organización.

Las características principales de Grasshopper son:

- Soporta interacción mediante CORBA[10], RMI[11] o *sockets*[12].
- Permite transparencia respecto a la localización de los objetos.
- La comunicación entre agentes puede ser asíncrona, dinámica o multipunto.
- Puede trabajar con los siguientes ORB[13]: JDK 1.2 de Sun, VisiBroker de Inprise y OrbixWeb de IONA.

[8] http://es.wikipedia.org/wiki/Plataforma_de_desarrollo
[9] *Grasshopper: The Agent Platform* (IKV++): http://www.ikv.de/products/grasshopper/
[10] CORBA (*Common Object Request Broker Architecture*) es un estándar que establece una plataforma de desarrollo de sistemas distribuidos facilitando la invocación de métodos remotos bajo un paradigma orientado a objetos.
[11] RMI (*Remote Method Invocation*) es el mecanismo ofrecido en Java que permite a un procedimiento (método, clase, aplicación) poder ser invocado remotamente.
[12] Socket designa un concepto abstracto por el cual dos programas (posiblemente situados en computadoras distintas) pueden intercambiarse cualquier flujo de datos, generalmente de manera fiable y ordenada. Un *socket* queda definido por una dirección IP, un protocolo y un número de puerto.
[13] ORB (*Object Request Broker*). En computación distribuida es el nombre que recibe una capa de software (también llamada middleware) que permite a los objetos realizar llamadas a métodos situados en máquinas remotas, a través de una red.

- Los agentes, otros entornos y las aplicaciones normales pueden acceder a la funcionalidad de agencias remotas y del registro de la región.
- Cada agencia mantiene servicios de seguridad interna y externa.
- Soporta las siguientes operaciones para la gestión de agentes: creación, borrado, suspensión, reanudación, clonación, copia, migración, almacenamiento e invocación de acciones.

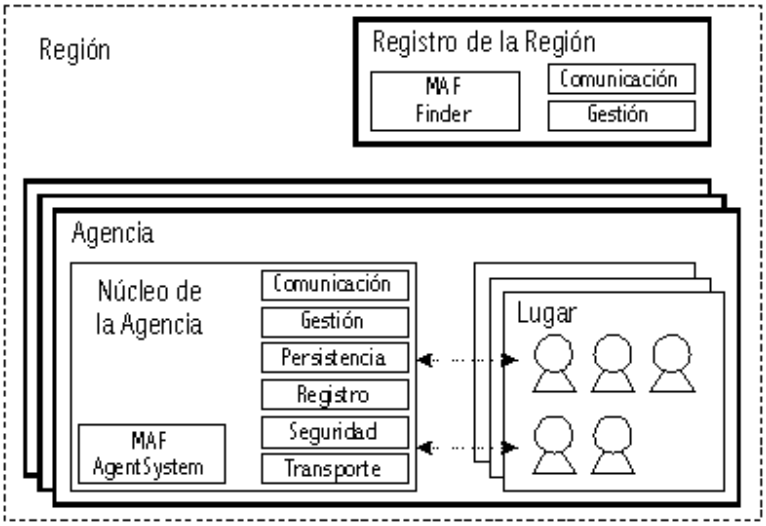


Figura 10. Arquitectura de Grasshopper (tomada de [38])

4.2.2. ZEUS

ZEUS[14] es una herramienta para construir aplicaciones multiagente colaborativas. Provee un entorno integrado para el desarrollo rápido de sistemas. ZEUS define una metodología de diseño de sistemas multiagente y lo soporta mediante un entorno visual para capturar las especificaciones de los agentes. Estas especificaciones son luego utilizadas para generar el código fuente en Java.

La herramienta ZEUS consiste de un conjunto de componentes, escritas en el lenguaje de programación Java, que puede ser categorizada en tres grupos funcionales o librerías:

[14] ZEUS. *The ZEUS Agent Building Tool*. British Telecommunications, http://more.btexact.com/projects/agents/zeus/, 2002.

una librería de componentes de agentes (*Agent Component Library*), una herramienta de construcción de agentes (*Agent building software*) y un conjunto de agentes utilitarios entre los cuales se pueden encontrar servidores de nombres, facilitadores y agentes visualizadores. A continuación se describe cada una de ellas.

Librería de componentes de agentes: es una colección de clases que forman los bloques de construcción de los agentes individuales. El contenido de esta librería muestra los siguientes puntos:

- Bloques de construcción de agentes.
- Proporciona mecanismos de comunicación
  - Lenguaje de comunicación (ACL[15], KQML[16]): provee un lenguaje de comunicación entre agentes basado en actos del habla y en *performatives.*
  - Mecanismos de paso de mensajes (TPC/IP[17]): posee un sistema de pasaje de mensajes asincrónico basado en *sockets.*
- Gestión y uso de ontologías: posee un editor para describir ontologías de dominio específico y un lenguaje de representación de conocimiento basado en marcos para representar los conceptos del dominio.
- Librerías de protocolos.

Herramienta de construcción de agentes: esta herramienta está diseñada para proveer un desarrollo rápido de agentes a alto nivel, ocultando la complejidad de la librería de componentes de agentes.

- Posee editores que ayudan al usuario a crear los agentes utilizando los "bloques" o componentes software
- Sigue la metodología propia en 6 etapas:
  - Análisis (identificación de agentes): definición de los ítems de la ontología en un dominio (Editor de ontologías). También se describen las instancias específicas de hechos y variables, utilizando las plantillas creadas usando el editor de ontologías (Editor de hechos/variables).

[15] *Agent Communication Lenguaje.* Estructura de comunicación entre agentes definida por la FIPA (*Foundation for Intelligent Physical Agents*)
[16] KQML *Knowledge Query and Manipulation Language*, Lenguaje de consulta y manipulación del conocimiento.
[17] Transmission Control Protocol/Internet Protocol, es el lenguaje que rige todas las comunicaciones entre todos los computadores en Internet. TCP/IP es un conjunto de instrucciones que dictan cómo se han de enviar paquetes de información por distintas redes.

- Definición de agentes (atributos): especificación de los atributos de las tareas y para resúmenes gráficos de las tareas. (Editor de definiciones de agentes).
- Organización de agentes: definición de las relaciones organizacionales entre los agentes, y las creencias de los agentes acerca de las habilidades de otros agentes (Editor de organización).
- Definición de las tareas de los agentes: descripción de los agentes lógicamente. Esto involucra la especificación de las tareas de cada agente, sus recursos iniciales y las dimensiones de su plan. (Editor de definición de tareas)
- Coordinación de agentes (protocolos): selección del conjunto de protocolos de coordinación con los cuales cada agente estará equipado. (Editor de coordinación)
- Generación automática de código (JAVA)

Agentes utilitarios: Todos los agentes utilitarios son construidos utilizando las componentes básicas de la librería de componentes de agentes, y son en realidad simplificaciones del agente genérico ZEUS. Los agentes utilitarios constan de:

- Un servidor de nombres.
- Un facilitador para el descubrimiento de información.
- Un agente para visualizar o realizar una *depuración* de sociedades de agentes. Una sociedad de agentes en ZEUS puede contener cualquier número de agentes utilitarios, con al menos un servidor de nombres.

4.2.3. JADE

JADE (*Java Agent DEvelopment Framework*) es una plataforma de software desarrollada en *TILab*[18] bajo la filosofía de código abierto, que proporciona tanto un entorno de desarrollo como uno de ejecución para la realización y mantenimiento de sistemas multiagente.

El entorno de desarrollo está formado por una serie de librerías en Java que permiten la implementación de agentes de forma independiente de la plataforma sobre la que se va a ejecutar.

[18] Telecomm Italia Lab

El entorno de ejecución permite a los agentes *vivir* y comunicarse entre ellos. Está realizado enteramente en Java y proporciona una serie de herramientas que permiten al desarrollador controlar y depurar a los agentes en tiempo real.

El proyecto JADE comenzó en 1998, y la primera versión (v. 1.3) estuvo lista en febrero del 2000, y fue lanzada bajo licencia LGPL[19].

JADE cumple con las especificaciones de la FIPA (*Foundation for Intelligent Physical Agents*):

- Arquitectura: Esto da muchas ventajas a la hora de la integración de diferentes aplicaciones, incluso con plataformas de diferentes propietarios.
- Lenguaje de comunicación empleado FIPA-ACL.
- Servicios de agentes: ciclo de vida, páginas blancas, páginas amarillas, transporte de mensajes, etc.
- Conjunto de herramientas gráficas que soportan la depuración y ejecución de agentes (RMA, sniffer, etc.)

FIPA

FIPA (*Foundation for Intelligent Physical Agents*) es una organización internacional que se dedica a promover la industria de agentes inteligentes abiertamente desarrollando las especificaciones que apoyan la interoperabilidad entre agentes. Esto ocurre con la colaboración entre sus organizaciones miembro, que son las compañías y las universidades que están activas en el campo de agentes[20].

FIPA define el modelo de una plataforma basada en agentes, el conjunto de servicios que debe proveer y la interface entre los servicios.

[19] Lesser General Public License. Es una licencia que pretende garantizar su libertad de compartir y modificar el software "libre", esto es para asegurar que el software es libre para todos sus usuarios.
[20] http://www.fipa.org

Características de JADE

Jade presenta las siguientes características [3]:

- **P2P:** Arquitectura peer-to-peer, cada agente puede tomar la iniciativa en una comunicación o bien responder a peticiones que le hagan otros agentes.
- **Interoperabilidad:** JADE cumple con las especificaciones FIPA, por lo que los agentes desarrollados en JADE pueden interactuar con otros agentes que no tienen porque estar desarrollados con JADE, aunque si deben seguir las especificaciones FIPA.
- **Portabilidad:** La API que proporciona JADE es independientemente de la red sobre la que va a operar, así como de la versión de Java utilizada, teniendo la misma API para J2EE, J2SE y J2ME.
- **Intuitiva:** JADE se ha desarrollado para ofrecer una API fácil de aprender y sencilla de manejar.

Los agentes JADE tienen nombres únicos y se permite a cada agente descubrir a otros agentes y comunicarse con ellos mediante comunicaciones punto a punto. Los agentes proporcionan servicios, cada agente puede buscar a otros dependiendo de los servicios que proporcionen otros agentes.

La comunicación entre agentes se lleva a cabo a través de mensajes asíncronos, es decir, el agente que envía el mensaje y el destinatario del mensaje no tienen porqué estar disponibles al mismo tiempo. Es más, el destinatario no tiene porqué existir en ese instante. Los mensajes se pueden enviar a un agente en concreto o se pueden enviar a agentes que se desconocen pero se sabe que poseen unas ciertas características.

JADE proporciona mecanismos de seguridad, ya que habrá agentes a los que no se les esté permitido comunicarse con otros agentes, de manera que una aplicación puede verificar la identidad del receptor y del agente que envía el mensaje y no dejar realizar actuaciones no permitidas para un determinado agente. La estructura de los mensajes se basa en el lenguaje ACL (*Agent Communication Lenguaje*) que ha sido definido por la FIPA.

JADE también permite a los agentes cambiar de host (en J2SE). Un agente puede interrumpir en un momento dado su ejecución, migrar a otro host y continuar la ejecución en el mismo punto en el que la interrumpió. Esto permite un balanceo de carga ya que permite a los agentes migrar a hosts menos cargados.

El entorno de ejecución proporciona un marco donde poder ejecutar los agentes y herramientas gráficas para su monitorización y depuración.

Arquitectura

Los agentes JADE necesitan del entorno de ejecución donde poder *"vivir"*. Cada instancia del entorno de ejecución se denomina *contenedor* (*container*).

Al conjunto de los contenedores se le denomina *plataforma* (*platform*) y proporciona una capa que oculta a los agentes (y al desarrollador) el entorno donde se ha decidido ejecutar la aplicación.

En cada plataforma debe existir un contenedor especial denominado *contenedor principal* (*main container*). La principal diferencia del contenedor principal respecto al resto de contenedores es que alberga dos agentes especiales: el AMS (*Agent Management System*) y el DF (*Directory Facilitator*)

A continuación se describe el modelo de referencia FIPA de plataforma de agentes y se describe la funcionalidad de cada uno de sus componentes:

- **AMS (*Agent Management System*):**
  - Garantiza que cada agente en la plataforma tenga un único nombre.
  - Encargado de proporcionar los servicios de páginas blancas y ciclo de vida, y de mantener el directorio de los identificadores de agentes (AID: *Agent Identifier*) y su estado.
  - Cada agente debe registrarse con el AMS para obtener un AID válido.

- Es posible crear y matar agentes en contenedores remotos si se le requiere al agente AMS

- **DF (*Directory Facilitator*):**
  - Agente que proporciona el servicio de páginas amarillas.
  - Un agente puede encontrar otros agentes que proporcionan los servicios que requiere para lograr sus objetivos

- **ACC (*Agent Communication Channel*):**
  - Gestiona el envío de mensajes entre agentes de la misma plataforma o de plataformas distintas.
  - Permite la migración de agentes.

La arquitectura se puede observar en la figura 11 donde aparecen dos plataformas diferentes (*platform1* y *platform2*), cada una con sus contenedores principales y contenedores normales.

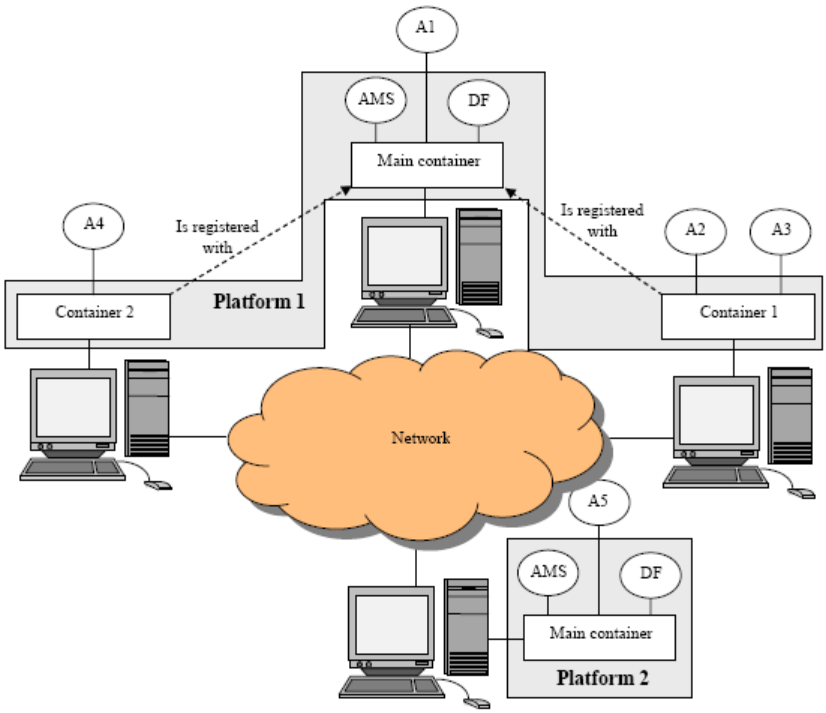


Figura 11. Esquema de distribución de los contenedores y las plataformas (tomada de [39])

Comportamientos

Un comportamiento es básicamente un conjunto de eventos, un método que describe cómo un agente reacciona a un evento. Formalmente, un evento es un cambio relevante del estado, como por ejemplo: la recepción de un mensaje o de una interrupción del contador de tiempo. En Jade, los comportamientos son clases y el código del evento se pone en un método llamado *action*.

Los comportamientos se implementan como un objeto de la clase *jade.core.behaviours.Behaviour*. Los objetos *Behaviour* describen pequeñas tareas que debe realizar el agente ejecutándose según un planificador que se encuentra implementado en la clase *Agent*. El planificador va ejecutando según una política round robin[21] los objetos *behaviour* que se encuentran en una cola FIFO[22], existiendo dos métodos, *addBehaviour(behaviourObjebct)* y *removeBehaviour(behaviourObject)*, que permiten gestionar la entrada y salida de los objetos *behaviour* en la cola del planificador.

Los objetos *behaviour* tienen dos métodos que se deben reescribir, *action()* y *done()*. El método *action()* es en donde se deben desarrollar las tareas que debe realizar el agente, mientras que el método *done()* es llamado cuando *action()* finaliza. El planificador va ejecutando uno a uno los métodos *action()* de la cola y cuando el método *action()* de un objeto finaliza se llama al método *done()*, este debe retornar un booleano, si retorna *true*, el objeto es sacado fuera del planificador ya que se da por concluida su tarea, mientras que si retorna *false* se vuelve a planificar.

Si en algún momento de la ejecución del método *action()* se requiere esperar por la llegada de un mensaje, existe el método *block()* para mandar a una cola de bloqueados cuando el método *action()* finaliza (no cuando se llama a *block()*). Cuando se produce la llegada de un mensaje, todos los objetos en la cola de bloqueados se planifican y deben

[21] Round robin es un método para seleccionar todos los elementos en un grupo de manera equitativa y en un orden racional, normalmente comenzando por el primer elemento de la lista hasta llegar al último y empezando de nuevo desde el primer elemento.
[22] First Input First Output. Es un método utilizado en estructuras de datos, contabilidad de costos y teoría de colas. Guarda analogía con las personas que esperan en una cola y van siendo atendidas en el orden en que llegaron, es decir, que *la primera persona que entra es la primera persona que sale*.

comprobar si el mensaje llegado es para ellos o no. En caso de que un objeto no sea el destinatario del mensaje debe volver a bloquearse.

Es importante que los métodos *action()* sean cortos, de esta forma se permite un cierto grado de paralelismo. Si se necesita realizar una tarea que va a requerir un largo periodo de tiempo, entonces se deberá dividir la tarea en subtareas y cada una de ellas implementarla mediante un objeto *behaviour*. Con el fin de ayudar en esta labor, JADE proporciona una jerarquía de clases *Behaviour*, que permitirán hasta simular Máquinas de Estados Finitos.

- **Clase SimpleBehaviour:** Representa un comportamiento atómico.
- **Clase OneShotBehaviour**: Representa un comportamiento que se debe ejecutar solo una vez, por eso, su método *done()* retorna *true*, si no es redefinido.
- **Clase CyclicBehaviour:** Representa un comportamiento que debe ejecutarse una serie de veces. El método *done*, si no es redefinido, devuelve *false*.
- **Clase CompositeBehaviour:** Esta clase se compone de diferentes sub-comportamientos que se pueden ejecutar siguiendo diferentes políticas de planificación. Las diferentes políticas vienen determinadas por la subclase elegida, *SequentialBehaviour*, *ParallelBehaviour* y *FSMBehavior*.
- **Clase SequentialBehaviour:** Esta clase deriva de *CompositeBehaviour* y ejecuta sub-comportamientos de forma secuencial, y termina cuando todos los métodos *action()* han terminado.
- **Clase ParallelBehaviour:** Esta clase deriva de *CompositeBehaviour* y ejecuta los sub-comportamientos de manera concurrente. En el constructor de la clase se puede especificar cuando se desea que acabe la ejecución: cuando todos los sub-comportamientos lo han hecho, cuando uno termine, o cuando un número especificado lo logre.
- **Clase FSMBehaviour:** Esta clase permite definir una Máquina de Estados Finita mediante sub-comportamientos. Cada sub-comportamiento representa un estado de la máquina, y las transiciones se van produciendo según la salida de dichos estados. La finalización se alcanza cuando se termine de ejecutar algún sub-comportamiento que se haya registrado como estado final.

