Expert Systems with Applications 39 (2012) 7546–7554

Contents lists available at SciVerse ScienceDirect

Expert Systems with Applications

journal homepage: www.elsevier.com/locate/eswa

An intelligent tutoring module controlled by BDI agents for an e-learning platform

Fernando A. Mikic Fonte [a,*], Juan C. Burguillo [b], Martín Llamas Nistal [b]

[a] E.T.S.I. Telecomunicacion, c/ Maxwell s/n, Universidade de Vigo, Vigo 36310, Spain
[b] Universidade de Vigo, Vigo, Spain

ARTICLE INFO

Keywords:
BDI
Intelligent tutors
E-learning platforms

ABSTRACT

INES (INtelligent Educational System) is an operative prototype of an e-learning platform. This platform includes several tools and technologies, such as: (i) semantic management of users and contents; (ii) conversational agents to communicate with students in natural language; (iii) BDI-based (Believes, Desires, Intentions) agents, which shape the tutoring module of the system; (iv) an inference engine; and (v) ontologies, to semantically model the users, their activities, and the learning contents. The main contribution of this paper is the intelligent tutoring module of the system. Briefly, the tasks of this module are to recognize each student (checking his/her system credentials) and to obtain information about his/her learning progress. So, it can be able to suggest to each student specific tasks to achieve his/her particular learning objectives, based on several parameters related to the existing learning paths and the student's profile.

1. Introduction

Nowadays the e-learning is at its very peak thanks to the possibilities offered by the communication and information technologies (particularly Internet) and for the needs of our society related to time availability. As a result, a lot of e-learning systems and architectures have been arising and consolidating. Nevertheless, most of them could be improving with several characteristics oriented to allow a better educational experience (both for teachers and students). It is necessary to develop systems which were able to offer specific and personalized contents to the students in an intelligent way, i.e., systems capable of making decisions about which is the more suitable educational content at every time for each student.

INES (INtelligent Educational System) is a functional prototype of an online learning platform, which combines essential capabilities related to e-learning activities. These capabilities are those concerning to Learning Management Systems (LMS) (Grace & Butler, 2005), Learning Content Management Systems (LCMS) (Horton, 2000), and Intelligent Tutoring Systems (ITS) (Murray, 1999).

An LMS is a software application installed in a server, which is used to administer, distribute, and supervise the educational activities of an organization or institution. Its main functions are to manage users, resources, and educational materials and activities, to control the access, to supervise the learning process and progress, to make evaluations, etc. An LMS often does not include authoring capabilities (to develop its own contents), which are carried out by an LCMS.

An LCMS is used to create and manage the content of a part of an educational program (for example a course), which may be used, managed, and personalized in different ways.

At last, an ITS is an educational support system (a kind of virtual tutor), used to help learners in their tasks and to provide them with specific and adapted learning contents.

To carry out all these functionalities, our system, as a whole, comprises a set of different tools and technologies (which we could not find working together in other similar systems), as follows: semantic managing users (administrators, teachers, and students) and contents tools, a chatterbot which is able to communicate with students in natural language (Neves, Diniz, & Barros, 2002), agents based on BDI (Believes, Desires, Intentions) technology (Bratman, 1999), an inference engine based on JESS (a rule engine for the Java platform) (Friedman-Hill, 2000) and an ontology (to model the users, their activities, and the learning contents) (Chandrasekaran, Josephson, & Benjamins, 1999) that contribute with the semantics of the system.

The main contribution of this paper is the presentation of the intelligent tutoring module of our system. Thanks to it, the system not only models its students, but is able to specify for each student what to learn and how to do that, offering personalized formation without human intervention. This module has been developed through the BDI intelligent agents paradigm, and it has an inference engine based on JESS (Java Expert System Shell).

This paper is organized as follows: in the following sections we will briefly present the BDI agents and how they work, and the Intelligent Tutoring Systems. In Section 4, we will address our

* Corresponding author. Tel.: +34 986813461; fax: +34 986812116.
E-mail addresses: mikic@det.uvigo.es (F.A. Mikic Fonte), J.C.Burguillo@det.uvigo.es (J.C. Burguillo), martin@det.uvigo.es (M.L. Nistal).

0957-4174/$ - see front matter © 2012 Elsevier Ltd. All rights reserved.
doi:10.1016/j.eswa.2012.01.161

e-learning platform INES, and their characteristics as LMS, LCMS, and ITS. The Section 5 will be dedicated to explain how the intelligent tutoring module works. At last, we will finish with some conclusions.

2. BDI agents

The specification, design, verification, and application of the BDI agents have been in the spotlight throughout last years. These agents are systems that work in a changing environment, receiving information continually. Besides, these agents have to carry out actions, which can affect the environment, based on their internal (mental) state.

The intelligent agent model is a paradigm inspired by the notion of rational agents based in mental attitudes. Specifically, the BDI model (based on the mental processes of Beliefs, Desires, and Intentions) shows up as a philosophical model for the modeling of the human rational behavior. Later, nevertheless, it was adopted and converted in a model of execution for software agents based on the notion of "beliefs", "goals", and "plans" (concepts which can be created and manipulated by agents).

At the beginning of the decade of 1990, Rao and Georgeff designed a framework in which the notions of Beliefs, Desires, and Intentions were the central part. In concrete, they designed both a logic system where these concepts were included and an architecture for the execution of BDI based programs (Rao & Georgeff, 1995).

Beliefs are related to everything that the agent knows (both related to the environment and to its internal state) and will be stored in the "belief base" of the agent. Goals allow describing what the agent must to achieve, but they do not include information about the actions the agents must exactly carry out to do that. Finally, plans are composed by a set of instructions which allow the agent to carry out several actions to try achieving the previously set goals. That is, if the current situation (that the agent knows) is not the desired one (i.e., the situation specified by the goals) then the agent will execute the plan or plans to achieve this desired situation. The relation between goals and plans will be carried out by a reasoning engine, which will decide what plan will be executed to try satisfying a specific goal.

The use of the words "to try" is not casual, because the achievement of the goal is not guaranteed. This is so because the BDI agent is designed to reason under certain conditions, which can change while the plan is being executed (or while the reasoning engine is taking a decision). Taking this in account, it is not out of mind to find alternative plans.

This representation of the behavior using mental notions has several benefits, such as to avoid the low level abstraction (because only specific goals to achieve and several plans to do that are presented) and easily understanding the autonomous behavior of the agent (even its prediction).

2.1. BDI platforms

Inspired by the design of Rao and Georgeff, several programming platforms have arisen. Jadex (Braubach, 2009) and JACK (Howden, Ronnquist, Hodgson, & Lucas, 2001) can be considered as two of the most important ones.

JACK Intelligent Agents allows creating, executing, and integrating commercial multi-agents systems based on Java (Gosling, Joy, Steele & Bracha, 2000). It includes three main extensions of Java:

- A set of syntactic extensions of the language.
- A compiler which converts those syntactic extensions of Java classes.
- A set of classes (kernel) to allow supporting for code execution.

The main objectives of JACK are: (i) to offer a robust, stable, and light product; (ii) to satisfy a wide variety of practical needs; and (iii) to facilitate the transfers between the developers and the industry (developing agents that can be components of a bigger system).

The Jadex project, the solution adopted for our platform (see Section 2.2), is been developed by the Distributed Systems and Information Systems group of the University of Hamburg. Jadex is a kind of reasoning engine following the BDI agents model that facilitate the construction of intelligent agents, allowing to use XML and Java to make these type of agents, and it is described in detail in the next section.

2.2. Jadex

As we mentioned above, Jadex is the solution adopted for our platform. Nowadays, this reasoning engine has several tools which can be grouped in three sets:

- A Java API and a set of predefined functionalities to make agents programming easy.
- A platform which allows the execution of these agents (reasoning engine).
- A set of tools to use in execution time which allow managing and monitoring different aspects of these agents.

Agents implemented in Jadex are known as Goal Oriented Agents. This paradigm allows including an abstraction level to the definition of the behavior of the agent, and so the developer can specify some goals to be achieved by the agent, without saying how to do it. Now the agent itself has certain degree of freedom to decide the more appropriate form to achieve those goals.

Jadex takes into account the concepts of beliefs, objectives, and plans, which can be created and managed by the agents. As for the BDI model in general, the beliefs will be any kind of Java object, which will be stored in the belief base of each agent; the objectives will allow to describe what an agent must to reach (not including information about how to reach those objectives); at last, plans will be a set of Java instructions which will allow an agent to do several actions to reach the previous described objectives. A reasoner will set the relations among objectives and plans to decide what plan will be executed at each moment.

The Jadex platform offers a complete set of execution tools to allow managing, testing, and debugging agents. The base of these tools is the Jadex Control Center, (Fig. 1) which acts as the access point for the rest of the tools (plugins).

Finally, Jadex is trying to be compliant with some standards that facilitate the interoperability among systems. For example, we can find one of the standards presents in Jadex in the interchange of messages among agents, which are adapted to the FIPA standard (FIPA Standard Specifications., 2002). Besides, Jadex provides certain predefined funtionalities through the called "capabilities", which allow to be reused in different agents.

All of these characteristics of Jadex are enough reasons to choose it for our system. In short, Jadex offers us a simple form to develop an agent-based system without loosing the benefits of the intelligent agents paradigm, supporting both middleware and reasoning, and easy to use reasoning capabilities (by exploiting the BDI model combining it with XML and Java).