- **Clase SenderBehaviour:** Encapsula la acción de envío de un mensaje ACL. Este mensaje se le debe especificar en el constructor.
- **Clase ReceiverBehaviour:** Encapsula la acción de recepción de un mensaje ACL. Termina cuando se recibe el mensaje o cuando pasa una cierta cantidad de tiempo especificada en el constructor.
- **Clase WakerBehaviour:** Implementa un comportamiento Honesto que se ejecuta justo después de que se haya pasado un tiempo especificado.

Comunicación entre agentes

La comunicación entre los agentes es una de las más importantes características que aporta JADE. Se basa en un modelo de paso de mensajes asíncrono. Cada agente tiene un buzón en el cual se van almacenando los mensajes enviados por otros agentes. Cuando llega un mensaje nuevo, se le notifica al agente que lo ha recibido para que lo procese.

Los mensajes intercambiados entre los agentes siguen un formato concreto que ha sido definido por la FIPA denominado ACL. El formato ACL define los diversos campos que debe constar un mensaje:

- **sender:** El remitente del mensaje.
- **receivers:** La lista de destinatarios.
- **performative:** El objetivo de la comunicación. El remitente debe indicar si la intención de la comunicación que se va a establecer es comunicar un suceso al destinatario (*INFORM*), solicitar que el destinatario realice una acción determinada (*REQUEST*), preguntar por una condición (*QUERY-IF*) o establecer una comunicación algo más compleja (CFP[23], PROPOSE, ACCEPT_PROPOSAL, REJECT_PROPOSAL)
- **content:** El contenido del mensaje.
- **language:** El tipo de sintaxis utilizado en *content*.
- **ontology:** El diccionario de símbolos usados en *content*.

[23] Call For Proposal

- **Protocol:** El protocolo empleado por el remitente del mensaje
- Campos de control usados para controlar conversaciones concurrentes y timeouts.

Los mensajes son implementados como objetos de la clase *jade.lang.acl.ACLMessage* y proporciona métodos *get y set* para acceder a los diversos campos de un mensaje.

Entorno de ejecución

La característica más importante del entorno de ejecución es la interfaz gráfica que proporciona, desde la que se puede controlar y depurar los agentes existentes. Todas las herramientas han sido desarrolladas como agentes y siguen sus mismas reglas.

Interfaz RMA

El agente RMA (Remote Monitoring Agent) permite controlar al resto de agentes en una plataforma. Provee una interfaz gráfica (figura 12) que facilita las funciones de monitorización y control. Solo puede existir un agente RMA por container, aunque puede haber varios por plataforma.

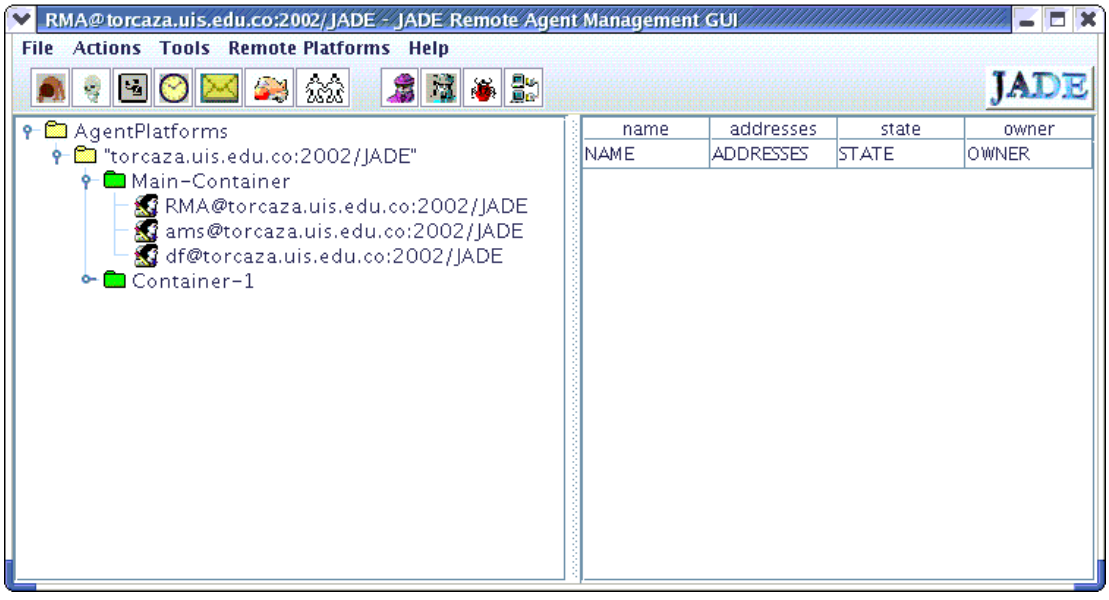


Figura 12. Interfaz gráfica de JADE

La interfaz gráfica permite las siguientes acciones, todas ellas llevadas a cabo a través de un sistema de menús:

- Terminar la ejecución del agente RMA (se invoca al método *doDelete()*).
- Terminar con la ejecución del agente RMA y de todos los agentes del container donde se encuentre RMA.
- Terminar con la ejecución de la plataforma en la que se encuentra.
- Comenzar un nuevo agente.
- Terminar con la ejecución de agentes. La terminación se realiza internamente llamando al método *doDelete()*.
- Detener la ejecución de los agentes. Internamente se realiza una llamada al método *doSuspend*.
- Continuar la ejecución de agentes suspendidos. Los pone en estado Activo y llama al método *doActivate()*.
- Mandar un mensaje (formato ACL) a agentes seleccionados. Esta función se realiza mediante un cuadro de diálogo en el cual se tendrán que rellenar los campos del mensaje.
- Migrar un agente de un contenedor a otro.
- Clonar un agente, introduciendo el nombre del nuevo agente y el container donde se encontrará.

Motivos de selección de JADE

Las razones por las cuales se puede seleccionar la herramienta JADE para el desarrollo de sistemas multiagente son las siguientes:

- JADE simplifica la comunicación y la cooperación entre los agentes.
- Los agentes JADE pueden controlar su propio ciclo de vida, y pueden ser programados para que dejen de funcionar o empiecen a hacerlo dependiendo del estado del sistema y de la función que debe realizar el agente.

- JADE cumple con la especificación de FIPA, luego puede comunicarse con agentes realizados en otros entornos que sigan FIPA.
- Es código abierto. Multitud de personas colaboran en la realización y mantenimiento de JADE.
- Los agentes JADE pueden correr en las diferentes versiones de Java: J2EE, J2SE y J2ME.
- El API proporcionado por JADE es intuitivo, fácil de aprender y sencillo de usar, haciendo que el desarrollo se produzca de manera más rápida.

4.3. METODOLOGÍAS PARA EL DESARROLLO DE SISTEMAS MULTIAGENTE

Una metodología es el conjunto de métodos empleados para el desarrollo de sistemas automatizados. Las metodologías proporcionan: guías para estimar costos, manejo del proyecto en las tareas y entregas, medidas y métricas, formas definidas y dirección en las entregas de la construcción, políticas y procedimientos para garantizar la calidad del software, descripciones de los roles y programas de entrenamiento detallados, ejemplos totalmente trabajados, ejercicios de entrenamiento, técnicas para adaptar el método, y técnicas definidas.[24] A continuación se presentan algunas metodologías orientadas al desarrollo de agentes inteligentes:

4.3.1. BDI

Las arquitecturas BDI se inspiran en un modelo cognitivo del ser humano [2]. Los agentes utilizan un modelo del mundo, una representación de cómo se les muestra el entorno. El agente recibe estímulos a través de sensores ubicados en el mundo. Estos estímulos modifican el modelo del mundo que tiene el agente (representado por un conjunto de creencias). Para guiar sus acciones, el agente tiene *Deseos*. Un *deseo* es un estado que el agente quiere alcanzar a través de *intenciones*. Éstas son acciones especiales que pueden abortarse debido a cambios en el modelo del mundo. Aunque la formulación

[24] www.monografias.com

inicial es de Bratman, fueron Georgeff, Rao y Kinny [22] quienes formalizaron este modelo y le dieron visos de metodología.

En esta metodología se trabaja a dos niveles de abstracción:

Un punto de vista externo:

- El sistema es modelado como una jerarquía de clases de agente, los agentes individuales son instancias.
- Las clases de agentes están caracterizadas por su propósito, sus responsabilidades, los servicios que desarrollan, la información acerca del mundo que requieran y las interacciones externas.

El punto de vista externo puede ser capturado en dos modelos:

- ***Modelo de agente:***
  - Describe la relación jerárquica entre diferentes clases abstractas y concretas de agente.
  - Permite también identificar las instancias de agente que deben existir en el sistema.
- ***Modelo de interacción:***
  - Describe las responsabilidades de una clase de agente, los servicios que provee, interacciones asociadas y relaciones de control entre clases de agente.
  - Incluye la descripción de los mensajes para la comunicación entre agentes y entre un agente y otros componentes del sistema.

Desde el punto de vista interno:

- Conjunto de modelos los cuales permiten estructurar el estado de motivación y de información de los agentes y las estructuras de control que determinan sus conductas.

En el punto de vista interno cada clase de agente se especifica en tres modelos:

- ***Modelo de creencias:*** describe la información acerca del entorno y el estado interno que un agente de una clase puede tener y las acciones que puede realizar.
- ***Modelo de objetivos:*** describe los objetivos que un agente puede adoptar, y los eventos a los que debe responder.

- ***Modelo de planes:*** describe los planes que un agente puede emplear para alcanzar sus objetivos o responder a eventos que percibe.

4.3.2. GAIA

GAIA es una metodología que se centra en la idea de que la construcción de sistemas basados en agente es un proceso de diseño organizacional. GAIA pretende ayudar al analista a ir sistemáticamente desde unos requisitos iniciales a un diseño que esté lo suficientemente detallado como para ser implementado directamente.

En GAIA se entiende que el objetivo del análisis es conseguir comprender el sistema y su estructura sin referenciar ningún aspecto de implementación. Esto se consigue a través de la idea de organización. Una organización en GAIA es una colección de roles, los cuales mantienen ciertas relaciones con otros y toman parte en patrones institucionalizados de interacción con otros roles. Los roles agrupan cuatro aspectos: responsabilidades del agente, los recursos que se le permite utilizar, las tareas asociadas e interacciones.

GAIA propone trabajar inicialmente con un análisis a alto nivel. En este análisis se usan dos modelos:

- **El modelo de roles:** para identificar los roles clave en el sistema junto con sus propiedades definitorias y
- **El modelo de interacciones:** que define las interacciones mediante una referencia a un modelo institucionalizado de intercambio de mensajes, como el FIPA-Request [11].

Tras esta etapa, se entraría en lo que GAIA considera diseño a alto nivel. El objetivo de este diseño es generar tres modelos:

- **El modelo de agentes:** que define los tipos de agente que existen, cuántas instancias de cada tipo y qué papeles juega cada agente,
- **El modelo de servicios:** que identifica los servicios (funciones del agente) asociados a cada rol,

- **El modelo de conocidos:** que define los enlaces de comunicaciones que existen entre los agentes.

4.3.3. MaSE

MaSE (Multi-*agent systems Software Engineering*) [9] se concibe como una abstracción del paradigma orientado a objetos donde los agentes son especializaciones de objetos. En lugar de simples objetos, con métodos que pueden invocarse desde otros objetos, los agentes se coordinan unos con otros vía conversaciones y actúan proactivamente para alcanzar metas individuales y del sistema.

En MaSE los agentes son simplemente una abstracción conveniente, que puede o no poseer inteligencia. En este sentido, los componentes inteligentes y no inteligentes se gestionan igualmente dentro del mismo armazón. Dado el enfoque inicial, los agentes se ven como especializaciones de objetos. De hecho, el sistema se construye sobre tecnología orientada a objetos y su aplicación a la especificación y diseño de sistemas multiagente.

El análisis en MaSE consta de tres pasos: capturar los objetivos, aplicar los casos de uso y refinar roles. El diseño consta de cuatro pasos: crear clases de agentes, construir conversaciones, ensamblar clases de agentes y diseño del sistema. La mayoría de estos pasos se ejecutan dentro de la herramienta que soporta MaSE, *AgentTool* [9].
Como productos de estas etapas, MaSE espera:

- Diagramas de secuencia para especificar interacciones,
- Diagramas de estados para representar procesos internos a las tareas y
- Modelar interacciones, descomposición del sistema (agente) en subsistemas (componentes del agente) e interconexión de los mismos (definición de la arquitectura del agente).

4.3.4. Ingenias

Ingenias ha sido desarrollada a partir de los resultados obtenidos en MESSAGE[25] por el grupo GRASIA[26]. Esta metodología propone un lenguaje de especificación de sistemas multiagente así como su integración en el ciclo de vida. El lenguaje se especifica con meta-modelos y lenguaje natural.

La integración en el ciclo de vida se consigue definiendo un conjunto de entregas y actividades involucradas en el desarrollo.

El método de desarrollo propuesto en Ingenias concibe el SMA como la representación computacional de un conjunto de modelos. Cada uno de estos modelos muestra una visión parcial del SMA: los agentes que lo componen, las interacciones que existen entre ellos, cómo se organizan para proporcionar la funcionalidad del sistema, qué información es relevante en el dominio y cómo es el entorno en el que se ubica el sistema a desarrollar.

Para especificar cómo tienen que ser estos modelos se definen meta-modelos. Un meta-modelo es una representación de los tipos de entidades que pueden existir en un modelo, sus relaciones y restricciones de aplicación.

Ingenias está conformado por cinco meta-modelos que giran alrededor de dos entidades la organización y el agente (Ver figura 13)[27]:

- **Meta-modelo de agente:** describe agentes particulares y los estados mentales en que se encontrarán a lo largo de su vida.
- **Meta-modelo de tareas y objetivos:** se usa para asociar el estado mental del agente con las tareas que ejecuta.

[25] Methodology for Engineering Systems of Software AGEnts: fue un proyecto que tuvo como objetivo extender las metodologías del software orientado a objetos para la realización de aplicaciones orientadas a agentes. Como resultado del proyecto se elaboró una metodología para desarrollo de sistemas multiagente. Las entidades participantes fueron: France Telecom, Telefónica I+D, British Telecom, Portugal Telecom, Telecom Italia, Belgacom y Broadcom Eireann Research.
[26] GRupo de Agentes de Software: Ingeniería y Aplicaciones de la Universidad Complutense Madrid.
[27] http://grasia.fdi.ucm.es

- **Meta-modelo de organización:** define cómo se agrupan los agentes, la funcionalidad del sistema y qué restricciones hay que imponer sobre el comportamiento de los agentes.
- **Meta-modelo de interacción:** detalla cómo se coordinan y comunican los agentes.
- **Meta-modelo de entorno:** define qué existe alrededor del nuevo sistema y cómo lo percibe cada agente.

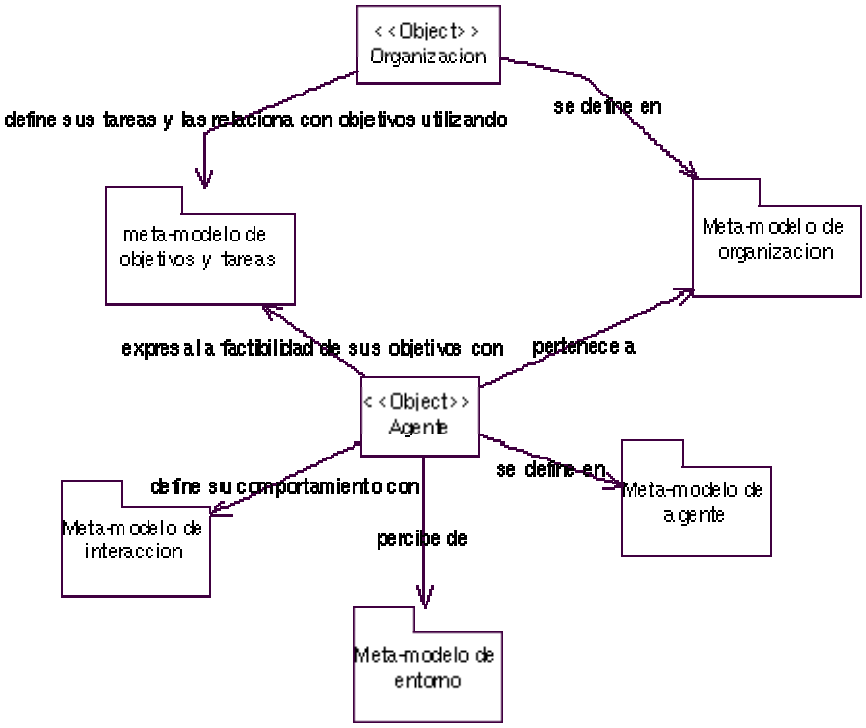


Figura 13. Relación entre los diferentes meta-modelos y las dos entidades principales, la organización y el agente (tomada de [40])

4.3.5. AgentUML

AgentUML (*Agent Unified Modeling Language*) es un conjunto de artefactos[28] de UML para su utilización en el proceso de desarrollo de los SMA, que une lo desarrollado en

[28] Un artefacto es una información que es utilizada o producida mediante un proceso de desarrollo de software. Pueden ser artefactos un modelo, una descripción o un software.

materia de metodologías de desarrollo de software de agentes con los estándares definidos para el desarrollo de software Orientados a Objetos.

AgentUML sugiere una representación en tres capas, denominada Agent Interaction Protocol (AIP) [4]:

- **Capa 1:** Representa los protocolos de comunicación y se utilizan los Paquetes y las Plantillas (*Templates*) de UML para la especificación de estos protocolos.
- **Capa 2:** Muestra la interacción de los agentes usando Diagramas de Interacción UML.
- **Capa 3:** Muestra el procesamiento interno de los agentes por medio de los Diagramas de Actividad y de Estados.

Inicialmente se identifican dos áreas para el desarrollo detallado de especificaciones [13]:

- **Diagramas de clases:** Especifican el comportamiento interno de un agente y su relación con el exterior usando diagramas de clases UML adaptados.
- **Diagramas de interacciones o protocolo:** Término genérico que se aplica a diversos tipos de diagramas centrados en la interacción entre agentes Similar a los diagramas de interacción usados en UML. Existen varios tipos de diagramas:
  - ***Diagramas de secuencia:*** muestran las sucesiones de tiempo entre agentes ordenados de forma secuencial. Los diagramas de secuencia tienen dos dimensiones: la dimensión vertical representa el orden en el tiempo y la dimensión horizontal representa los diferentes roles o los agentes que poseen roles específicos (ver la figura 14).

    AgentUML permite representar varios agentes en una misma línea de vida, ya que no necesariamente representa a un agente sino que representa el rol[29] que desempeña uno o varios agentes. Un agente puede: asumir múltiples roles o cambiar los roles. Múltiples roles implican que los agentes participan varias veces en la misma interacción.

[29] Rol: nombre de comportamientos poseídos por una clase o por parte de un participante en un contexto determinado.

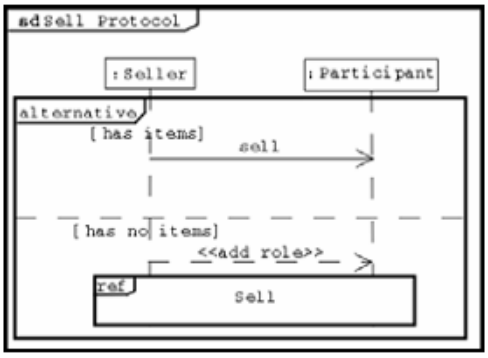


Figura 14. Ejemplo de un diagrama de secuencia en AgentUML (tomada [41])

- ***Diagramas de descripción de interacciones:*** Son una variante de los Diagramas de Actividades de UML. Están enfocados a la descripción del flujo de control donde los nodos son interacciones, sin mostrar los mensajes.

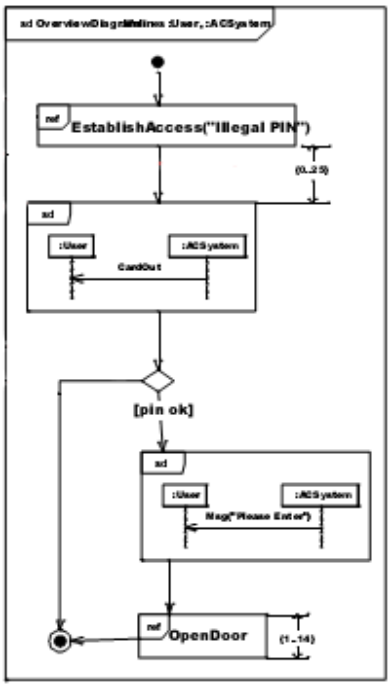


Figura 15. Ejemplo de un diagrama de interacción en AgentUML (tomada [41])

- ***Diagramas de colaboración o diagrama de comunicación:*** Enfocan las interacciones entre las líneas de vida. El encadenamiento de los mensajes se da a través de un esquema secuencial numerado.

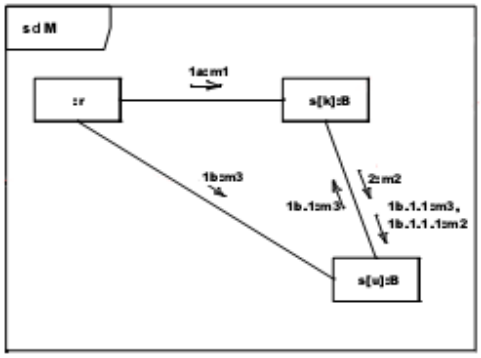


Figura 16. Ejemplo de un diagrama de colaboración en AgentUML (tomada [41])

- ***Diagramas de estado:*** Muestran cambios en los estados u otra condición de un elemento estructural. El propósito es mostrar los cambios en los estados durante el tiempo.

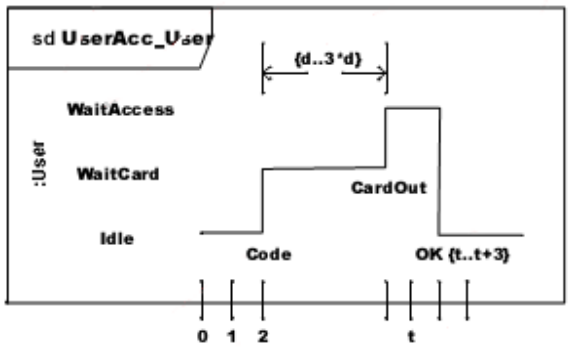


Figura 17. Ejemplo de un diagrama de estado en AgentUML (tomada [41])

5. DESARROLLO DEL AGENTE GENERADOR DE EJERCICIOS INTERACTIVOS PARA EL e-ESCEN@RI$_{UIS}$

Este capítulo tratará sobre las funcionalidades de la plataforma educativa institucional e-ESCEN@RI$_{UIS}$ de la Universidad Industrial de Santander, desde el punto de vista de la arquitectura, las herramientas que ofrece a los usuarios y el sistema multiagente con el cual se pretende mejorar la enseñanza y motivar a los estudiantes a aprender según sus preferencias en un entorno amigable y lo más cercano posible a su estilo de aprendizaje. Así mismo, se hablará sobre el desarrollo del agente generador de ejercicios, el cual hace parte del sistema multiagente de la plataforma educativa institucional.

5.1. PLATAFORMA EDUCATIVA INSTITUCIONAL

La plataforma educativa e-ESCEN@RI$_{UIS}$ de la Universidad Industrial de Santander, hace parte del *Proyecto Soporte al Proceso Educativo UIS mediante Tecnologías de Información y Comunicación – ProSPETIC* adscrito a la Vicerrectoría Académica, el cual "define la política de uso de las TICs en los procesos educativos institucionales y las estrategias encaminadas a lograr el desarrollo sistemático y planificado de experiencias educativas mediadas por las TICs, como soporte a los programas académicos de la Universidad. El propósito es fortalecer las experiencias de educación en línea existentes, llevar la oferta de formación a nuevos ámbitos geográficos, flexibilizar los procesos de enseñanza y aprendizaje, promocionar la innovación educativa y agregar valor a los procesos de investigación, transferencia tecnológica y gestión e integración de la Universidad con la sociedad."[30]

[30] Resumen ejecutivo del Proyecto "Soporte al proceso educativo UIS mediante Tecnologías de Información y Comunicación – ProSPETIC" de la Universidad Industrial de Santander.

Figura 18. Interfaz gráfica del estudiante de la plataforma educativa e-ESCEN@RI$_{UIS}$

El e-ESCEN@RI$_{UIS}$ es un Sistema Tutorial Inteligente (STI) implementado con tecnología Web (Java, javascript, HTML, XML, XHTML, FLASH, actionScript, etc.) que a través de un sistema multiagente busca modelar al estudiante con el fin de ofrecer los contenidos didácticos, las herramientas de navegación y las estrategias pedagógicas según las características del estilo de aprendizaje y del nivel de conocimiento del estudiante. La plataforma proporciona un conjunto de herramientas para permitir a los profesores crear y editar materiales, transferir, organizar y gestionar los archivos de estos materiales, generar y gestionar diferentes tipos de ejercicios interactivos y crear y gestionar los contenidos de las unidades docentes. El desarrollo de esta plataforma educativa institucional está basado en el prototipo del PLAN G[31] de la Universitat de Girona en España que está plasmado en la tesis doctoral de la Dra. Clara Inés Peña de Carrillo[32].

[31] PLAN G: PLAtaforma telemática de Nueva Generación para el soporte de enseñanza abierta y a distancia. Es un proyecto de investigación soportado por el Ministerio de Educación y Ciencia de España.
[32] Peña, C.I., Intelligent Agents to Improve Adaptivity in a Web-Based Learning Environment, PhD Thesis, University of Girona, Spain, 2004.

El e-ESCEN@RI$_{UIS}$ consta de un conjunto de *unidades docentes*, las cuales son un conjunto de páginas HTML, que incluyen el material educativo a utilizar para el soporte a la enseñanza de determinadas asignaturas y, de una *estructura de navegación* predefinida para facilitar el acceso. Al estar separadas la estructura y el contenido se facilita la reutilización de una misma página en diferentes unidades docentes. Así mismo, al contar con una sola estructura para el acceso al contenido, una misma unidad docente se puede presentar en diferentes idiomas. En la figura 19 se pueden observar los nodos (A-1 a C-2) y las estructuras de navegación (e-1, e-2 o e-3).

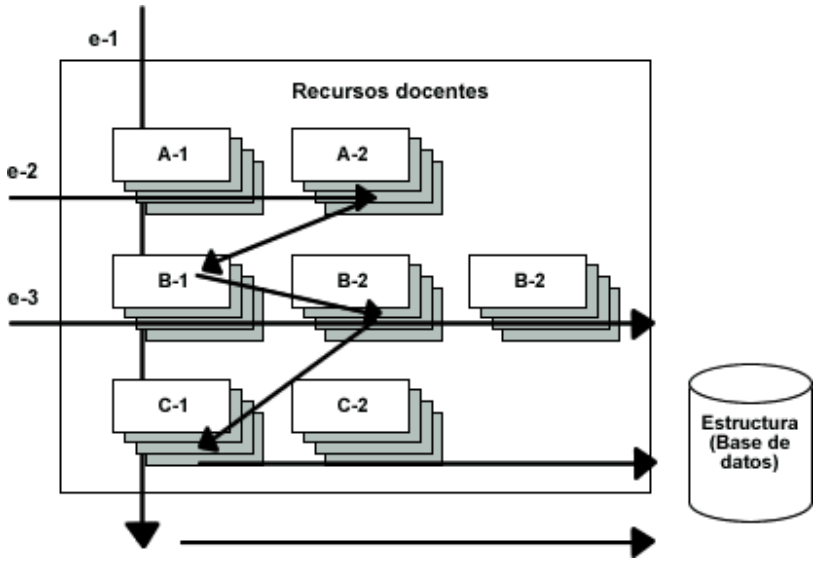


Figura 19. Representación de la estructura de navegación y contenidos de aprendizaje de e-ESCEN@RI$_{UIS}$ (tomado de [30])

Para la determinación de los estilos de aprendizaje se ha adoptado el Modelo FSLSM (*Felder and Silverman Learning Style Model*) que permite categorizar a los estudiantes de acuerdo a su habilidad para procesar, percibir, recibir, organizar y entender la información. Para ello se aplica a los estudiantes el instrumento de diagnóstico del modelo FSLSM denominado ILS (*Index for Learning Styles*), el cual ha sido ampliamente probado en entornos educativos en línea.

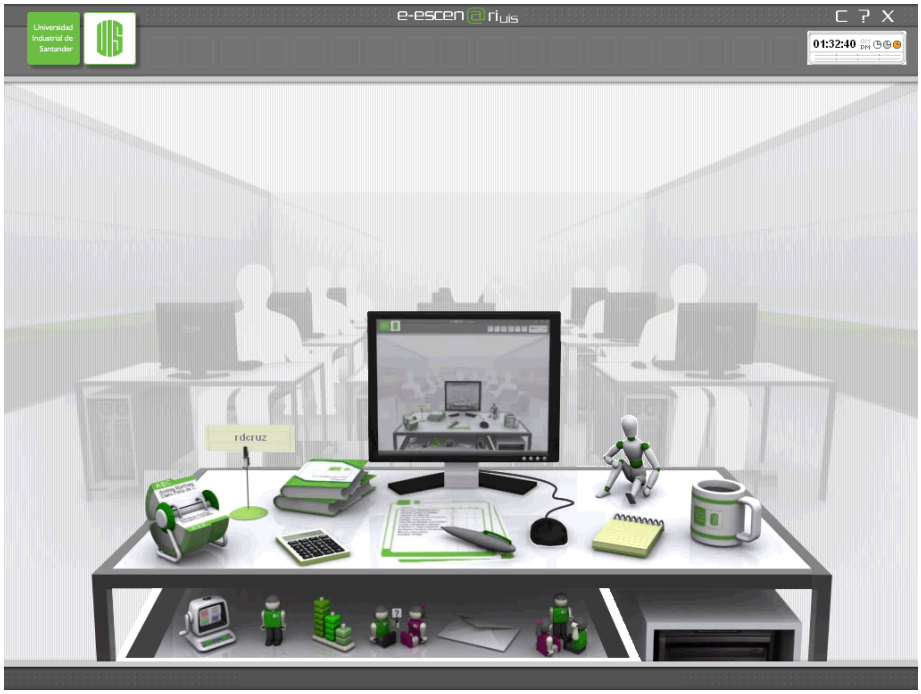


Figura 20. Interfaz gráfica del profesor de la plataforma educativa e-ESCEN@RI$_{UIS}$

La interfaz gráfica de e-ESCEN@RI$_{UIS}$ está diseñada de forma amigable e intuitiva; representa un salón de clase, con las herramientas propias que se pueden encontrar en él. Cada una de estas herramientas representa un ícono que ayuda al usuario a interactuar con las funcionalidades que ofrece el sistema. En las figuras 18 y 20 se observan las interfaces gráficas del estudiante y profesor, respectivamente. La diferencia entre ambas, además de las funcionalidades de cada una de las herramientas de la plataforma, es la perspectiva del salón de clase que varía de acuerdo al perfil del usuario.

5.1.1. Usuarios del e-ESCEN@RI$_{UIS}$

La plataforma e-ESCEN@RI$_{UIS}$ cuenta con perfiles de usuarios para establecer las actividades que pueden realizar cada uno de los actores del sistema. Así mismo, el ingreso se hace a través de la identificación, autenticación y autorización de los usuarios, es decir, el usuario se identifica con un nombre de inicio de sesión y contraseña, se autentican estos datos y se procede a autorizar el acceso a las herramientas a las que

tenga permiso. Un usuario puede tener más de un perfil, dependiendo de los roles que desempeñe dentro de la comunidad universitaria[33].

A continuación se describen los perfiles de usuarios usados en e-ESCEN@RI_{UIS}:

- **Invitado:** Usuario que no hace parte de la comunidad universitaria, accede con una cuenta pública, por lo tanto no tiene permisos sobre todas las herramientas de la plataforma. Solo puede interactuar con aquellos contenidos que han sido compartidos a todos los usuarios por sus respectivos autores.

- **Estudiante:** Actor principal en el proceso de aprendizaje. Es el responsable de desarrollar ciertas habilidades especiales que le permitan sacar el máximo provecho de las estrategias pedagógicas definidas por su profesor. También se encarga de responder las pruebas psicosociales aplicadas por la División de Bienestar Universitario (DBU).

- **Profesor:** Actor principal en el proceso de enseñanza. Es el responsable de la definición de objetivos, preparación de los contenidos, selección de una metodología apropiada, elaboración de material didáctico y elaboración de un plan de evaluación.

- **Administrador:** Es el gestor y auditor de la plataforma educativa.

- **Profesional DBU:** Profesional de la Sección de Salud y Desarrollo Psicosocial de la División de Bienestar Universitario (DBU) que interactúan con las pruebas psicosociales que se les aplican a los estudiantes.

- **Administrador DBU:** Profesional de la Sección de Salud y Desarrollo Psicosocial de la DBU que gestiona las pruebas psicosociales que se les aplican a los estudiantes.

[33] Comunidad universitaria son los miembros que hacen parte de la Universidad Industrial de Santander, tales como: profesores, estudiantes, personal administrativo, personal de outsourcing, personal de prestación de servicios, etc.

- **Tutor:** Estudiante que brinda asesoría académica a través del programa PAMRA (Programa de Asesoría para el Mejoramiento del Rendimiento Académico) de la DBU. Este tipo de usuario hace las veces de profesor a estudiantes que requieren asesoría académica en las diferentes asignaturas que ofrece la UIS.

- **Auxiliar:** Estudiante que le colabora al profesor en sus funciones académicas.

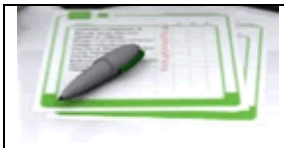
5.1.2. Herramientas del e-ESCEN@RI$_{UIS}$

La plataforma cuenta con una serie de herramientas que les permiten a los usuarios interactuar con el sistema para realizar sus actividades. Algunas de estas herramientas se encuentran todavía en desarrollo por parte de los miembros del Laboratorio de Investigación y Desarrollo del CENTIC[34]. A continuación se describen las funcionalidades que ofrecen las herramientas de e-ESCEN@RI$_{UIS}$ a los diferentes usuarios:

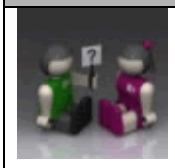
Tabla 2. Herramientas de la plataforma educativa e-ESCEN@RI$_{UIS}$

 **GESTOR DE CONTENIDOS**	
Profesor y Tutor	Permite crear y mantener la estructura de navegación de los contenidos temáticos del curso de acuerdo al estilo de aprendizaje y nivel de conocimiento del estudiante.
Estudiante	Permite navegar sobre los contenidos de aprendizaje de forma libre o guiada, con flechas de avanzar y retroceder o a través del mapa del curso.
Invitado	Tiene acceso a los contenidos que han sido compartidos por sus autores.

[34] CENTIC: CENtro de Tecnologías de Investigación y Comunicación de la Universidad Industrial de Santander.

 **GESTOR DE EVALUACIÓN**	
Profesor y Tutor	Permite crear y mantener ejercicios de evaluación y/o entrenamiento de las temáticas tratadas en la asignatura. Los ejercicios están categorizados por: temáticas, tipo de ejercicio (asociación, selección, cuestionario, pregunta abierta, completar, sopa de letras y ordenar), nivel de dificultad (fácil, normal y difícil) y competencias (interpretativa, propositiva y argumentativa). Esta herramienta permite la combinación de los diferentes tipos de ejercicio, creando así, ejercicios mixtos.
Estudiante	Permite la resolución de los ejercicios propuestos por el profesor, la creación de ejercicios de autoevaluación (configurables y adaptados de acuerdo al nivel de conocimiento) y la resolución de pruebas psicosociales.
Invitado	Tiene acceso a los ejercicios que han sido compartidos por sus autores.
Profesional DBU	Permite consultar los cuestionarios de las pruebas psicosociales y el resultado de las mismas.
Administrador DBU	Permite la creación y mantenimiento de pruebas psicosociales, así como la consulta sobre las pruebas contestadas por los estudiantes.
 **BIBLIOGRAFÍA**	
Profesor y Tutor	Permite poner a disposición del estudiante la bibliografía para las temáticas tratadas en la asignatura.
Estudiante	Consulta de la bibliografía propuesta por el profesor para las temáticas estudiadas durante sus sesiones de aprendizaje.
Invitado	Consulta de la bibliografía de los contenidos a los que está autorizado navegar.

 **ASISTENTE PERSONAL**	
Profesor y Tutor	Permite automatizar algunas tareas de soporte, permitiendo al profesor programar sus actividades, tales como: llamar la atención de los estudiantes conectados a la plataforma en un momento dado para recibir indicaciones o explicaciones en línea.
Estudiante	Permite automatizar algunas actividades, tales como: anunciar la conexión a la plataforma de un compañero, sugerir la revisión de la bibliografía, programar mensajes personalizados durante la sesión, etc.
 **CONFIGURACIÓN DE LA PANTALLA**	
Profesor, Estudiante, Administrador, Profesional DBU, Administrador DBU, Tutor y Auxiliar	Configuración del entorno de trabajo en cuanto a los colores del escritorio e idioma.
 **PERFIL DEL USUARIO**	
Profesor, Estudiante, Administrador, Profesional DBU, Administrador DBU, Tutor y Auxiliar	Actualización de la información personal del usuario, cargue de la fotografía del usuario y cambio de la contraseña para acceder a la plataforma. Al estudiante le permite realizar consulta sobre su información académica.

 **ESTADÍSTICAS**	
Profesor, Tutor y Estudiante	Consulta de las estadísticas de las actividades realizadas por los estudiantes en la plataforma. Tales como: sesiones, ejercicios, resultados del Test de Felder, etc.
Profesional DBU y Administrador DBU	Consulta de las estadísticas generadas por las pruebas psicosociales aplicadas a los estudiantes.
Administrador	Consulta de las estadísticas generales del sistema.
 **CHAT**	
Profesor y Tutor	Permite la comunicación en línea con los estudiantes para brindar asesoría en las temáticas tratadas en la asignatura.
Estudiante	Permite la comunicación en línea con el profesor y compañeros de estudio para recibir y brindar asesoría en las temáticas tratadas en la asignatura.
Profesional DBU y Administrador DBU	Permite la comunicación en línea con los pacientes para brindar asesoría psicosocial a aquellos que necesiten alguna intervención para lograr una buena salud mental.
 **CORREO**	
Profesor y Tutor	Servicio de mensajería electrónica para mantener una comunicación con los estudiantes. A través de listas de distribución se pueden enviar y recibir archivos complementarios sobre las temáticas de la asignatura.
Estudiante	Servicio de mensajería electrónica para mantener una comunicación con el profesor y los compañeros de estudio. A través de listas de distribución se pueden recibir y enviar archivos complementarios sobre las temáticas de la asignatura.