3. Intelligent Tutoring Systems

Intelligent Tutoring Systems (Corbett, Koedinger, & Anderson, 1997) are designed as intelligent tutors based on knowledge, to serve as a guide in the student learning process. They are programs that aim at improving the learning process through personalization

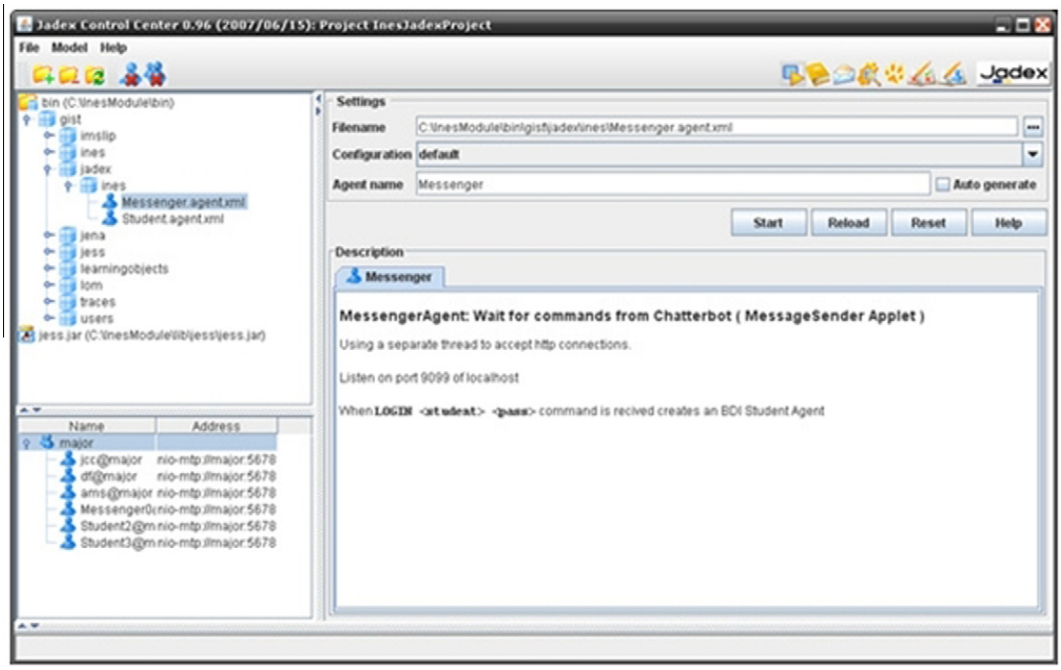

Fig. 1. Jadex control center.

of contents depending on the student's skills and knowledge about the topics that they are learning about. Based on these premises, ITSs can adapt the educational content or the style used to teach. They try to emulate the way in which a human tutor guides his/her students throughout the learning process.

ITSs have their origin in the field of Computer Assisted Instruction (CAI) (Jenks & Springer, 2005), whose first systems began to develop in the fifties and sixties. It was not until the eighties when ITSs started to develop, and to difference from CAI (Jill & Larkin, 1992). Moving to the nineties, the ITSs evolved from an educational proposal to the design of environments for the discovery of knowledge and the experimentation. In spite of these advances, ITSs have not been used by the general population, in part due to its design complexity, which has limited their practical application. We can mention some examples of systems, architectures, and methodologies related to ITSs, such as:

- ELM-ART (ELM Adaptative Remote Tutor) (Brusilovsky, Schwarz, & Weber, 1996): It is a web-based ITS, which we can see as a kind of intelligent online text book with an integrated problem resolution tool. It has all the material that belongs to a considered course (presentation of new concepts, tests, examples, problems, etc.). This system is able to recognize the material presented by the student and to help him/her through the learning process.
- AutoTutor (Graesser, Chipman, Haynes, & Olney, 2005): It is a system that emulates a human tutor maintaining a conversation with the students in natural language, presenting several questions or problems to them. The system has, for each problem, a list with the right answers and another one with possible wrong answers. The main objective of the system is to train the students to answer the proposed questions correctly. Other objectives are to correct mistakes committed by the students during the conversation, and to give them feedback (positive, neutral, and negative one) about it.
- KnowledgeTree (Brusilovsky, 2004): It is a distributed architecture for adaptive e-learning based on the reutilization of intelligent educational activities. This architecture has at least four kinds of servers (activity, services, learning portals, and student models ones). The students are able to work with all the learning tools and the educational contents through a central service provided by a learning portal.
- MEDEA (Methodology and Tools for the Development of Intelligent Environments for Teaching and Learning) (Trella, 2006): It is a methodology to develop intelligent learning environments for the web based in the integration of educational systems. The development of the systems has two levels: (i) first, it is defined a development methodology in which the elements and actors involved in the construction process of the system are identified; and (ii) finally a framework adjusted to the defined methodology is built.

Concerning the use of BDI technology in ITSs, we can see in Giraffa, Móra, and Zamberlan (2001) an ITS with a student model based on mental states through the use of BDI agents, and the MCOE (Giraffa, Móra, & Vicari, 1998), another project where the student and the tutor are modeled with cognitive agents within the BDI architecture. Another example of these systems is ABITS (Mowlds, Roche, & Mangina, 2005), which instantiates rule-enhanced BDI agents to identify personal preferences, learner type, and learning style (an individualized student model is used in conjunction with an agent who is assigned to each student and assists the student in achieving his/her goals). At last, we can mention REAL (Bai & Black, 2006), a framework that includes a simulated gaming environment, using a multi-agent system comprising a reflective agent, pedagogical agent, expert agent, and communication agent to model the human roles of user, teacher, domain expert, and a coordinator.