Profesional DBU y Administrador DBU	Servicio de mensajería electrónica para mantener una comunicación con los pacientes. A través de listas de distribución se pueden enviar archivos complementarios a los pacientes sobre los temas tratados en las sesiones.
 **FORO**	
Profesor y Tutor	Permite crear temas de discusión sobre las diferentes temáticas tratadas en la asignatura, consulta de estadísticas de las actividades realizadas durante los foros, asignación de moderadores de los foros y programación de tareas.
Estudiante	Participación en los foros de discusión propuestos por el profesor y resolución de las tareas.
Auxiliar	Consulta de las estadísticas y moderación de los temas.
Profesional DBU y Administrador DBU	Creación de temas de discusión sobre las diferentes temáticas tratadas en las sesiones psicológicas, consulta de estadísticas de las actividades realizadas durante los foros, asignación de moderadores de los foros y programación de tareas.
 **LIBRETA DE NOTAS**	
Profesor, Estudiante, Administrador, Profesional DBU, Administrador DBU, Tutor y Auxiliar	Toma de apuntes a través de un editor HTML.
 **CALCULADORA**	
Profesor, Estudiante, Administrador, Tutor y Auxiliar	Calculadora para realizar operaciones básicas de matemáticas.

 **DESCANSO**	
Estudiante	Pausa durante la sesión de estudio.

5.1.3. Arquitectura conceptual

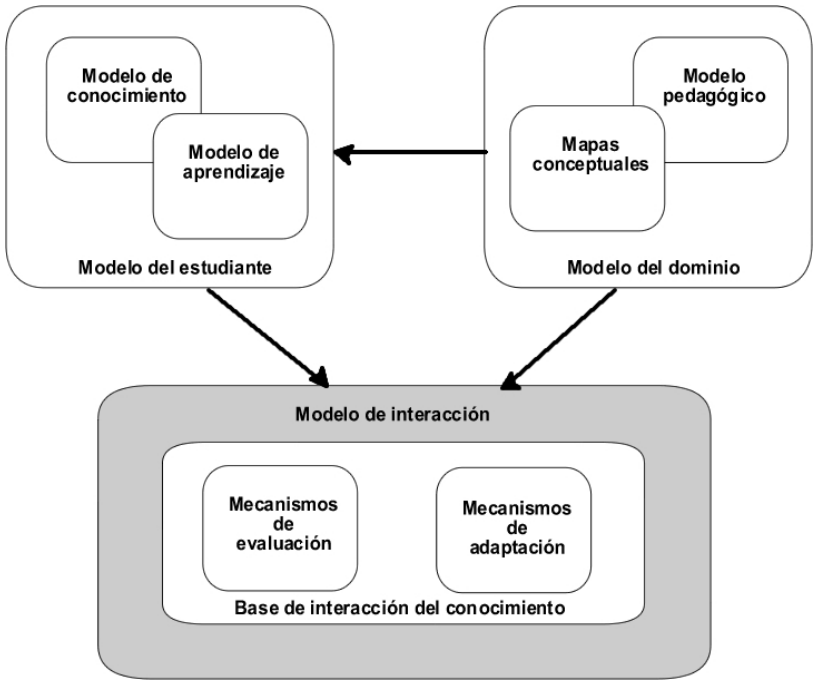


Figura 21. Arquitectura conceptual de e-ESCEN@RI$_{UIS}$ (tomada de [30])

La arquitectura conceptual mostrada en la figura 21, está conformada por tres modelos principales: el *modelo del dominio*, el *modelo del estudiante* y el *modelo de interacción*. El *modelo de dominio* contiene la información que el estudiante recibirá y una serie de reglas que deciden cómo transmitir la información (*modelo pedagógico*). El *modelo del*

estudiante registra la información sobre el conocimiento y habilidades adquiridas por el estudiante (*Modelo de conocimiento* y *modelo de aprendizaje*). El *modelo de interacción* supervisa las interacciones del estudiante y ofrece mecanismos adaptativos para dar al estudiante la información acorde a sus necesidades.

- **Modelo del dominio:** representa tanto el conocimiento sobre un dominio particular que será transmitido al estudiante, como la forma de presentar esa información (reglas definidas en un modelo pedagógico). El conocimiento del modelo de dominio y su estructura determina el contenido de la interacción tutorial, junto con la estructura que gobierna la instrucción adaptativa.

  El modelo del dominio es declarativo y su conocimiento es representado por medio de un *mapa conceptual* o grafo semántico cuya estructura toma en cuenta los enfoques *estático* (el *qué*) y *evolucionario* (el *cómo*).

  Desde el punto de vista *estático*, los conceptos de enseñanza son representados por medio de una red conceptual estructurada usando diferentes taxonomías. Cada nodo corresponde a un concepto del dominio y puede estar desagregado en otros nodos utilizando relaciones clase-subclase (Ej: estructura de un árbol). La red conceptual resultante es una representación estática del conocimiento en el dominio de la enseñanza (Ej: *qué* será enseñado).

  Desde el punto de vista *evolucionario*, la red conceptual está estructurada usando relaciones para describir las reglas pedagógicas necesarias para seleccionar los contenidos y/o determinar su secuencia. En este estudio, relaciones conceptuales (Ej. Relaciones de *propiedad* como “X” es parte de “Y”) y relaciones de procedimiento son consideradas. Las relaciones de procedimiento son usadas para determinar el *orden* en el cual el nodo conceptual debe ser enseñado o las *decisiones* que deben ser evaluadas para alcanzar cualquier objetivo instruccional (Ej. Si la condición A es verdadera, entonces el estudiante puede estudiar los nodos 1.1 y 1.2 del *Concepto 1*). Esta estructura corresponde a la organización didáctica del dominio (Ej: *Cómo* los conceptos serán enseñados).

- **Modelo del estudiante:** para modelar al estudiante se tienen en cuenta dos elementos: la base del conocimiento del modelo del estudiante y el *agente de usuario*.

  Las características de aprendizaje del estudiante establecidas en la base de conocimiento siguen un modelo híbrido, una combinación de un modelo *overlay* [6] y un modelo *deducido*, que representa el conocimiento del estudiante sobre el dominio. Este modelo es a su vez dividido en dos modelos conceptuales más: uno *permanente* y otro *temporal*.

  El modelo permanente contiene información concerniente a los datos sobre las características personales del estudiante y su perfil de aprendizaje, el conocimiento que tiene sobre el dominio: el material didáctico que él ha utilizado para aprender y la memoria de las sesiones de aprendizaje que han tenido (acciones comunes, memoria de ejercicios, etc.). Este modelo está disponible durante todo el proceso de instrucción y se actualiza sesión por sesión. El *conocimiento sobre el dominio*, es el conocimiento que el estudiante ha adquirido a través del proceso de aprendizaje. La estructura particular del dominio se modifica para incluir nuevos atributos que controlan dicha adquisición (la navegación a través del grafo se adapta al estado de conocimiento del estudiante). Estos atributos son: el *nivel de conocimiento* que el estudiante tiene sobre el concepto y los *conceptos que el estudiante ha aprendido*.

  El *material didáctico que el estudiante ha utilizado para aprender* (contenidos básicos y ejercicios) identifica el material usado por el sistema para presentar el contenido de aprendizaje o para evaluarlo. Esta información es utilizada por el agente pedagógico de la estructura multiagente para generar una opción apropiada de contenidos que el estudiante pueda aprender en un momento determinado. El agente *generador de ejercicios* utiliza la información concerniente al ejercicio que el estudiante ya resolvió para adaptar nuevos ejercicios a su estado de conocimiento.

  En el modelo del estudiante, la información sobre el desarrollo del proceso de instrucción es también considerada. Los datos sobre la última sesión sintetizan los eventos ocurridos en la última sesión de aprendizaje. Esta información es importante para fijar ciertos elementos a ser representados en la siguiente sesión.

La *memoria de todo el proceso instruccional* es representada por un conjunto de acciones comúnmente llevadas a cabo, como los nodos visitados junto con el tiempo gastado en cada visita y la información sobre el desempeño del estudiante cuando resuelve ejercicios (una lista de ejercicios que el estudiante ha hecho y la forma como los ha realizado). Esta información está disponible a través del botón de estadísticas de aprendizaje en la barra de herramientas general del ambiente de aprendizaje.

El *modelo temporal*, solo tiene sentido para la sesión actual. Este dato es manejado por el *agente de usuario*, que al finalizar la sesión, actualiza el *modelo permanente* con la información relevante que debe alterar el estado de conocimiento del estudiante.

5.1.4. Sistema multiagente del e-ESCEN@RI$_{UIS}$

El sistema multiagente de e-ESCEN@RI$_{UIS}$ está basado en [30], según la arquitectura presentada en la figura 22. Este sistema se ha construido bajo los estándares de FIPA mediante una arquitectura de dos niveles de agentes (nivel superior o asistentes personales y nivel inferior o agentes de información) teniendo en cuenta las siguientes propiedades:

- **Reactividad:** los agentes necesitan mantener una continua relación con su ambiente y responder a los cambios que suceden en él.
- **Interactividad:** los agentes necesitan interactuar entre ellos para lograr sus objetivos. El agente debe ser capaz de interactuar con su entorno cuando represente a un individuo o a una entidad. Durante esta fase también debe ser capaz de llevar a cabo diferentes tipos de comunicación con otros agentes de acuerdo a la entidad con la que interactúa.
- **Autonomía:** los agentes necesitan conocer cuándo y cómo llevar a cabo las tareas que les han sido encomendadas. Los agentes deben ser semiautónomos, es decir, que no necesitan una directa y constante supervisión. Esta característica es esencial para la representación de sus tareas, pero como existen diferentes grados de

autonomía, el agente debe siempre estar bajo del control de la entidad o persona que él representa.

- **Proactividad:** los agentes tienen metas y objetivos definidos y necesitan actuar de manera autónoma para lograrlos.
- **Aprendizaje:** los agentes inteligentes deben adquirir conocimiento de su representado y del entorno donde llevan a cabo sus funciones. Este conocimiento debe ser dinámico porque cambia con el tiempo. El agente debe ser capaz de aprender de su entorno y de la interacción con otros agentes e incorporar estos cambios en su base de conocimiento.

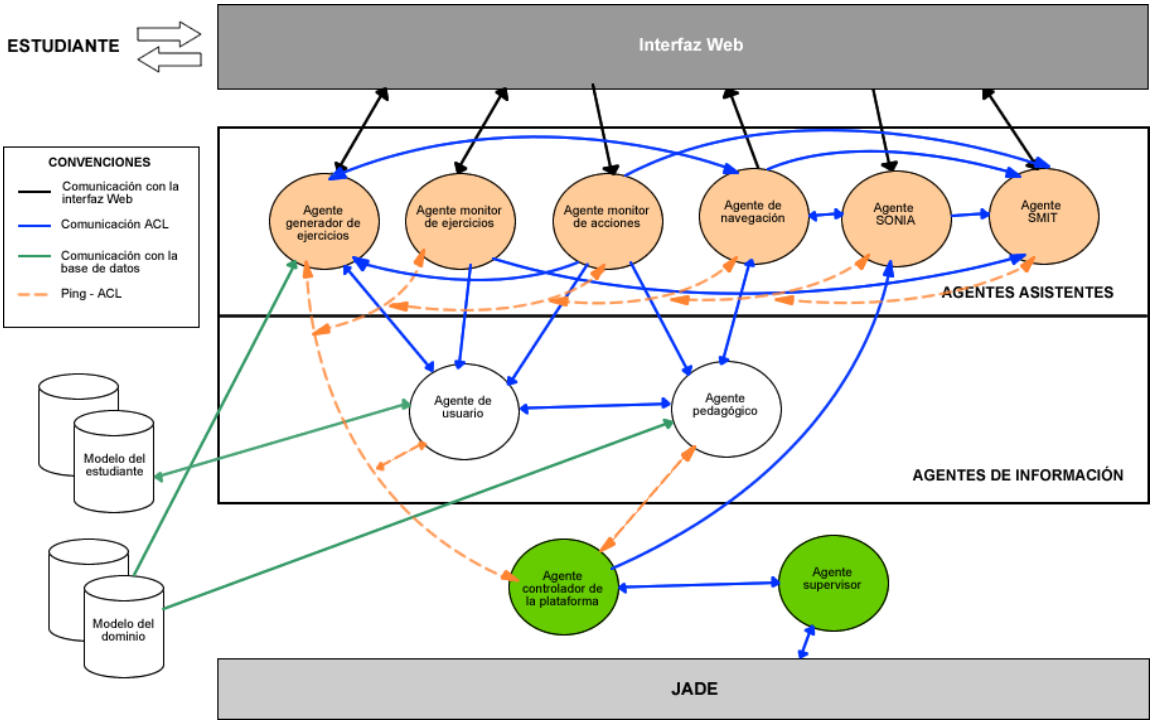


Figura 22. Arquitectura del sistema multiagente del e-ESCEN@RI$_{UIS}$ (tomado de [30])

Agentes de nivel superior o asistentes personales

Los agentes asistentes son los encargados de atender a los estudiantes mientras trabajan con el material didáctico de un curso. Estos agentes aprenden del entorno e interactúan con el estudiante para ayudarlo en el desarrollo de sus actividades de aprendizaje. Los

estudiantes son motivados a través de interfaces animadas y la adaptación de ejercicios de acuerdo al nivel de conocimiento o preferencias. A continuación se muestran los agentes que hacen parte de este nivel:

- **SONIA (Student Oriented Network Interface Agent):** es un agente de reflejo simple que recibe instrucciones del profesor o estudiante e información de eventos ocurridos en el entorno de aprendizaje. Es un agente programable que trata de automatizar tareas de aprendizaje, ya sea permitiendo al estudiante programar sus actividades basándose en ejemplos o imitando su comportamiento y adaptándose a él. En la figura 23 se puede observar la apariencia que tiene dentro de la plataforma educativa e-ESCEN@RI$_{UIS}$. Entre las tareas que se pueden programar se tienen las siguientes:
  - Anunciar cuando un determinado compañero de clase se conecte al sistema.
  - Sugerir la revisión de referencias bibliográficas en las secciones del curso que así lo requieran.
  - Sugerir el desarrollo de ejercicios interactivos propuestos para determinada sección del curso.
  - Informar al estudiante cuando lleve determinado tiempo de estudio.
  - Recordar al usuario mensajes personalizados en un tiempo determinado durante la sesión de aprendizaje.
  - El profesor puede programar un llamado de atención de los estudiantes conectados a la plataforma en un momento dado para que reciban indicaciones o explicaciones en línea.

  Para realizar estas tareas el agente SONIA trabaja de forma cooperativa con los agentes *controlador*, de *navegación* y *SMIT* de la siguiente manera:
  - Al agente de *navegación*, le pregunta por la existencia de referencias bibliográficas o ejercicios interactivos recomendados para una determinada lección.
  - Al agente *controlador*, le pregunta por el estado de ciertos eventos del sistema (como la hora, ingreso de un usuario específico o existencia de un mensaje de difusión).
  - Al agente *SMIT*, le envía el contenido de los mensajes que se deben representar al usuario indicando el cumplimiento de una tarea programada.

Figura 23. Aspecto del agente SONIA en la plataforma e-ESCEN@RI$_{UIS}$

- **SMIT (Synthetic Multimedia Interactive Tutor):** es un agente sintético. Se introduce en el entorno utilizando una interfaz animada de tipo antropomórfico para presentar al estudiante los mensajes que provienen de otros agentes del sistema (por ejemplo, el agente SONIA). A este agente se le puede configurar su apariencia física de acuerdo al gusto del usuario como se puede observar en el prototipo de la figura 24.

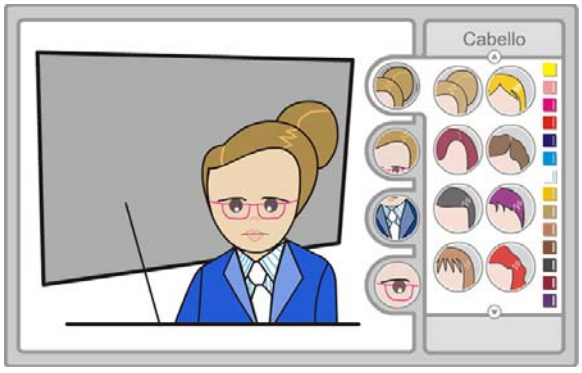


Figura 24. Prototipo de la interfaz para configurar la apariencia física del agente SMIT de la plataforma e-ESCEN@RI$_{UIS}$

La representación de cada mensaje requiere de la selección de ciertas animaciones y movimientos corporales que definen el comportamiento del agente en una situación particular como se puede observar en la figura 25. Con la utilización de este agente se pretende acompañar al estudiante en su proceso de aprendizaje a través de la representación "humanizada" del profesor.

Figura 25. Prototipo para la representación de los mensajes del agente SMIT en la plataforma e-ESCEN@RI$_{UIS}$

- **Agentes monitores:** El primer agente es el de *acciones*, el cual guarda las actividades realizadas por el estudiante en el entorno de aprendizaje, es decir, registra en la base de datos el ingreso a determinadas herramientas y a enlaces del sistema. El segundo agente es el de *ejercicios*, el cual supervisa las actividades del estudiante en el entorno de aprendizaje.

  La información recopilada por estos agentes permite generar información de retroalimentación al modelo de comportamiento del estudiante, para verificar o refinar su estilo de aprendizaje y evaluar su estado de conocimiento.

- **Agente generador de ejercicios:** Se encarga de la construcción de ejercicios adaptativos para el estudiante (ver figura 26). El agente puede escoger un nivel de dificultad conveniente para las preguntas de acuerdo al progreso del estudiante en el sistema. Este proceso se lleva a cabo teniendo en cuenta los siguientes aspectos:

- Las preferencias del estudiante, en cuyo caso es este quien configura los temas, categorías, competencias y tipos de preguntas que desea contestar (ejercicio configurado).
- El nivel de conocimiento del estudiante, en cuyo caso es el agente quien selecciona los temas, categorías, competencias y tipos de preguntas que el estudiante puede responder en un momento dado (ejercicio adaptado al nivel de conocimiento del estudiante).

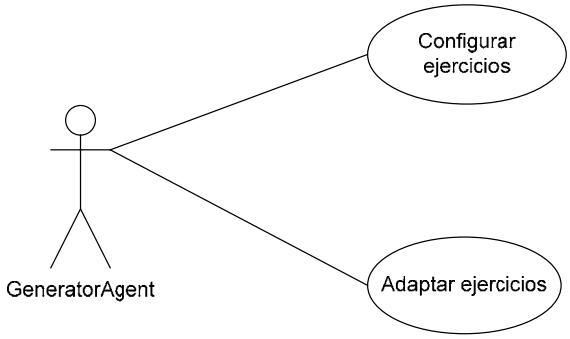


Figura 26. Diagrama de casos de usos para el agente generador de ejercicios (tomado de [30])

- **Agente de navegación:** es el que organiza mediante su interacción con el agente de *usuario* y el agente *pedagógico*, los caminos de navegación a seguir por el estudiante sobre los contenidos didácticos adaptados a su estilo de aprendizaje. Este agente se comunica con otros agentes del sistema de la siguiente forma:
  - Con el agente *pedagógico* para refrescar la información a presentar en el sistema. (El agente *pedagógico* construye y mantiene el árbol de navegación y el diagrama de estado de los conceptos de acuerdo al modelo del estudiante).
  - Con el agente *SONIA*, para indicar qué nodos tienen información particular para revisar (si el estudiante le ha programado dicha tarea).
  - Con el agente *generador de ejercicios*, si la lección tiene ejercicios asignados.
  - Con el agente *SMIT*, para enviar la información que será presentada al estudiante mediante interfaces amigables y atractivas (para motivar o para reforzar comportamientos).

Agentes de nivel inferior o de información

Son los encargados del mantenimiento de los modelos pedagógicos y del estudiante. Estos agentes residen en el servidor y están muy cerca de la base de datos del sistema, es decir, que actúan de intermediarios entre los agentes del nivel superior y la base de datos para recomendar los contenidos de las unidades docentes adaptadas a las preferencias del estudiante de acuerdo a su estilo de aprendizaje. A continuación se muestran los agentes que hacen parte de este nivel:

- **Agente de usuario:** Construye y mantiene el modelo del estudiante de acuerdo al análisis de las acciones del estudiante (recopiladas por los *agentes monitores*) y a su estado de conocimiento. Los agentes *pedagógico* y *generador de ejercicios* consultan el agente de *usuario* para obtener información acerca del modelo del estudiante, para adaptar los contenidos y los caminos de navegación (ver la figura 27).

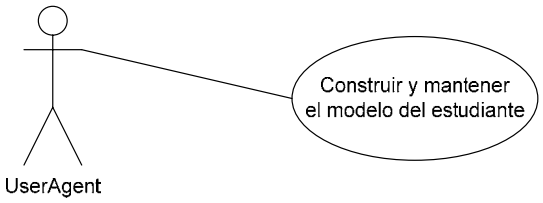


Figura 27. Diagrama de casos de uso para el agente de usuario (tomado de [30])

- **Agente pedagógico:** Selecciona las estrategias pedagógicas apropiadas al estudiante de acuerdo a su estilo de aprendizaje. Para su funcionamiento verifica las reglas de decisión pedagógica establecidas en el modelo del dominio e interactúa directamente con el agente de *usuario* para recibir el progreso del estudiante y con el agente de *navegación* para definir su actuación (navegación adaptativa) en el entorno de aprendizaje.

Los agentes *supervisor* y *controlador* llevan a cabo tareas generales para el control de funcionamiento de la plataforma.

5.2. AGENTE GENERADOR DE EJERCICIOS INTERACTIVOS

El *agente generador de ejercicios* es el primer agente que se desarrolla del sistema multiagente de la plataforma e-ESCEN@RI$_{UIS}$ y como se mencionó en la sección anterior, es el encargado de la construcción de ejercicios adaptativos interactivos para el estudiante.

El *agente generador de ejercicios* se desarrolló en JADE en la versión 3.3, utilizando el ejemplo generado por Claudiu Anghel que se encuentra en la página oficial de JADE http://jade.tilab.com/, en el cual se ejecuta un agente JADE a través de un applet[35]. Este ejemplo es de mucha ayuda, ya que el usuario puede acceder a la aplicación directamente a través de un explorador Web. Cabe resaltar que un applet integrado a agentes JADE ofrece la posibilidad que tales agentes vivan en la máquina del cliente sin que ellos necesiten más que un explorador Web que soporte Java.

A continuación se presentan las posibles percepciones y acciones que intervendrán, las metas que cumplirá y el tipo de entorno en el que operarán los agentes *UserAgent* y *GeneratorAgent* (a esto se le conoce como elementos PAMA - Percepciones, Acciones, Metas y Ambiente).

Tabla 3. Elementos PAMA que caracterizan a los agentes *usuario* y *generador de ejercicios*

	Agente usuario	**Agente generador de ejercicios**
Percepciones	Modelo de dominio y modelo del estudiante.	Ejercicios, tipos de ejercicios, temáticas, categorías y competencias
Acciones	Envío del modelo del estudiante.	Solicitud al agente usuario de los ejercicios resueltos por el estudiante
Metas	Mantener y construir el modelo del estudiante.	Generación de ejercicios interactivos de acuerdo al nivel de conocimiento del estudiante y a sus preferencias.
Ambiente	Plataforma educativa en línea e-ESCEN@RI$_{UIS}$	Plataforma educativa en línea e-ESCEN@RI$_{UIS}$

[35] Un ***applet*** es un componente de software que corre en el contexto de otro programa, por ejemplo un navegador web.

5.2.1. Comportamientos

Los casos de usos mostrados en la sección 4.1.3 sobre el agente *generador de ejercicios* (GeneratorAgent) definen los siguientes comportamientos:

- **AdaptarEjercicio:** comportamiento que construye un ejercicio adaptado al estado de conocimiento del estudiante
- **ConfigurarEjercicio:** comportamiento que permite al estudiante configurarse un ejercicio de acuerdo a sus preferencias.

5.2.2. Diagrama de actividad y ontología

- **AdaptarEjercicio:** este comportamiento sucede cuando el estudiante le solicita al agente *generador de ejercicios* que le construya un ejercicio adaptado a su nivel de conocimiento. En este caso, el agente *generador de ejercicios* le solicita al agente de *Usuario* (el cual almacena el modelo del estudiante), que le envíe el identificador de un ejercicio que cumpla con las características de acuerdo al nivel de conocimiento del estudiante. Si existe información en el modelo del estudiante, entonces obtiene el identificador de una instancia, con está información se busca en la base de datos y se construye el ejercicio (ver figura 28).

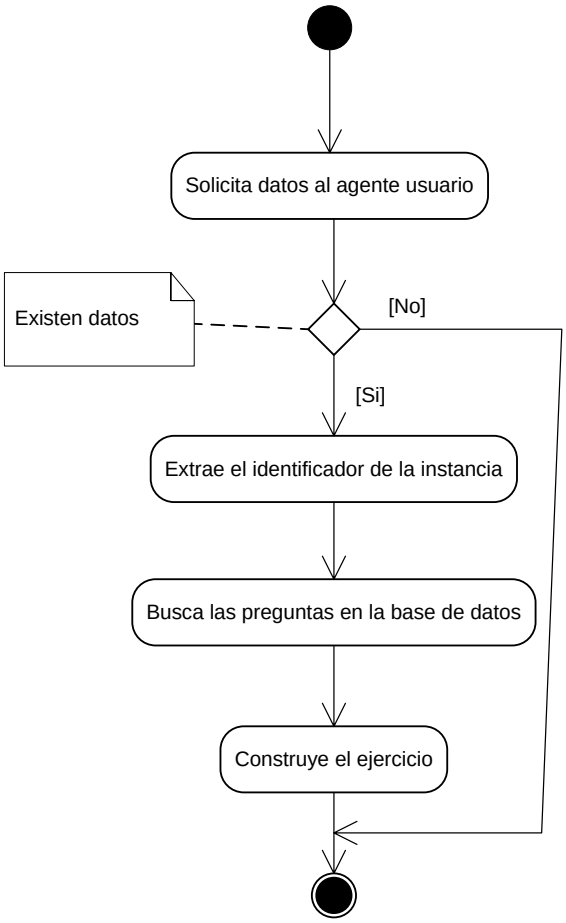


Figura 28. Diagrama de actividades para adaptar un ejercicio (tomado de [30])

- **ConfigurarEjercicio:** este comportamiento se ejecuta cuando el estudiante se configura un ejercicio de acuerdo a sus preferencias. En este caso, el estudiante selecciona el tipo de ejercicio, tema, subtema, categoría y competencia del ejercicio que desea configurar, se crea una nueva instancia, se busca en la base de datos la pregunta que cumpla con esas características y se construye el ejercicio (ver figura 29).

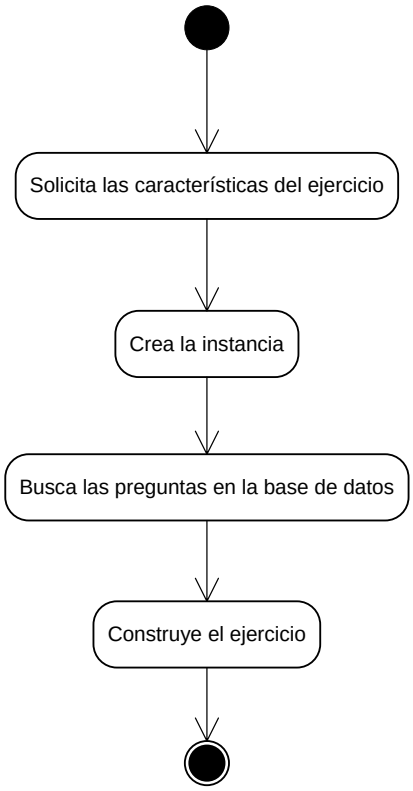


Figura 29. Diagrama de actividades para configurar un ejercicio (tomado de [30])

A continuación se presenta la documentación generada en formato HTML a través de la herramienta *javadoc*[36] de Sun Microsystems, de las clases que hacen parte de la ontología de los agentes *generador de ejercicios* y de *usuario*, estas clases son:

- **La interface *Vocabulario*:** contiene las constantes que se requieren en la aplicación (Ver tabla 4).

co.edu.uis.escenari.ontologia

Interface Vocabulario

`public interface` **`Vocabulario`**

Clase que contiene el vocabulario usado por el agente generador de ejercicios

---

[36] JavaDoc es una herramienta integrada al SDK de Java, que permite documentar de manera rápida y sencilla las clases y los métodos que se proveen.

Tabla 4. Definición de las constantes en la interface Vocabulario

Field Summary	
`static java.lang.String`	**`GENERATOR_AGENT_NAME`** Nombre del agente generador de ejercicios.
`static java.lang.String`	**`USER_AGENT_NAME`** Nombre del agente de usuario.

- **La clase *Ejercicios*:** contiene los métodos para comunicarse entre los diferentes objetos de la aplicación (Ver tablas 5 y 6).

co.edu.uis.escenari.ontologia

Class Ejercicios

```
java.lang.Object
  └co.edu.uis.escenari.ontologia.Ejercicios
```

```
public class Ejercicios
extends java.lang.Object
```

Clase que contiene la ontología del agente generador de ejercicios

---

Tabla 5. Definición de la variables de la clase Ejercicios

Field Summary	
`java.lang.String`	**`categoria`** Identificador de la categoría (Fácil, Normal o Difícil) del ejercicio
`java.lang.String`	**`competencia`** Identificador de la competencia (Argumentativa, Interpretativa o Propositiva) del ejercicio
`java.lang.String`	**`ejercicio`** Identificador del ejercicio
`java.lang.String`	**`parametro`** Identificador del parametro del esquema de la base de datos
`java.lang.String`	**`subtema`** Identificador del subtema del ejercicio
`java.lang.String`	**`tema`** Identificador del tema del ejercicio
`java.lang.String`	**`tipo_ejercicio`** Identificador del tipo de ejercicio

Tabla 6. Definición de los métodos usados en la clase Ejercicios

Method Summary	
`java.lang.String`	**`getCategoria`**`()` Método "get" para obtener el identificador de la categoría (Fácil, Normal o Difícil) del ejercicio
`java.lang.String`	**`getCompetencia`**`()` Método "get" para obtener el identificador de la competencia (Argumentativa, Interpretativa o Propositiva) del ejercicio
`java.lang.String`	**`getEjercicio`**`()` Método "get" para obtener el identificador del ejercicio
`java.lang.String`	**`getParametro`**`()` Método "get" para obtener el identificador del parametro del esquema de la base de datos
`java.lang.String`	**`getSubtema`**`()` Método "get" para obtener el identificador del subtema del ejercicio
`java.lang.String`	**`getTema`**`()` Método "get" para obtener el identificador del tema del ejercicio
`java.lang.String`	**`getTipoEjercicio`**`()` Método "get" para obtener el identificador del tipo de ejercicio
`void`	**`setCategoria`**`(java.lang.String _categoria)` Método "set" para enviar el identificador de la categoría (Fácil, Normal o Difícil) del ejercicio
`void`	**`setCompetencia`**`(java.lang.String _competencia)` Método "set" para enviar el identificador de la competencia (Argumentativa, Interpretativa o Propositiva) del ejercicio
`void`	**`setEjercicio`**`(java.lang.String _ejercicio)` Método "set" para enviar el identificador del ejercicio
`void`	**`setParametro`**`(java.lang.String _parametro)` Método "set" para enviar el identificador del parametro del esquema de la base de datos
`void`	**`setSubtema`**`(java.lang.String _subtema)` Método "set" para enviar el identificador del subtema del ejercicio
`void`	**`setTema`**`(java.lang.String _tema)` Método "set" para enviar el identificador del tema del ejercicio
`void`	**`setTipoEjercicio`**`(java.lang.String _tipo_ejercicio)` Método "set" para enviar el identificador del tipo de ejercicio

5.2.3. Diagrama de protocolos

En la figura 30 se observa el diagrama de protocolos para los agentes generador de ejercicios y de usuario de la plataforma educativa institucional e-ESCEN@RI$_{UIS}$.

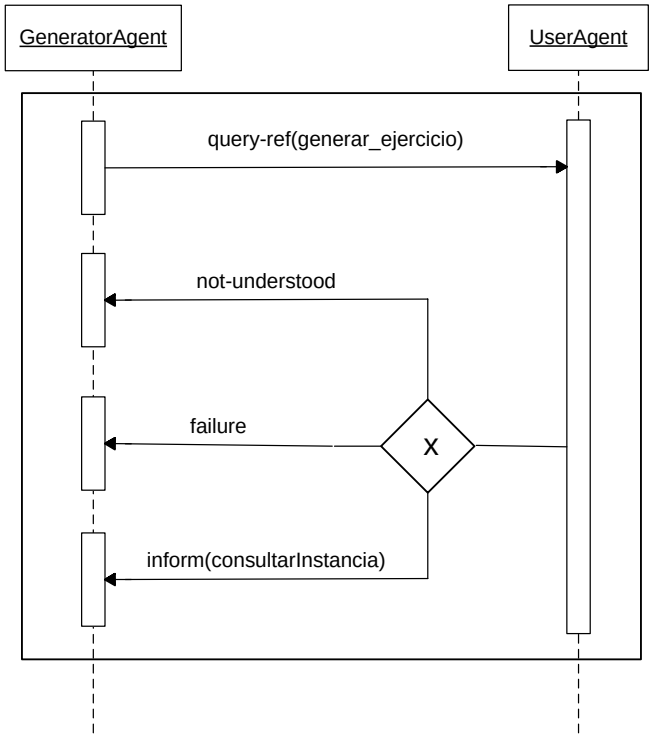


Figura 30. Diagrama de protocolos del agente generador de ejercicios (tomado de [30])

5.2.4. Desarrollo de la aplicación

Requisitos técnicos

Entre los requisitos de software instalados en la parte del servidor se requiere:

- **JDK (*Java Development Kit*)** Versión 1.5.0 o superior, el cual provee la API de Java, la máquina virtual de java (JVM - *Java Virtual Machine*), compiladores y depuradores necesarios para desarrollar y correr applets y aplicaciones escritas en el lenguaje de programación Java.

- **JADE (*Java Agent DEvelopment Framework*)** Versión 3.0b1 o superior. Es una plataforma de software que proporciona tanto un entorno de desarrollo como uno de ejecución para la realización y mantenimiento de sistemas multiagente.
- **Apache Tomcat** Versión 5.0.28 o superior. Es un servidor de aplicaciones web que funciona como un contenedor de servlets para implementar las tecnologías de Java Servlet y JavaServer Pages (JSP).

En la parte del cliente se requiere:

- **Explorador web.** e-ESCEN@RI$_{UIS}$ es una plataforma abierta e interoperable, es decir, que puede ser ejecutada desde cualquier plataforma, por lo tanto se pueden usar exploradores como: Internet Explorer 6.0 o superior, mozilla firefox, opera, nestcape, etc.

Características

La creación del *agente generador de usuarios* a través de un applet tiene las siguientes características:

1. Se crea un contenedor de agentes en el applet.
2. Se instala el HTTP MTP[37] en el contenedor creado en el applet.
3. Se crean dos agentes (*UserAgent* y *GeneratorAgent*) en el contenedor del applet.
4. Se intercambian mensajes entre el agente JADE del applet (*UserAgent*) y otro agente JADE que vive en otro contenedor (*ServerAgent*).
5. Se migra un agente (*MobileAgent*) al contenedor del applet y de este a otro contenedor.

La aplicación tiene dos partes principales: la parte del servidor y la parte del cliente.

La parte del servidor realiza las siguientes operaciones:

- Inicia la plataforma JADE en un puerto dado.

[37] Message Transport Protocols

- Crea un contenedor y le instala el HTTP MTP.
- Crea un *agente servidor* (*ServerAgent*) dentro de este contenedor.
- El *agente servidor* espera por un mensaje de otro agente (*UserAgent*).
- Después de recibir el mensaje, crea un *agente móvil (MobileAgent)*. El *agente móvil* pregunta al AMS (*Agent Management System*) dónde está el *UserAgent* y se mueve al contenedor de este agente. Después de llegar al contenedor del applet, se mueve nuevamente al contenedor del servidor.

La parte del cliente realiza las siguientes operaciones:

- Crea un applet y dentro de este crea un contenedor de agente JADE después de recibir el nombre del host y el puerto de la plataforma principal del JADE.
- Instala el HTTP MTP en el contenedor creado.
- Dentro del contenedor crea a los agentes *UserAgent* y *GeneratorAgent*, el primero envía un mensaje al *ServerAgent* que solicita la migración del *MobileAgent*.

Desarrollo

En esta sección se explican los pasos seguidos para el desarrollo del agente generador de ejercicios interactivos de la plataforma educativa institucional e-ESCEN@RI_{UIS}.

Lo primero que se hizo para el desarrollo de la aplicación fue crear un sitio llamado *jade* en el servidor tomcat del servidor *torcaza.uis.edu.co* con la estructura mostrada en la figura 31.

Luego se procedió a realizar una serie de pruebas sobre la construcción de agentes en JADE y de applets en java con el objetivo de aprender sobre estos tipos de aplicaciones. Finalmente se realizó la ejecución de la aplicación con las características explicadas anteriormente.

A continuación se muestran los pasos seguidos para:

- El desarrollo del *agente generador de ejercicios* desde la consola de JADE,
- La creación de un applet que consulta las unidades académicas de un estudiante y

- El desarrollo del *agente generador de ejercicios interactivos* ejecutado desde un applet.

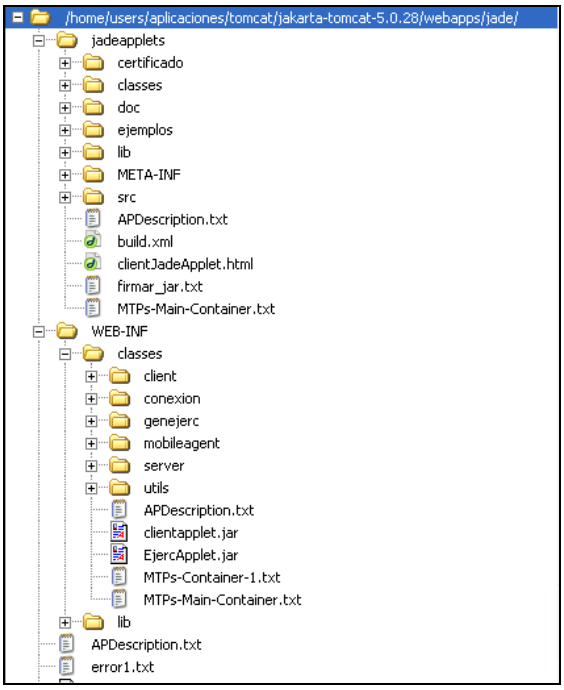


Figura 31. Estructura del sitio de la aplicación

Agente generador de ejercicios ejecutado desde consola

A través de este ejemplo se programa la comunicación que deben tener los agentes de *usuario* (*UserAgent*) y *generador de ejercicios* (*GeneratorAgent*) del sistema multiagente del e-ESCEN@RI$_{UIS}$.