Our background in expert systems dates back to several years ago. It is mainly based in the development of X-Learn (Burguillo-Rial & Vázquez, 2004) and our work about case-based reasoning and modeling. X-Learn is an e-learning environment designed to simplify the authoring and visualization of multimedia courses, which includes several intelligent assistants to help the teacher and the student during the learning process. Related to case-based reasoning and modeling we proposed a student model structured

as a multi-agent system (González, Burguillo, & Llamas, 2005) to describe how a multi-agent intelligent learning environment can provide adaptive tutoring based in case-based student modeling (to illustrate the process of student modeling and to validate the student model an algorithm and a case study based on an intelligent tutoring system for learning in public health domain were presented). Finally, we also designed a case-based peer-to-peer multi-agent system for collaborative management of student models in ITSs (Burguillo, González, Llamas, & Mikic, 2008), whose goal is twofold: first, to initialize the student model when a new student logs on the tutor system and second, to update the student model depending on the student's interaction with the system and exchanging this information with its peers.

All this work has led us to the development of an intelligent e-learning platform. In front of the mentioned systems, more specific and mainly centred in the student model and preferences, our system design based on BDI-agents has a general purpose and it can be adopted for a complete e-learning experience, as we can see in the next sections. Briefly, we can mention now that our system not only models the students, but all the users of the system and its educational contents in a semantic way through the use of ontologies, communicating with students in natural language, and benefits from the use of BDI-agents to take appropriate decisions at every moment. Thus, we can say that our system as a whole is able to develop a complete educational experience, with a BDI-based virtual and personalized tutor for every student, which is able to help (without human intervention) this student to pass any kind of courses defined by his/her teachers through our creation tool provided (these courses may be simple or as complete as the teachers want by the use of rich multimedia contents).

4. E-learning platform: INES

INES is a functional prototype of an online learning platform, which combines capabilities concerning to LMSs, LCMSs, and ITSs. That is to say, INES is able to carry out specific tasks concerning to these three types of systems, such as:

- Management of students, administrators, resources, activities, access, evaluations, etc.
- Creation, management, and distribution of educational contents.
- Tutoring, helping, and guiding the student.The main parts of INES can be grouped into several blocks (Fig. 2).

- Ontology (Mikic, Burguillo, Llamas, & Fernández, 2009a): There is an ontology divided into three sub-ontologies: (i) one to semantically define the contents of the courses (learning objects), which is based on LOM (Hodgins & Duval, 2002); (ii) another one, based on IMS LIP (Norton & Treviranus, 2005), to model users' data; and (iii) the last one to define relations among users and learning contents.
- Contents and users managing module: This module allows administrators to manage both the users of the system and the contents of the courses. Besides, it allows the BDI agent and the inference engine gain access to the ontology.
- Inference engine: It processes the requests of the BDI agent and decides what will be allowed to do and what not.
- BDI agents: The actual brain of the system. They are based on BDI technology, and they are responsible of taking personalized decisions about the learning of every student in an intelligent manner (see Section 5).
- Chatterbot: Responsible for the communication with the students (Mikic, Burguillo, Llamas, Rodríguez, & Rodríguez, 2009b) (based on (Mikic, Burguillo, & Llamas, 2010)).

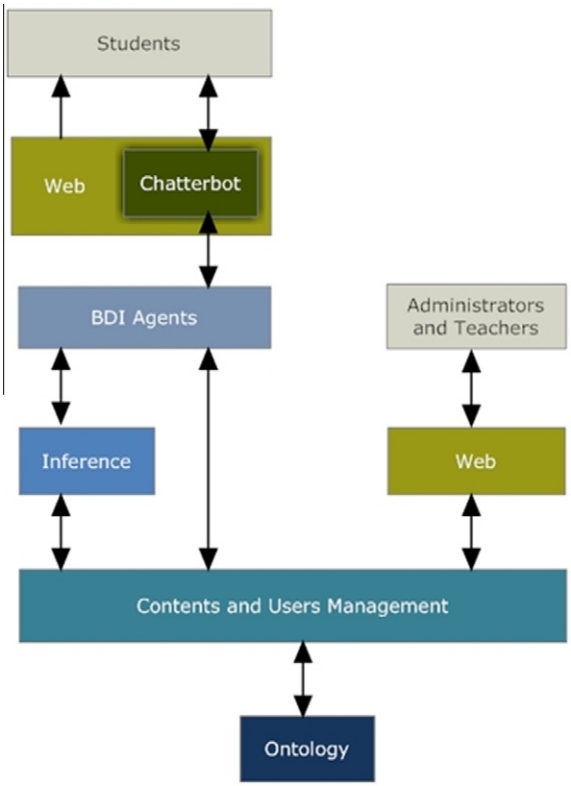

Fig. 2. Block diagram of INES.

4.1. Functionalities as LMS

An LMS can be defined as a software application based on Internet, used to design, implement, and evaluate a specific learning process, that is to say, it includes all the needed services for the correct online managing of learning activities. In particular, it tries to offer managing functionalities to the users of the platform: system administrators, teachers, and students.

The services an LMS offers may be different according to the platforms, and in our case, INES offers among others:

- Administration and management of courses (registration, directory of users, consult of qualifications, etc.).
- Distribution of contents.
- Collaborative work tools.
- Control and assessment of users.
- Design of personalized learning plans.

In short, our platform offers to the instructors, as an LMS, functionalities to distribute learning contents, to control the participation of the students, and to evaluate them.

4.2. Functionalities as LCMS

An LCMS is used to design, create, and manage the contents of a learning program (usually, contents belong to courses). As an LCMS, INES provides functionalities such as:

- Simplifying the creation and management of content, so experts in the matter to teach can carry out these tasks with a minimum effort (even if they do not have experience with technology).
- Several types of learning material can be used (from simple text to rich multimedia contents).
- Contents are stored in a repository.

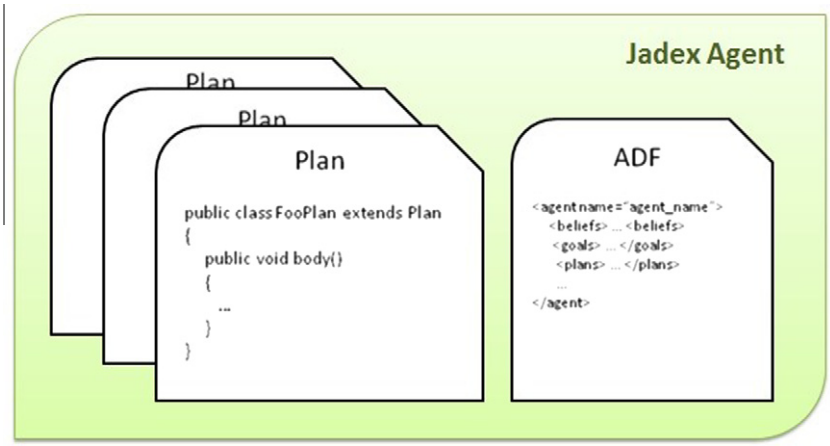

Fig. 3. Jadex agent.

- Learning contents can be reused and exported (even to different systems because the research of a LOM-based ontology).

So, functionalities related to an LCMS and an LMS are combined to carry out a complete management of the learning, from the point of view of the users (students, teachers, and administrators) and the contents.

4.3. Functionalities as ITS

ITSs are educational computer systems able to specify what to teach and how to do that, i.e., they are a kind of virtual intelligent tutors based on knowledge, which guide the students in their learning process, trying to emulate a human tutor. In this sense, INES makes inferences over the knowledge a student has about certain topics or tasks, and so it offers him/her suitable learning contents. Besides, it is in charge to do a personalized assessment over each student and, taking this into account, it can recommend specific learning tasks to him/her.

5. Intelligent tutoring module

The use of a virtual tutor allows INES to specify to each student what to learn and how to do that, offering him/her a personalized education in a direct form. To carry out all of this, it will be done inferences over the knowledge a student has about the contents to be learnt or about a task to be done, and so the educational activity or the style of learning can be adapted and personalized. The main objective of this module is to offer to each student an individualized and more appropriated sequence of knowledge units and tasks, i.e., helping him/her to find the optimum path through the educational material.

We can see this module as the actual brain of the system. It carries out the tasks of a personal intelligent virtual tutor while a student is following a course. The main part of this module in an intelligent agent (ISMAEL, Intelligent System Manager Agent for E-Learning), and it make several tasks, such as: recognizing each student, checking his/her credentials when he/she logins into the system, and obtaining information related to his/her educational progress. So, the virtual tutor can suggest the student several tasks to reach specific educational objectives. By the way, it will be the own student which will decide if he/she accepts the proposed tasks or asks for other ones trough the interaction with ISMAEL. These other asked tasks can be satisfied or not by ISMAEL, depending on the decisions the agent make about the education of the considered student (these decisions will be based on several parameters related to the proposed learning paths of the course and the profile of each student).

This module has been developed through the BDI intelligent agents paradigm, and its agents (EMMA and ISMAEL, see Section 5.1 and 5.2 respectively) run over the Jadex platform. To create these agents, Jadex offers a Java API and a set of generic functionalities. This API offers access to plans programming concepts, which are Java classes that inherit from an abstract class several methods to send messages, wait for events, launch sub-objectives, etc. Besides these Java plans, an ADF (Agent Definition File) file must be created. This file will be an XML one, which will include the agent definition, where beliefs, plans, and initial objectives of the agent are shown (Fig. 3). The execution engine of Jadex will read this file to create an instance of the agent, and it will run it keeping a record of the objectives while selecting the plans to execute (this selection is based on internal events and messages of other agents).