- **Ruta de los archivos:** la ubicación de los archivos requeridos para el ejemplo se encuentra en la ruta: /home/users/aplicaciones/tomcat/jakarta-tomcat-5.0.28/webapps/jade/WEB-INF/classes/genejerc

- **Clases en java**
  - ***CUserAgent.java:*** Contiene el método *consultarInstancia(String _user_id, String _unidad_id)* que recibe como parámetros el identificador del usuario y de la unidad

docente. A través de este método se consultan en la base de datos los ejercicios resueltos por el estudiante y se selecciona aleatoriamente un ejercicio.

- ***UserAgent.java:*** Agente que llama el método *consultarInstancia()* de la clase *CUserAgent* y envía un mensaje ACL FIPA al agente *EjercAgent* con el resultado.
- ***EjercAgent.java:*** Agente que muestra el identificador del ejercicio generado por el *agente de usuario* (*UserAgent*).

- **Compilación de las clases:** la ruta de ubicación para compilar las clases es /home/users/aplicaciones/tomcat/jakarta-tomcat-5.0.28/webapps/jade/WEB-INF/classes, luego se escribe lo siguiente:

```
[tomcat@torcaza classes]$ javac genejerc/CUserAgent.java
[tomcat@torcaza classes]$ javac genejerc/UserAgent.java
[tomcat@torcaza classes]$ javac genejerc/EjercAgent.java
```

- **Ejecución de los agentes:** en la figura 32 se muestra el comportamiento de la plataforma JADE al ejecutar ambos agentes al tiempo con la siguiente instrucción:

```
[tomcat@torcaza classes]$ java jade.Boot sender:genejerc.UserAgent a1:genejerc.EjercAgent
```

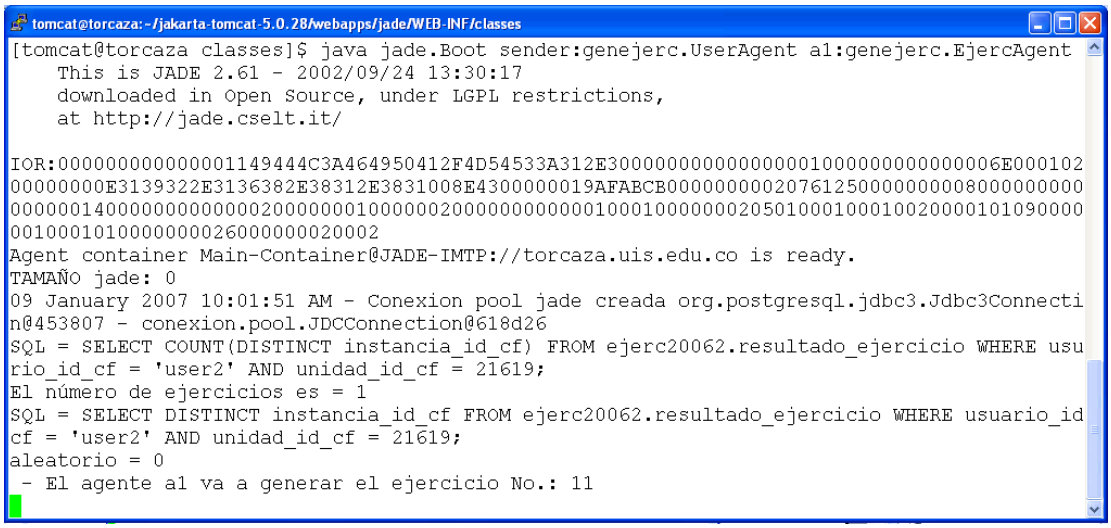


Figura 32. Ejecución simultanea de los agentes JADE EjercAgent y UserAgent desde consola

Para crear ambos agentes independientemente, primero se ejecuta el agente *EjercAgent* así (ver figura 33):

[tomcat@torcaza classes]$ java jade.Boot a1:genejerc.EjercAgent y luego se ejecuta al agente *UserAgent* con la siguiente instrucción (ver figura 34):

[tomcat@torcaza classes]$ java jade.Boot -container sender:genejerc.UserAgent

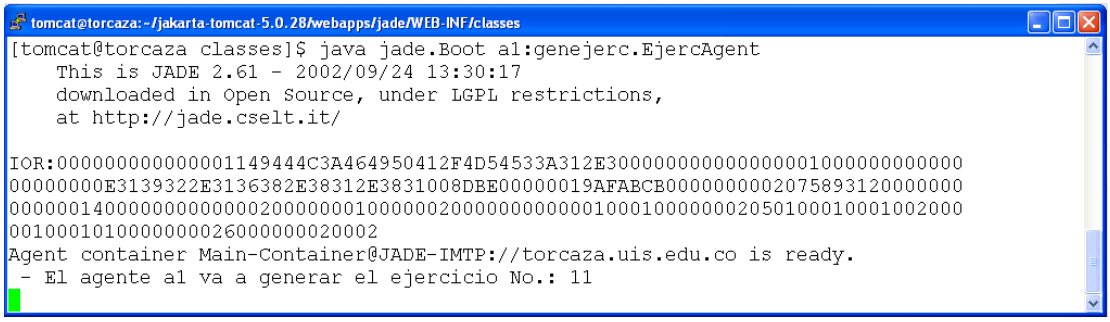


Figura 33. Ejecución del agente JADE EjercAgent desde consola

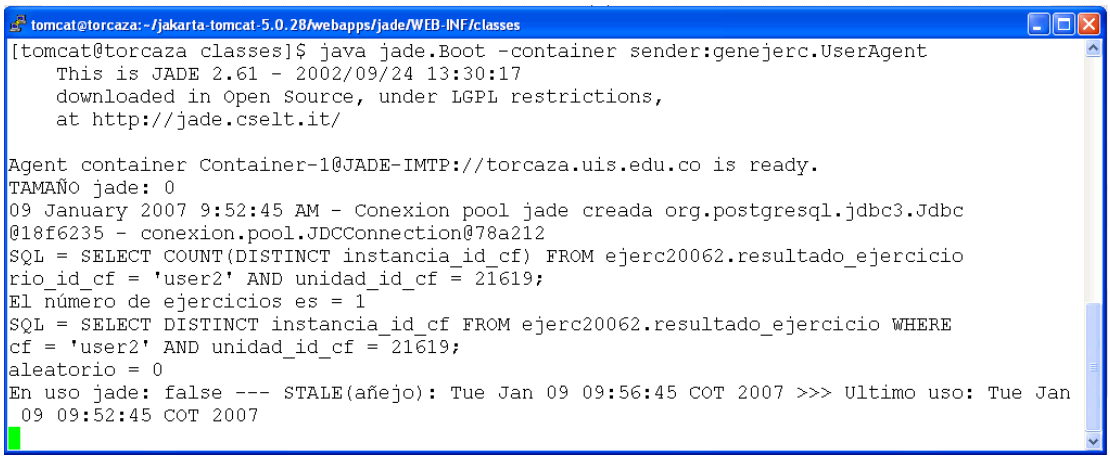


Figura 34. Ejecución del agente JADE UserAgent desde consola

Creación de un applet

A través de este ejemplo se crea un applet que consulta las asignaturas cursadas por el estudiante y las muestra en un combo.

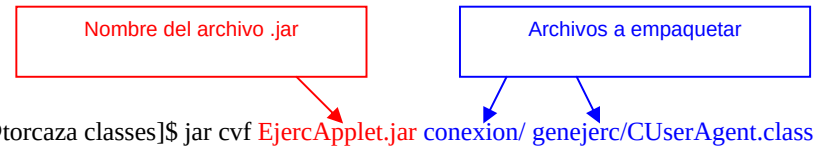
- **Ruta de los archivos:** la ubicación de las clases requeridas para la creación del applet se encuentra en la ruta: /home/users/aplicaciones/tomcat/jakarta-tomcat-5.0.28/webapps/jade/WEB-INF/classes/genejerc

  - **CUserAgent.java:** Contiene el método *consultarUnidadesDocentesEst(String _user_id)* para realizar la consulta en la base de datos de las asignaturas que tiene asociado el estudiante.
  - **getUnidadesDoc.java:** Contiene el método *consultarUnidades (String _user_id)*, en donde se realiza la conexión a la base de datos y la llamada al método *consultarUnidadesDocentesEst* de la clase *CUserAgent.java*
  - **EjercApplet.java:** Applet donde se muestran los resultados de la consulta de las unidades académicas existentes. En esta clase se llama al método *consultarUnidades* de la clase *getUnidadesDoc.java*

- **Compilación de las clases:** la ruta de ubicación para compilar las clases es /home/users/aplicaciones/tomcat/jakarta-tomcat-5.0.28/webapps/jade/WEB-INF/classes, y se compila así:

```
[tomcat@torcaza classes]$ javac genejerc/CUserAgent.java
[tomcat@torcaza classes]$ javac genejerc/getUnidadesDoc.java
[tomcat@torcaza classes]$ javac genejerc/EjercApplet.java
```

- **Creación del archivo .jar[38]:** las clases compiladas anteriormente se empaquetan a través de un jar. Estando ubicados en /home/users/aplicaciones/tomcat/jakarta-tomcat-5.0.28/webapps/jade/WEB-INF/classes se ejecuta la siguiente línea de comando:




```
[tomcat@torcaza classes]$ jar cvf EjercApplet.jar conexion/ genejerc/CUserAgent.class
genejerc/EjercApplet.class genejerc/getUnidadesDoc.class utils/Split.class
```

- **Firma del jar:** previamente se ha generado un certificado para firmar el jar en la ruta /home/users/aplicaciones/tomcat/jakarta-tomcat-5.0.28/webapps/jade/jadeapplets/certificado, con la siguiente instrucción:

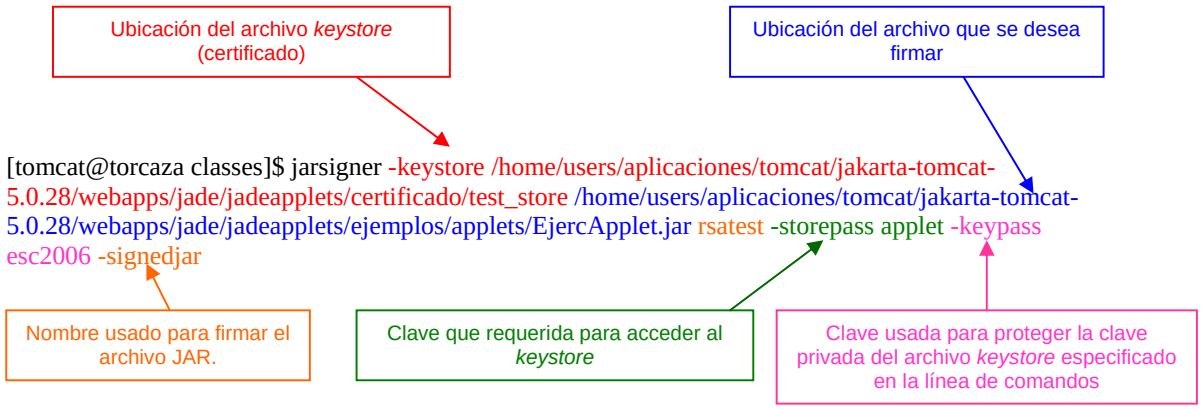
[38] .jar: conjuntos de varios archivos comprimidos en formato ZIP

[tomcat@torcaza classes]$ keytool -genkey -alias **rsatest** -keypass **esc2006** -keystore **test_store** -storepass **applet**

Donde:

- *keystore* es la ubicación del archive *keystore* (certificado).
- *storepass* es la clave requerida para acceder al *keystore*.
- *keypass* es la clave usada para proteger la clave privada del archivo *keystore* especificado en la línea de comandos.
- *alias* para el archivo del certificado.

Luego, el archivo *EjercApplet.jar* se copia en /home/users/aplicaciones/tomcat/jakarta-tomcat-5.0.28/webapps/jade/jadeapplets/ejemplos/applets y se firma así:



Enter Passphrase for keystore: applet
Enter key password for rsatest: esc2006

- **Creación de la página jsp:** se crea la página *pruebaApplet.jsp* en la ruta /home/users/aplicaciones/tomcat/jakarta-tomcat-5.0.28/webapps/jade/jadeapplets/ejemplos/applets, para mostrar el applet la cual contiene el siguiente código en el *body*:

```
<body>
<applet code=genejerc.EjercApplet.class width=300 height=200>
<param name="archive" value="EjercApplet.jar, postgresql-8.1-406.jdbc3.jar">
<param name="user_id" value="user2">
</applet>
</body>
```

- **Ejecución del applet:** finalmente se ejecuta el applet en el explorador Web así: http://torcaza.uis.edu.co:8080/jade/jadeapplets/ejemplos/applets/pruebaApplet.jsp

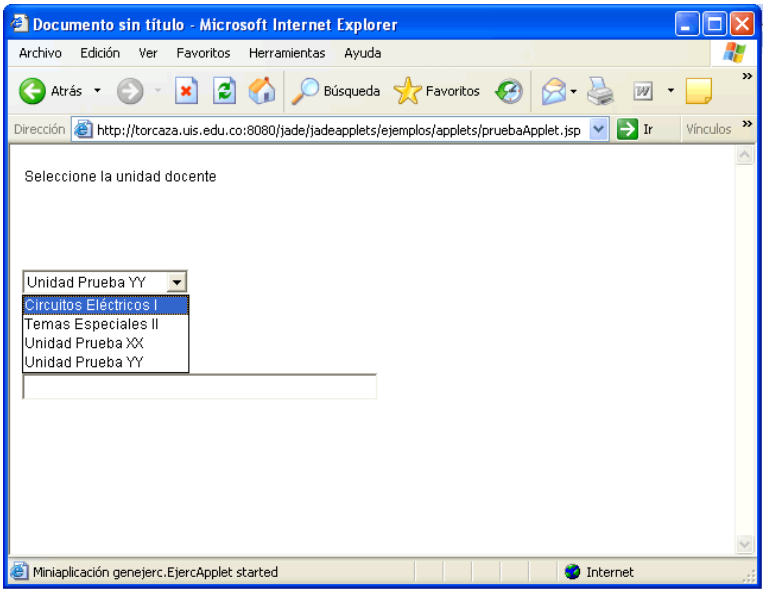


Figura 35. Ejecución del applet que consulta las asignaturas cursadas por el estudiante

Creación del agente generador de ejercicios

A continuación se presenta el procedimiento seguido para la creación del *agente generador de ejercicios* de la plataforma educativa institucional e-ESCEN@RI$_{UIS}$.

- **Estructura del sitio de la aplicación:** la aplicación tiene la siguiente estructura de carpetas:
  - **src:** contiene el código fuente en java del ejemplo de Claudiu Anghel que se encuentra en la página oficial de JADE http://jade.tilab.com, organizado en tres paquetes: server (implementa el servidor, el *agente servidor* y su comportamiento), mobileagent (implementa el *agente móvil* y su comportamiento) y client (implementa el applet, el *agente applet* y su comportamiento).
  - **lib:** contiene el archivo *clientapplet.jar*; los archivos jar del JADE *(jade.jar, Base64.jar, iiop.jar, jadeTools.jar, http.jar, sax2.jar*); el driver de postgresql

(*postgresql-8.1-406.jdbc3.jar*); la página jsp donde se ejecuta la plataforma del JADE del lado del cliente (*clientJadeApplet.jsp*).

- **WEB-INF/classes:** contenedor de las clases compiladas. Su estructura es la siguiente
  - **client:** clases para la implementación de los *agentes UserAgent y GenaratorAgent* (*AppletAgent.class, AppletAgentBehaviour.class, ClientApplet.class, GeneratorAgent.class, GeneratorApplet.class y UserAgent.class*).
  - **conexion:** clases de conexión con la base de datos *escenari* creada en postgresql.
  - **genejerc:** clases que contienen los métodos de consulta a la base de datos para obtener la información del modelo del dominio y del estudiante para el usuario.
  - **mobileagent:** clases para la implementación del *agente móvil* (*MobileAgent.class y WhereIsAgent.class*).
  - **ontologia:** clases para la implementación de la ontología de los agentes (*Ejercicios.class* y *Vocabulario.class*)
  - **server:** clases para la implementación del *ServerAgent* (*Server.class, ServerAgent.class y ServerAgentBehaviour.class*).

- **Compilación de las clases:** la ruta de ubicación para compilar las clases es /home/users/aplicaciones/tomcat/jakarta-tomcat-5.0.28/webapps/jade/WEB-INF/classes, y se compila así:

```
[tomcat@torcaza classes]$ javac -cp /home/users/aplicaciones/tomcat/jakarta-tomcat-
5.0.28/webapps/jade/jadeapplets/lib/jade.jar:.:/home/users/aplicaciones/tomcat/jakarta-tomcat-
5.0.28/webapps/jade/WEB-INF/classes/server/ server/Server.java

[tomcat@torcaza classes]$ javac -cp /home/users/aplicaciones/tomcat/jakarta-tomcat-
5.0.28/webapps/jade/jadeapplets/lib/jade.jar:.:/home/users/aplicaciones/tomcat/jakarta-tomcat-
5.0.28/webapps/jade/WEB-INF/classes/server/ ontologia/*.java

[tomcat@torcaza classes]$ javac -cp /home/users/aplicaciones/tomcat/jakarta-tomcat-
5.0.28/webapps/jade/jadeapplets/lib/jade.jar:.:/home/users/aplicaciones/tomcat/jakarta-tomcat-
5.0.28/webapps/jade/WEB-INF/classes/server/ client/GeneratorApplet.java
```

- **Creación del archivo .jar:** las clases compiladas anteriormente se empaquetan a través de un jar. Estando ubicados en /home/users/aplicaciones/tomcat/jakarta-tomcat-5.0.28/webapps/jade/WEB-INF/classes se ejecuta la siguiente línea de comando:

  ```
  [tomcat@torcaza classes]$ jar cvf clientapplet.jar genejerc/*.class conexion/*.class ontologia/*.class
  utils/Split.class client/*.class mobileagent/*.class server/*.class
  ```

- **Firma del applet:** previamente se ha generado un certificado para firmar el applet. El archivo clientapplet.jar, se copia en /home/users/aplicaciones/tomcat/jakarta-tomcat-5.0.28/webapps/jade/jadeapplets/lib y se firma de la siguiente forma:

  ```
  jarsigner -keystore /home/users/aplicaciones/tomcat/jakarta-tomcat-
  5.0.28/webapps/jade/jadeapplets/certificado/test_store /home/users/aplicaciones/tomcat/jakarta-tomcat-
  5.0.28/webapps/jade/jadeapplets/lib/clientapplet.jar rsatest -storepass applet -keypass esc2006 .signedjar

  Enter Passphrase for keystore: applet
  Enter key password for rsatest: esc2006
  ```

- **Creación de la página jsp:** se crea la página *clientJadeApplet.jsp* para arrancar el sistema multiagente del e-ESCEN@RI$_{UIS}$ en la ruta /home/users/aplicaciones/tomcat/jakarta-tomcat-5.0.28/webapps/jade/jadeapplets/lib, la cual contiene en el *body* el siguiente código fuente:

  ```
  <body>
  <applet code=client.ClientApplet.class width=300 height=150>
  <param name="archive" value="clientapplet.jar, postgresql-8.1-406.jdbc3.jar, jade.jar, http.jar,
  base64.jar, iiop.jar, jadeTools.jar, sax2.jar, sax2r2.jar">
  <param name="jadeHostName" value="torcaza.uis.edu.co">
  <param name="jadePort" value="2002">
  <param name="appletContainerPort" value="2003">
  <param name="user_id" value="<%=user_id%>">
  </applet>
  </body>
  ```

- **Ejecución del agente:** finalmente, para ejecutar el agente se fusiona la aplicación del agente generador de ejercicios interactivos con la plataforma educativa institucional e-ESCEN@RI$_{UIS}$. A continuación se presentan los resultados de la ejecución del agente:

  En el servidor se debe ejecutar la siguiente instrucción para arrancar el JADE con su interfaz gráfica y con los agentes especiales del contenedor principal (AMS - *Agent*

Management System y el DF - *Directory Facilitator*), así como los agentes *ServerAgent y MobileAgent* de la aplicación (Ver figura 36):

```
java -classpath ./WEB-
INF/classes:./jadeapplets/lib/jade.jar:./jadeapplets/lib/base64.jar:./jadeapplets/lib/iiop.jar:./jadeapplets/lib/
jadeTools.jar:./jadeapplets/lib/http.jar:./jadeapplets/lib/sax2.jar:./jadeapplets/lib/clientapplet.jar
server.Server 2002 2003
```

La información mostrada en la consola del servidor es la siguiente:

```
[root@torcaza jade]# java -classpath ./WEB-
INF/classes:./jadeapplets/lib/jade.jar:./jadeapplets/lib/base64.jar:./jadeapplets/lib/iiop.jar:./jadeapplets/lib/
jadeTools.jar:./jadeapplets/lib/http.jar:./jadeapplets/lib/sax2.jar:./jadeapplets/lib/clientapplet.jar
server.Server 2002 2003
El puerto es = 2002
09-Jan-2007 16:26:12 jade.core.Runtime beginContainer
INFO: ----------------------------------
   This is JADE 3.3 - 2005/03/02 16:11:05
   downloaded in Open Source, under LGPL restrictions,
   at http://jade.cselt.it/
----------------------------------------
09-Jan-2007 16:26:13 jade.core.BaseService init
INFO: Service jade.core.management.AgentManagement initialized
09-Jan-2007 16:26:13 jade.core.BaseService init
INFO: Service jade.core.messaging.Messaging initialized
09-Jan-2007 16:26:13 jade.core.messaging.MessagingService boot
INFO: MTP addresses:
http://torcaza.uis.edu.co:7778/acc
09-Jan-2007 16:26:13 jade.core.BaseService init
INFO: Service jade.core.mobility.AgentMobility initialized
09-Jan-2007 16:26:13 jade.core.BaseService init
INFO: Service jade.core.event.Notification initialized
09-Jan-2007 16:26:13 jade.core.AgentContainerImpl joinPlatform
INFO: ----------------------------------------
Agent container Main-Container@JADE-IMTP://torcaza.uis.edu.co is ready.
---------------------------------------------
09-Jan-2007 16:26:13 jade.core.Runtime beginContainer
INFO: ----------------------------------
   This is JADE 3.3 - 2005/03/02 16:11:05
   downloaded in Open Source, under LGPL restrictions,
   at http://jade.cselt.it/
----------------------------------------
09-Jan-2007 16:26:14 jade.core.BaseService init
INFO: Service jade.core.management.AgentManagement initialized
09-Jan-2007 16:26:14 jade.core.BaseService init
INFO: Service jade.core.messaging.Messaging initialized
09-Jan-2007 16:26:14 jade.core.PlatformManagerImpl localAddNode
INFO: Adding node <Container-1> to the platform
09-Jan-2007 16:26:14 jade.core.PlatformManagerImpl$1 nodeAdded
INFO: --- Node <Container-1> ALIVE ---
```

09-Jan-2007 16:26:14 jade.core.messaging.MessagingService boot
INFO: MTP addresses:
http://torcaza.uis.edu.co:2003/server
09-Jan-2007 16:26:14 jade.core.BaseService init
INFO: Service jade.core.mobility.AgentMobility initialized
09-Jan-2007 16:26:14 jade.core.BaseService init
INFO: Service jade.core.event.Notification initialized
09-Jan-2007 16:26:14 jade.core.AgentContainerImpl joinPlatform
INFO: --
Agent container Container-1@JADE-IMTP://torcaza.uis.edu.co is ready.

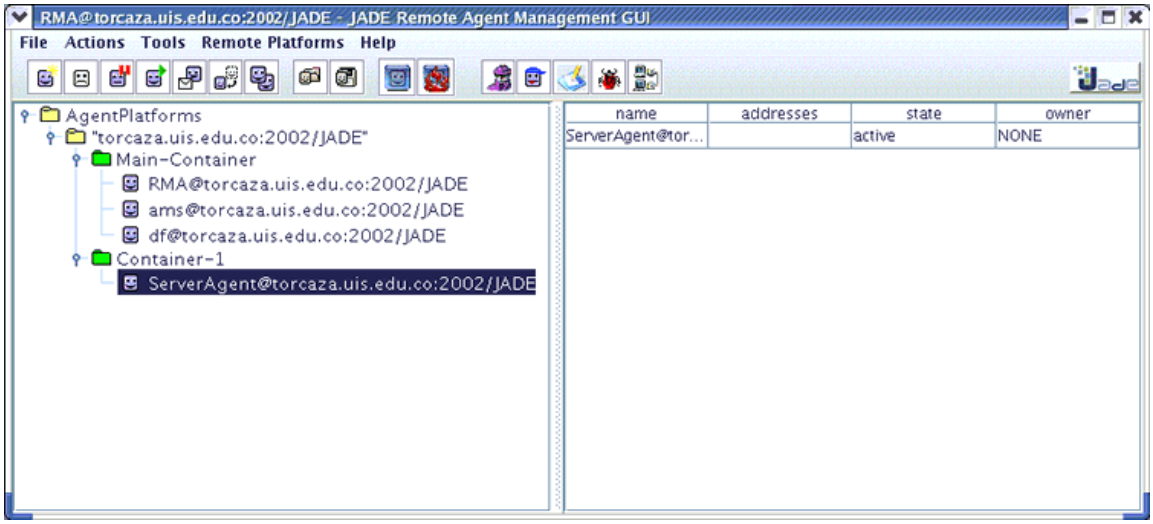


Figura 36. Interfaz gráfica del JADE para el sistema multiagente de la plataforma e-ESCEN@RI$_{UIS}$

Luego se procede a abrir el sitio de la plataforma educativa institucional e-ESCEN@RI$_{UIS}$ (http://torcaza.uis.edu.co:8080/escenariII/cuenta1.jsp), se ingresa con una cuenta de perfil de estudiante, automáticamente se carga el sistema multiagente a través de un applet, como se observa en la figura 37.

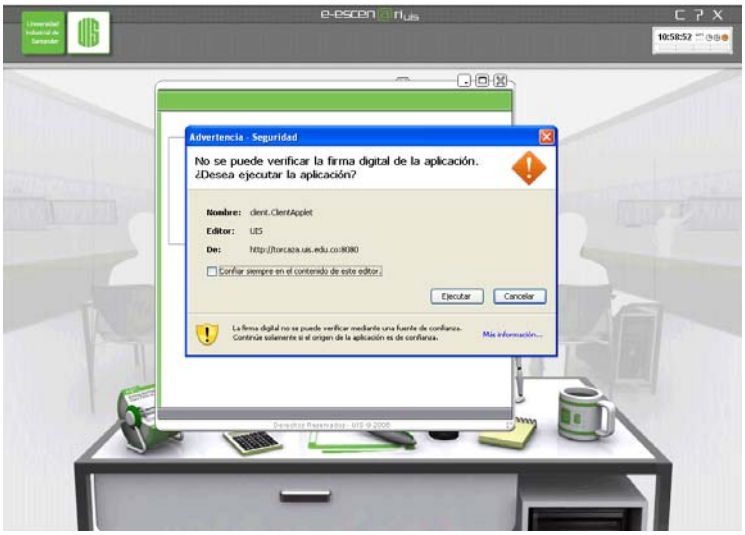


Figura 37. Ejecución del sistema multiagente del lado del cliente en la plataforma educativa e-ESCEN@RI$_{UIS}$

En la consola de java del explorador de Internet se observa lo siguiente:

```
Java Plug-in 1.5.0_10
Usar versión JRE 1.5.0_10 Java HotSpot(TM) Client VM
Directorio local del usuario = C:\Documents and Settings\UIS1


----------------------------------------------------
c:   borrar ventana de consola
f:   finalizar objetos en la cola de finalización
g:   liberación de recursos
h:   presentar este mensaje de ayuda
l:   volcar lista del cargador de clases
m:   imprimir sintaxis de memoria
o:   activar registro
p:   recargar configuración de proxy
q:   ocultar consola
r:   recargar configuración de norma
s:   volcar propiedades del sistema y de despliegue
t:   volcar lista de subprocesos
v:   volcar pila de subprocesos
x:   borrar antememoria del cargador de clases
0-5: establecer nivel de rastreo en <n>
----------------------------------------------------

jadePort = 2002
clientPort = 2003
25/01/2007 11:03:21 AM jade.core.Runtime beginContainer
INFO: ----------------------------------
```

```
   This is JADE 3.3 - 2005/03/02 16:11:05
   downloaded in Open Source, under LGPL restrictions,
   at http://jade.cselt.it/
----------------------------------------
25/01/2007 11:03:22 AM jade.core.BaseService init
INFO: Service jade.core.management.AgentManagement initialized
25/01/2007 11:03:22 AM jade.core.BaseService init
INFO: Service jade.core.messaging.Messaging initialized
25/01/2007 11:03:22 AM jade.core.messaging.MessagingService boot
INFO: MTP addresses:
http://torcaza.uis.edu.co:2003/test
25/01/2007 11:03:22 AM jade.core.BaseService init
INFO: Service jade.core.mobility.AgentMobility initialized
25/01/2007 11:03:22 AM jade.core.BaseService init
INFO: Service jade.core.event.Notification initialized
25/01/2007 11:03:22 AM jade.core.AgentContainerImpl joinPlatform
INFO: ----------------------------------------
Agent container Container-2@JADE-IMTP://labcentic6 is ready.
---------------------------------------------
El UserAgent ha sido creado
*** ClientApplet 6 -- mensaje El UserAgent ha sido creado.
Se inicia la ejecución del UserAgent
Message 'Send Mobile Agent' sent to server agent.
*** ClientApplet 6 -- mensaje Mensage 'Iniciando ejecución de Agente Movil' enviado a agente servidor.
Se REGISTRÓ UserAgent en el DF
+++ Creado: ( agent-identifier :name user2GeneratorAgent@torcaza.uis.edu.co:2002/JADE )
Se inicia la ejecución del GeneratorAgent
Se REGISTRÓ GeneratorAgent en el DF
```

En el JADE se crea el contenedor y los agentes del usuario (ver figura 38) y se registran en el *Directory Facilitator* (ver figura 39). Es importante resaltar que se crean tantos contenedores en JADE como número de clientes conectados haya en la plataforma educativa institucional e-ESCEN@RI_{UIS}, cada contenedor tiene los agentes de cada uno de los usuarios. El nombre de estos agentes se forma con el nombre de la cuenta del usuario seguido del nombre del agente, por ejemplo para el usuario "*user2*" serían los siguientes agentes: *user2UserAgent* y *user2GeneratorAgent*.

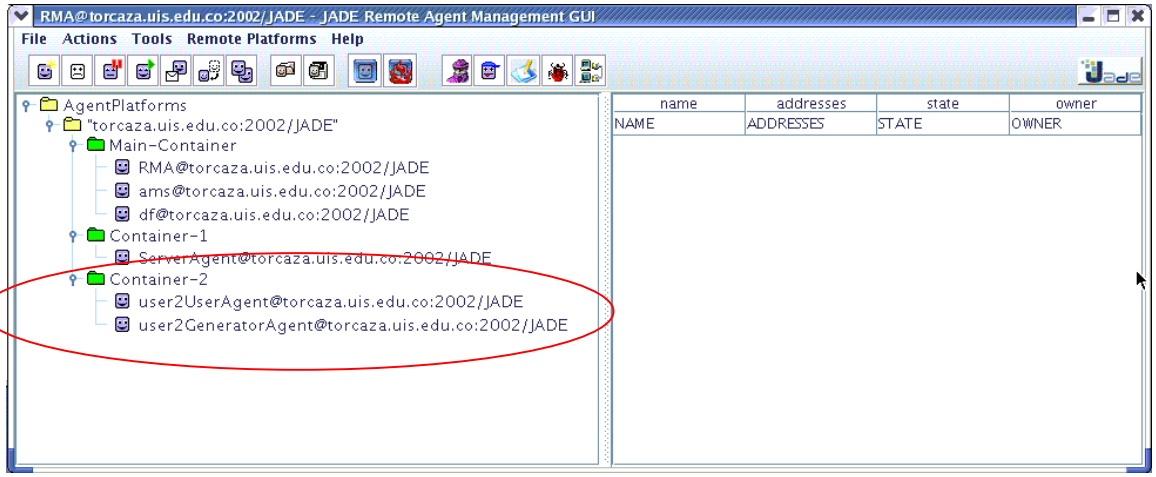


Figura 38. Creación del contenedor y los agentes del usuario de la plataforma e-ESCEN@RI$_{UIS}$

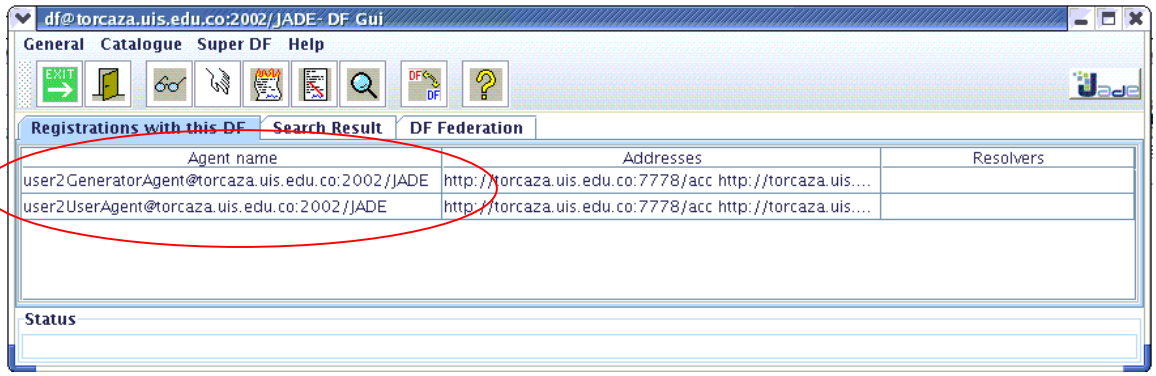


Figura 39. Registro de los agentes en el Directory Facilitator de la plataforma multiagente de e-ESCEN@RI$_{UIS}$

En la figura 40, se puede observar la monitorización a través del agente *sniffer* de los mensajes intercambiados entre el *Directory Facilitator* y los agentes *User2UserAgent* y *User2GeneratorAgent*. La comunicación es la siguiente: el agente (*UserAgent* y *GeneratorAgent*) a través de un mensaje (*REQUEST*) le solicita al DF que lo registre y el DF le envía un mensaje (*INFORM*) aceptando la solicitud del agente.

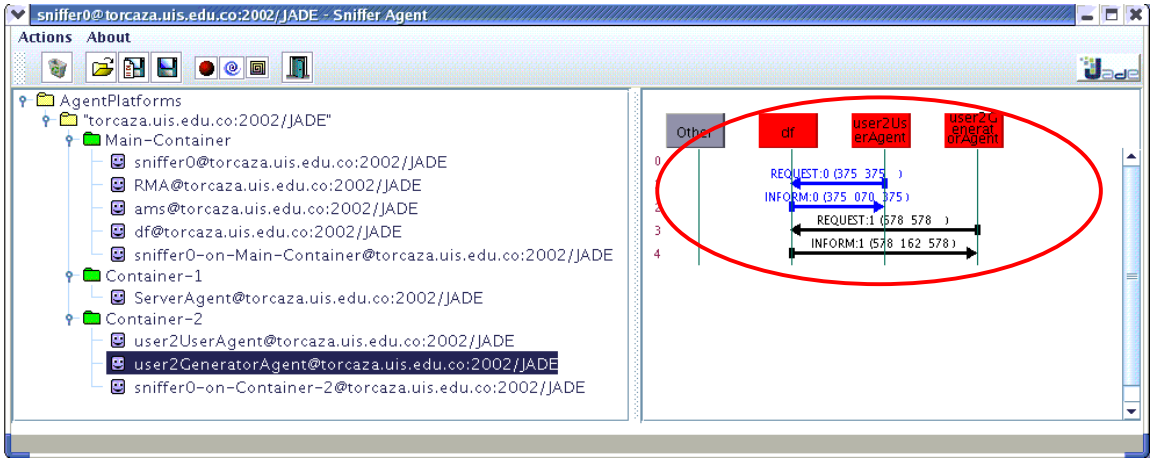


Figura 40. Monitorización de los agentes de la plataforma a través del agente sniffer

Posteriormente, el estudiante abre el *gestor de evaluación* a través del icono señalado en la figura 41 con el número 1. Se despliega una ventana que contiene las unidades docentes que el estudiante está cursando, para cada una de éstas aparece la opción de ver: los ejercicios propuestos por el profesor (ver en la figura 41 el número 2), los ejercicios configurables (ver en la figura 41 el número 3) y los ejercicios adaptativos (ver en la figura 41 el número 4).

Si se desea resolver un ejercicio de acuerdo con el nivel de conocimiento se hace clic sobre el icono de "*Ejercicio adaptativo*" y automáticamente el agente *generador de ejercicios* (*GeneratorAgent*) le solicita al agente de *usuario* (*UserAgent*) el modelo del estudiante para adaptarle un ejercicio al estudiante y envía como parámetro el identificador del ejercicio a la página *form_inst_mix.jsp*, en la cual se cargan las preguntas y respuestas (la interfaz con el ejercicio se puede observar en la figura 43).

Pero, si se desea configurar un ejercicio de acuerdo a las preferencias del estudiante, se hace clic sobre el icono de "*Ejercicio configurable*"; aparece una interfaz para seleccionar el tipo de ejercicio, tema, subtema, categoría y competencia como se observa en la figura 42; posteriormente el agente *generador de ejercicios* (*GeneratorAgent*) consulta en la base de datos, el modelo del dominio de la unidad docente para construirle el ejercicio

con la configuración deseada y envía como parámetro el identificador del ejercicio a la página *form_inst_mix.jsp*, en la cual se cargan las preguntas y respuestas (la interfaz con el ejercicio se puede observar en la figura 43).