Every time the system is active, the Jadex platform will be running, and so, when a student is accessing into the system a new instance of ISMAEL will be created under demand, which acts as a virtual tutor for the student.

The communication with the students is carried out by a chatterbot (CHARLIE – CHAtteR Learning Interface Entity), which acts as an interface between each student and the system. Specifically, CHARLIE will contact a messenger agent (EMMA – Events and Messages Manager Agent) who will be the intermediary between it and ISMAEL (Fig. 4).

5.1. EMMA agent

The objective of this agent is acting as a kind of messenger or intermediary between the chatterbot and the BDI agent ISMAEL. It is in charge of managing the events and messages that can arise between them (this agent has been implemented as a BDI agent too).

When the system start up, an instance of EMMA will be created, which will be running waiting connections from the chatterbot. When a connection arises, it will be created a thread which will be in charge of reading the received messages. If a login message is received, then a student wants to log into the system, and so this messenger agent will launch an instance of ISMAEL (obviously, when a student log off the system, the messenger agent will destroy the corresponding instance of ISMAEL previously created).

When the connection between the chatterbot and the new instance of ISMAEL is on, EMMA will format the messages from the first one to a format that the second one can understand, and then

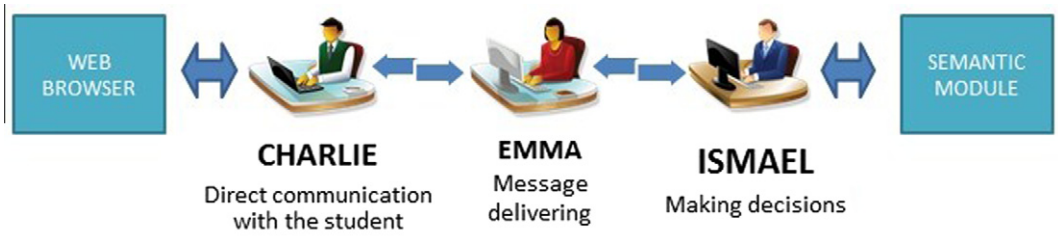

Fig. 4. Behavioral schema of CHARLIE, EMMA, and ISMAEL.

it will send these messages to this last one. Likewise, when ISMAEL wants to communicate with the chatterbot, the inverse process will take place (Fig. 5).

5.2. ISMAEL agent

As we mentioned before, ISMAEL is a BDI agent which acts as the heart of the intelligent tutoring module of the system. In short, its tasks are: (i) to recognize each student, checking his/her credentials when he/she tries to log into the system; and (ii) to obtain information related to his/her educational progress, and so to suggest him/her personalized tasks to achieve a specific learning goal.

Following, we will briefly present some of its more important BDI characteristics:

- Beliefs: They will be mainly compounded by a series of references to: instances to access the ontology module, instance of the messenger agent to communicate with the chatterbot, identifier of the student who is being tutored, identifiers of the tasks the student is doing, temporizations, etc. In short, they are a whole set of parameters (information) the agent need to take decisions related to the goals to achieve at every moment, and the actions which will be carried out to try reach those goals.
- Desires: They are the goals. They consist of making the student does a series of educational activities, which have previously defined by a teacher of the platform. Usually these activities will be included in a semantic schema which defines the relations among several learning objects. These relations will make a learning path, which the student must follow. This path, obviously, will not have to be linear, but in our prototype it will have a structure divided into Courses, Modules, Units, and Elements (Fig. 6). The main objective will be that the student achieves certain kind of competencies related to the taught matter. To do this, this objective will be divided into others sub-objectives of low level, and so on, recursively. So, when these sub-objectives are achieved, then the high level ones will be achieved too (and so on until achieving the main initial objective).
- Intentions: These intentions are the plans the agent owns to try the student reaches the set out goals. So, the agent will execute a series of plans (actions) taking into account at every time the beliefs which it has, the goals to achieve at this moment, and the messages that can be received from the student. Besides, a series of plans to do routine tasks are included, such as: tasks to process messages (both the received ones and the desired to send), tasks to manage the temporization, etc.

Talking about temporization, only to point out the existence of timers, which will be used to detect the presence of a student, when the system detects inactivity periods. There are two mainly reasons to check this situation:

- To detect if the user left the system: If the period of inactivity is considerable (it exceeds certain limits) the system will assume that the student has left his/her task without a communication about it. So, the system will try to communicate with the student, and if he/she does not answer, it will proceed to shut down the session of this student (to save resources).
- To monitor the time that the students spend with every educational task: The semantic knowledge about the learning objects includes information about the time a student would spend with them. So, the actual spent time to finish a task is really important to know if this expected time is properly set. Besides, the spent time by a student to do his/her tasks is valuable data to take into account in his/her learning profile.