Figura 41. Interfaz del gestor de evaluación de la plataforma educativa e-ESCEN@RI$_{UIS}$

Figura 42. Interfaz para configurar un ejercicio de acuerdo a las preferencias del estudiante en la plataforma educativa institucional e-ESCEN@RI$_{UIS}$

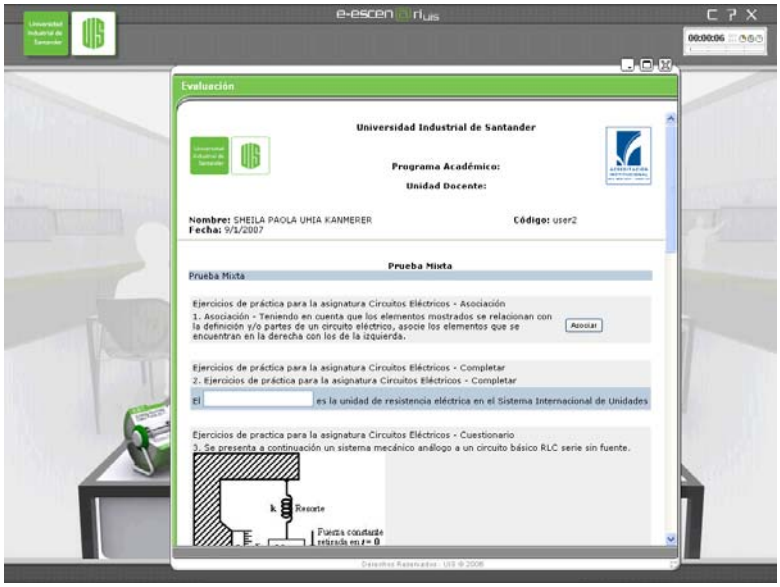


Figura 43. Ejercicio creado por el agente generador de ejercicios de la plataforma educativa e-ESCEN@RI$_{UIS}$

La construcción y evaluación de los ejercicios tipo test se hace siguiendo los lineamientos propuestos por el Instituto Colombiano para el Fomento de la Educación Superior (ICFES) para los Exámenes de estado de la Calidad de la Educación Superior (ECAES).

De esta forma funciona el agente generador de ejercicios de la plataforma educativa institucional e-ESCEN@RI$_{UIS}$.

CONCLUSIONES

El e-ESCEN@RI$_{UIS}$ es usado como apoyo a la educación presencial impartida en la Universidad Industrial de Santander, para reforzar el conocimiento y dar asistencia a los estudiantes de acuerdo a su estilo de aprendizaje y nivel de conocimiento. Esta plataforma cuenta con una serie de herramientas que le permite tanto al docente como al estudiante interactuar con el sistema para realizar sus actividades curriculares. De igual forma, a través de esta plataforma educativa institucional el profesor puede contar con información relevante sobre el desempeño del estudiante para que pueda enfocar sus estrategias pedagógicas según el rendimiento de los estudiantes.

El agente generador de ejercicios les permite a los estudiantes la construcción de ejercicios adaptativos considerando su estado de conocimiento o sus preferencias para contribuir en su proceso de aprendizaje.

El desarrollo del agente generador de ejercicios mediante JADE le permite a la plataforma educativa institucional e-ESCEN@RI$_{UIS}$ contar con un sistema multiagente de código abierto, que cumple con las especificaciones de FIPA y donde los agentes se pueden comunicar y cooperar de forma simple.

Los Sistemas Tutores Inteligentes son una buena opción para el uso de los computadores en el proceso de formación de personas. Sin duda, una de sus grandes ventajas es la capacidad de brindar enseñanza personalizada a través del uso de las Tecnologías de Información y Comunicación. Cabe resaltar que enseñar no es una tarea fácil y la construcción de un sistema que pueda simular totalmente el comportamiento de un profesor, es tal vez imposible, a pesar de que cada día haya más avances en el área de la Inteligencia Artificial.

El uso de sistemas educativos en línea genera un cambio en la forma de la enseñanza tradicional, ya que implica asumir un compromiso y plantear estrategias de enseñanza/aprendizaje por parte de los diferentes actores del proceso educativo. Es bueno aclarar que el propósito de estos sistemas no es reemplazar al profesor, sino servir como una herramienta eficaz y de valor tecnológico en el proceso de enseñanza.

RECOMENDACIONES

Aumentar el apoyo por parte de las directivas de la Universidad Industrial de Santander y demás miembros de la comunidad universitaria (profesores, estudiantes, etc.) para crear compromiso institucional para seguir las políticas y lineamientos propuestos en el Proyecto ProSPETIC (*PROyecto Soporte al Proceso Educativo UIS mediante Tecnologías de Información y Comunicación*).

Seguir con el mejoramiento continuo del agente generador de ejercicios de la plataforma educativa institucional e-ESCEN@RI$_{UIS}$, de acuerdo con los cambios generados en la plataforma de desarrollo de agentes JADE, la cual está en constante revisión por parte del grupo de desarrollo del Tilab (*Telecomm Italia Lab*).

Incentivar este tipo de proyectos de investigación en la Universidad para contribuir con la misión de formar personas integrales que generen cambios por el progreso y mejoramiento de la calidad de vida de la comunidad.

GLOSARIO

ACL (Agent Communication Lenguaje): Estructura de comunicación entre agentes definida por la FIPA (Foundation for Intelligent Physical Agents)

Adaptabilidad: Es la posibilidad de permitir al usuario, modificar los parámetros del sistema para adaptarlo así a su comportamiento.

Adaptatividad: Es la capacidad del sistema de adaptarse *automáticamente* al usuario, basado en suposiciones sobre el mismo.

Agencia: Ensamblado de partes (agentes) con una tarea resultante de su unidad integrada, descartando lo que cada una de las partes realiza por sí mismas.

API: (Application Program Interface) Servicio de un sistema operativo disponibles para programas que funcionan con dicho sistema operativo.

Applet: Es un componente de software que corre en el contexto de otro programa, por ejemplo un navegador Web.

CENTIC: CENtro de Tecnologías de Investigación y Comunicación de la Universidad Industrial de Santander.

CORBA (Common Object Request Broker Architecture): es un estándar que establece una plataforma de desarrollo de sistemas distribuidos facilitando la invocación de métodos remotos bajo un paradigma orientado a objetos.

e-Learning: Incluye una amplia gama de aplicaciones y procesos, tales como aprendizaje basado en la red, en el computador, aulas virtuales, cooperación digital. Incluye la entrega

de contenidos vía Internet, extranet, intranet, (LAN/WAN), audio y vídeo, emisión satelital, televisión interactiva y CD-ROM.

Enlace: Resultado de la marcación en lenguaje HTML, un enlace indica al navegador que los datos en un documento se comunicarán automáticamente con datos anidados o con recursos externos. Utilizado en el diseño de hipertexto.

FIPA (Foundation for Intelligent Physical Agents): es una organización internacional que se dedica a promover la industria de agentes inteligentes abiertamente desarrollando las especificaciones que apoyan la interoperabilidad entre agentes. Esto ocurre con la colaboración entre sus organizaciones miembro, que son las compañías y las universidades que están activas en el campo de agentes.

Formación asistida por computador: Instrucción por medio de un computador, en donde el sistema permite la recuperación basándose en respuestas, pero no permite un cambio en la estructura subyacente al programa.

HTML (HyperText Markup Language - Lenguaje de marcación de hipertexto): Código usado para crear una página Web y permitir acceso a documentos en la red.

HTTP (HyperText Transfer Protocol - Protocolo de transferencia de hipertexto): Protocolo usado para indicar que un sitio de Internet es un sitio World Wide Web.

Hipertexto: Sistema para intercambiar información de servidores en Internet utilizando la World Wide Web. El hipertexto consiste en palabras o frases claves.

IEEE: Instituto de Ingenieros eléctricos y electrónicos (USA). Su Comité de Estándares para las Tecnologías Educativas trabaja con el objetivo de desarrollar estándares técnicos, prácticas recomendadas y guías para la implementación informática de sistemas de formación y educación.

Interfaz gráfico de usuario: Interfaz de computador que utiliza iconos o imágenes. Por ejemplo: Macintosh, Windows, y simulaciones gráficas. También llamado GUI por sus siglas en inglés

Internet: Red Internacional en principio usada por el gobierno de Estados Unidos para conectar las redes educacional y de investigación. Actualmente, Internet provee servicios de comunicación y aplicaciones a toda una gama de negocios Internacionales, instituciones educacionales, gobiernos e Institutos de Investigación.

JADE (Java Agent DEvelopment Framework): es una plataforma de software desarrollada en TILab bajo la filosofía de código abierto, que proporciona tanto un entorno de desarrollo como uno de ejecución para la realización y mantenimiento de sistemas multiagente.

Java: Lenguaje de programación portátil desarrollado por Sun Microsystems que requiere de poca memoria. Java no requiere de hardware específico y puede utilizarse sólo o con un documento HTML.

JavaScript: Lenguaje de programación similar a Java pero menos potente que éste. Los comandos de Java Script permiten que las tareas sean completadas por el navegador cuando el usuario accede a la página Web (Por ejemplo la aparición de un cuadro de diálogo cuando el usuario hace clic en una palabra o imagen de la página).

KQML (Knowledge Query and Manipulation Language): Lenguaje de consulta y manipulación del conocimiento.

LGPL (Lesser General Public License): Es una licencia que pretende garantizar su libertad de compartir y modificar el software "libre", esto es para asegurar que el software es libre para todos sus usuarios.

Metadatos: Información sobre el contenido que permite almacenarla y ser recibida desde la base de datos.

Navegador: Software que permite encontrar y visualizar información en Internet. Los más comunes son Internet Explorer y Netscape Navigator.

Objeto de aprendizaje: Unidad reusable de información independiente de los medios. Bloque modular de contenido de teleformación.
ORB (Object Request Broker): En computación distribuida es el nombre que recibe una capa de software (también llamada middleware) que permite a los objetos realizar llamadas a métodos situados en máquinas remotas, a través de una red.

Página Web: Documento en la World Wide Web que es visto a través de un navegador como Internet Explorer o Netscape Navigator.

Protocolo: Conjunto de estándares, normas y formatos para el intercambio de datos que asegura la uniformidad entre computadores y aplicaciones.

RMI (Remote Method Invocation): es el mecanismo ofrecido en Java que permite a un procedimiento (método, clase, aplicación) poder ser invocado remotamente.

Servomecanismos: Sistema electromecánico que se regula por sí mismo al detectar el error o la diferencia entre su propia actuación real y la deseada.

Sistema adaptativo: Es aquel que, basado en el conocimiento, altera automáticamente aspectos de funcionalidad e interacción para lograr acomodar las distintas preferencias y requerimientos de sus distintos usuarios.

Socket: designa un concepto abstracto por el cual dos programas (posiblemente situados en computadoras distintas) pueden intercambiarse cualquier flujo de datos, generalmente de manera fiable y ordenada. Un *socket* queda definido por una dirección IP, un protocolo y un número de puerto.

TICs (Tecnologías de la Información y Comunicación): son las tecnologías que se necesitan para la gestión y transformación de la información, y muy en particular el uso de

computadores y programas que permiten crear, modificar, almacenar, proteger y recuperar esa información.

TCP/IP (Transmission Control Protocol/Internet Protocol): es el lenguaje que rige todas las comunicaciones entre todos los computadores en Internet. TCP/IP es un conjunto de instrucciones que dictan cómo se han de enviar paquetes de información por distintas redes.

URL (Uniform Resource Locator = Localizador uniforme de recursos): Dirección de una página principal en la World Wide Web. Por ejemplo: http://www.ahciet.net

BIBLIOGRAFÍA

[1] ATKINSON, B., et. al., 1995. IBM Intelligent Agents. Unicom Seminar on Agent Software, London, UK, May 25.

[2] BRATMAN, M. E.: *Intentions, Plans, and Practical Reason*. Libro completo. Harvard University Press. 1987.

[3] BELLIFEMINE, F., CAIRE, G., POGGI, A. y RIMASSA, G. *"JADE, a White Paper"*. EXP- Volume3 -n3- September 2003. Disponible en *http://jade.tilab.com*

[4] BAUER B., MÜLLER J. P. y ODELL J., AGENT UML: A Formalism for specifying Multiagent Interaction. Agent-Oriented Software Engineering, 2001 (Springer-Verlag, Berlin): P. 91-103.

[5] BRENNER, W., ZARNEKOW, R. y WITTIG, H. *Intelligent Software Agents*. Springer, 1998.

[6] CARR, B. y GOLDSTEIN, I. Overlays: A theory of modeling for computer arded instruction. (Al Memo 406), Cambrigde, MA: Massachusetts Institute of Technology, Al Laboratory, 1977.

[7] COSTA, M. Sistemas Tutores Inteligentes. www.nce.ufrj.br/ginape/publicacoes/trabalhos/MacarioMaterial/Sti.htm

[8] CHAVEZ, A. y MAES, P. Kasbah: An agent marketplace for buying and selling goods. Proceedings of the First International Conference on the Practical Application of Intelligent Agents and Multi-Agent Systems. London, UK. 1996

[9] DeLoach, S.: *Analysis and Design using MaSE and agentTool*. Actas de conferencia. Proceedings of the 12th Midwest Artificial Intelligence and Cognitive Science Conferece (MAICS). 2001.

[10] FERBER, J.: *Multi-Agent Systems*. Libro completo. Addison-Wesley. 1999.

[11] FIPA: *ACL Message Structure Specification*. http://www.fipa.org

[12] FRANKLIN, S. y GRAESSER, A. Is it an Agent, or just a Program?: A Taxonomy for Autonomous Agents. Proceedings of the Third international Workshop on Agent Theories, Architectures and Languages. Springer-Verlag, 1996.

[13] GONZÁLEZ PEÑA, D. y FERNÁNDEZ GAVILANES, M. Fipa Agent UML. Escuela superior de ingeniería informática. Universidad de Vigo, España. 2005

[14] GÓMEZ-PÉREZ, A. Y BENJAMINS, V. R. Overview of Knowledge Sharing and Reuse Components: Ontologies and Problem-Solving Methods. *Proc. of the IJCAI-99 workshop on Ontologies and Problem-Solving Methods (KRR5)*. 1999. Stockholm, Sweden.

[15] HAYES-ROTH, B., BROWNSTON, L. y GEN, R. V. Multiagent collaboration in directed improvisation. In *Proceedings of the International Conference on Multi-Agent Systems (ICMAS-95),* 1995.

[16] HAYES-ROTH, B. y LARSSON, J. E. Guardian: An Intelligent Autonomous Agent for Medical Monitoring and Diagnosis. IEEE Intelligent Systems 13(1): 58-64 (1998)

[17] HOGG, T. y HUBERMAN, B. A. Controlling Chaos in Distributed System. IEEE Trans. on Systems, Man and Cybernetics 21, 1325-1332 (1991)

[18] IGLESIAS FERNÁNDEZ, C. Á. Fundamentos de los Agentes Inteligentes. Informe Técnico UPM/DIT/GSI 16/97. Departamento de ingeniería de sistemas telemáticos. Universidad Politécnica de Madrid

[19] JENNINGS, N. R., CORERA, J. M. y LARESGOITI, I. Developing Industrial Multiagents Systems. Proceedings of the First International Conference of Multi-Agent Systems (ICMAS-95). Páginas 423-430. 1995

[20] JIMÉNEZ REY, E., GROSSI, M. D. y PERICHINSKY, G. Una Aplicación de la Tecnología de MultiAgentes a los Sistemas Tutores Inteligentes: Enseñanza de Computación en Carreras de Ingeniería. VII Workshop de Investigadores en Ciencias de la Computación WICC 2005. UNRC-REDUNCI. Mayo 13-14. Río Cuarto. 2005.

[21] JUCHEM, M., MELO BASTOS, R. 2001. Engenharia de Sistemas Multiagentes: Uma Investigacão sobre o Estado da Arte. Technical Report Series. Number 014. Faculdade de Informática. PUCRS-Brazil. 2001.

[22] KINNY, D., GEORGEFF, M., y RAO, A.: *A Methodology and Modelling Technique for Systems of BDI Agents*. Informe. 1997

[23] KNAPIK, M. y JOHNSON, J. 1998. Developing Intelligent Agents for Distributed Systems. McGraw Hill.

[24] LJUNGBERG, M. y LUCAS, A. The oasis air-traffic management system. In Proceedings of the Second Pacific Rim International Conference on Artificial Intelligence, PRICAI '92, Seoul, Korea, 1992. http://citeseer.ist.psu.edu/ljungberg92oasis.html

[25] MAES, P. Agents that reduce work and information overload. Communications of the ACM, 37(7). Pages 31-40. 1994

[26] MAES, P. Artificial Life Meets Entertainment: Life like Autonomous Agents. Communications of the ACM, 38 (11). 1995.

[27] NWANA, H. Software Agents: An Overview. Knowledge Engineering Review. 1996

[28] PARUNAK, H. V. D. Manufacturing Experience with the Contract Net. In M. N. Huhns, Editor, *Distributed Artificial Intelligence*, Pitman, London (1987), 285-310.

[29] PEÑA, C. I. Sistemas Multiagente para el Tratamiento de la Información en el WEB: Agentes de Interfaz, Agentes de Información, Agentes de Aprendizaje, Agentes Intermediarios. Universidad Industrial de Santander. Escuela de Ingeniería de Sistemas e Informática. I Congreso Internacional de Ingeniería de Sistemas EISI 30 años. Noviembre de 2000

[30] PEÑA, C. I. Intelligent Agents to Improve Adaptivity in a Web-Based Learning Environment, PhD Thesis, University of Girona, Spain, 2004.

[31] RUSSELL, S. y NORVIG, P. 1997. Inteligencia Artificial un Enfoque Moderno. Prentice Hall.

[32] SHOHAM, Y., *Agent Oriented Programming*, Artificial Intelligence, vol. 60 pp. 51-92, 1993.

[33] SMITH, S.C., CYPHER, A. y SPOHRER, J. "KidSim: Programming Agents Without a Programming Language," *Communications of the ACM*, 37, 7, 55- 67.1994.

[34] URRETAVIZCAYA LOINAZ, M. Sistemas Inteligentes en el ámbito de la Educación. Inteligencia Artificial, Revista Iberoamericana de Inteligencia Artificial. No.12 pp. 5-12. ISSN: 1137-3601. 2001

[35] WOOLDRIDGE, M. y JENNINGS, N. R.: *Intelligent agents: Theory and practice*. The Knowledge Engineering Review, 10(2):115–152, (1995).

[36] JULIAN, V., BOTTI, V. Agentes Inteligentes: el siguiente paso en la Inteligencia Artificial. Dpto. Sistemas Informáticos y computación. Universidad Politécnica de Valencia. Novatita. Mayo – junio 2000. Especial 25 aniversario. http://www.ati.es/novatica/2000/145/vjulia-145.pdf

[37]http://cruzrojaguayas.org/inteligencia/Estructura%20de%20Agentes%20Inteligentes.htm: Estructura de los agentes inteligentes

[38] http://www.ikv.de/products/grasshopper/

[39] http://jade.tilab.com/: Página oficial del Telecomm Italia Lab

[40] http://grasia.fdi.ucm.es: Página oficial de GRASIA (GRupo de Agentes de Software: Ingeniería y Aplicaciones de la Universidad Complutense Madrid)

[41] http://auml.org: Página oficial de AgentUML

[42] http://java.sun.com: Página oficial de java