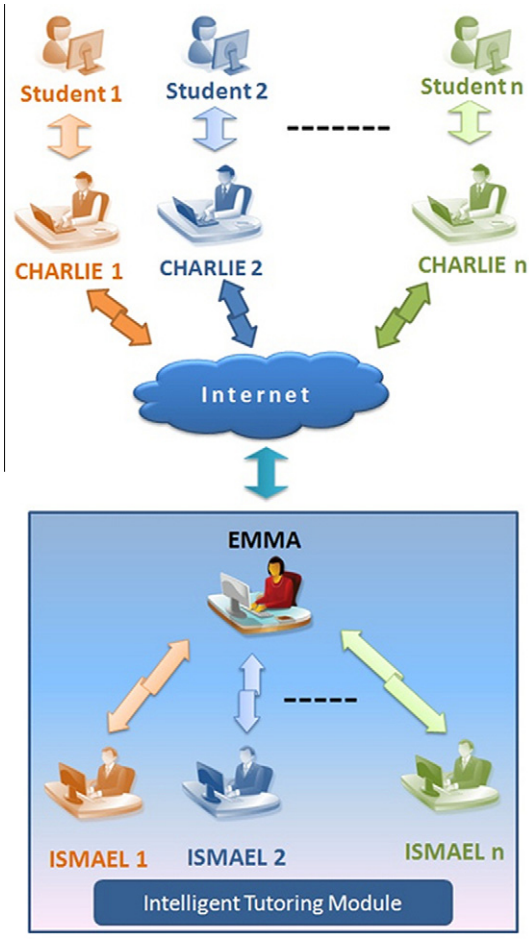

Fig. 5. Working schema of CHARLIE, EMMA, and ISMAEL.

5.3. Flow diagram of the intelligent tutoring module

Now we are going to briefly comment the behavior of this module when the student is already doing a learning task (Fig. 7):

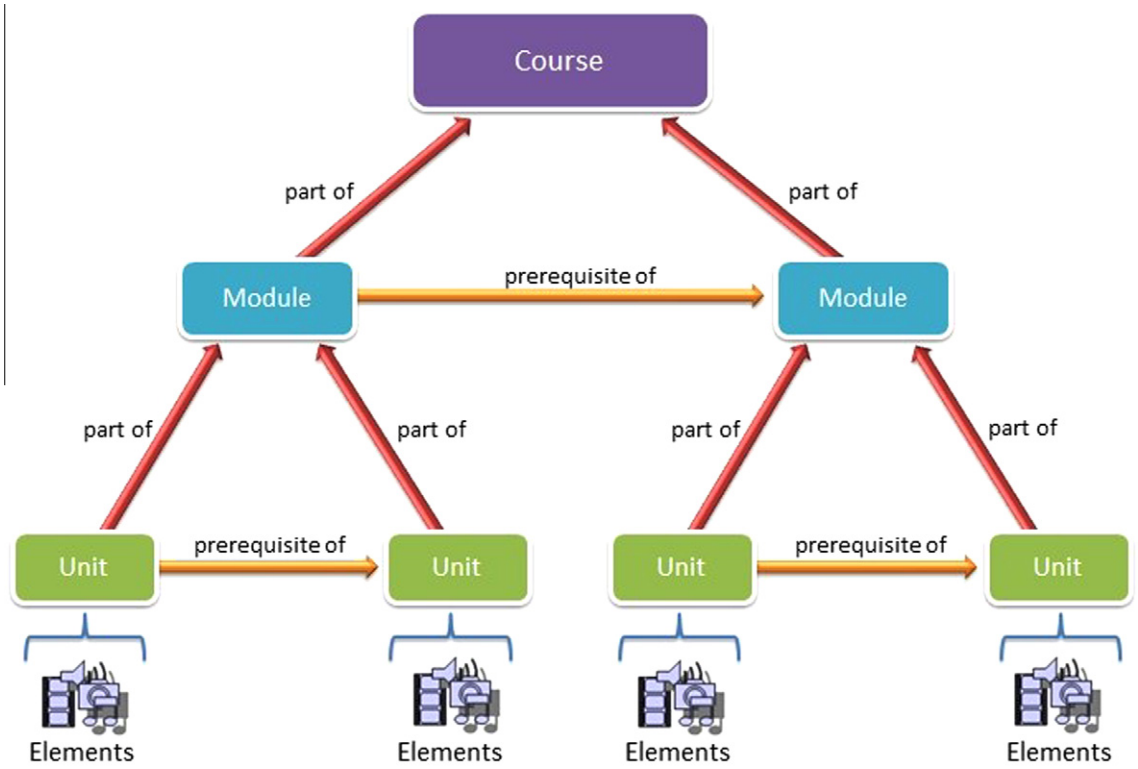

Fig. 6. Basic structure of a course.

- When the user sends a message to the chatterbot, it processes it to know if it needs to send this message to the BDI agent, or if it is able (through its knowledge base) to answer the student by itself.
- If the chatterbot needs to send the message (because this message is about the user has finished his/her task or because the user requests something from the system), CHARLIE sends it to EMMA (in other case, the chatterbot itself will answer the student).
- EMMA formats the message before sending it to ISMAEL (so this one is able to understand it).
- ISMAEL checks if the user has finished his/her task, achieving the assigned goal.
- If the user finished his/her task: (i) the system will be updated (beliefs and ontology); (ii) the next goal will be selected; (iii) the plans to achieve this new goal will be launched; and (iv) the user will be informed about this (through EMMA and CHARLIE).
- If the user did not finish his/her task, the system processes the request of the user and: (i) if the request is rejected, the system sends to the user an appropriate message about it; or (ii) if the request is accepted, the system is updated.

At last, we must point out that the selection of the next goal to achieve is a process which implies taking into account both the learning path and the current situation of the student related to his/her learning tasks. ISMAEL is able to access this knowledge through the inference engine module of the system (see Section 5.4). For example, the next selected goal could be one of the follows:

- To do the next task of the student's learning path (this learning path has been previously defined by the teacher who created the course the student is carrying out).
- To recommend to the student a review of specific content which he/she has already seen before (this recommendation could be based on the results of a test which he/she had done to evaluate his/her knowledge, on the amount of time that has passed from the student see that specific content, etc.).
- To show the student additional educational content (either for a student's request or for a decision of the system).

5.4. Inference engine module

The inference engine module will be used to guide the students throughout their learning tasks, offering them appropriated educational material at every moment. So, its main functionality is to analyze the structure of the defined learning paths (which are implicitly stored into de ontology) and, through a set of rules, to offer to the virtual tutor information about which could be the next learning object to show to each student (or even if a specific educational object can be shown to a student or not).

This module is implemented as a set of Java classes which use the JESS (Java Expert System Shell) inference engine.

5.4.1. Jess

JESS is a tool for the definition and development of Expert Systems, which tries to emulate certain aspects of the human reasoning. The main characteristic of these systems is that they are based on rules, i.e., they have a predefined knowledge to make decisions. They have a Knowledge Base and an Inference Engine, where the Knowledge Base has system rules designed for a specific scope and the Inference Engine combines "facts" and "questions" to obtain a solution. To do this, they use rules defined into the Knowledge Base and selects data from a working memory.

JESS is developed by Ernest Friedman-Hill from Sandia National Laboratories, and it is a portable, extensible, and fast reasoning engine, which is capable of an easy integration into Java applications.

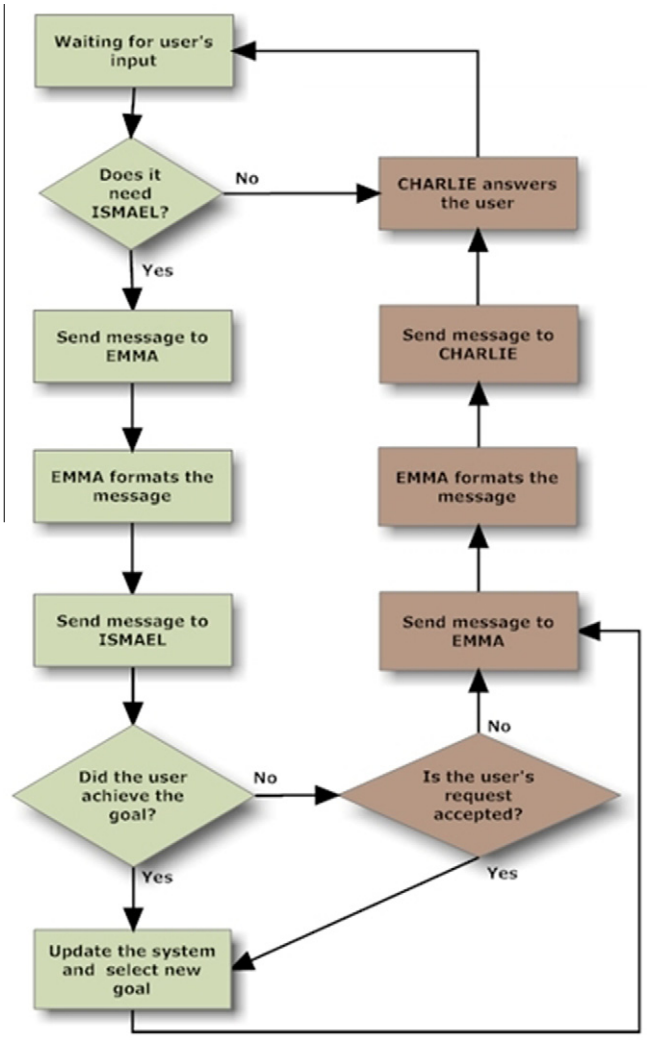

Fig. 7. Flow diagram of the intelligent tutoring module.

To process the facts, JESS uses Forward Chaining, which is based on getting facts from other input ones.

5.4.2. Integration and processing

To integrate the inference engine in our system it was necessary to develop a set of Java classes that use the API of JESS, which have been grouped into a package. These classes are used to: (i) allow to store information about the educational objects, temporarily storing data from the ontology; and (ii) call the JESS inference engine, apply appropriate rules, and return the results.

The process of making a call to the inference engine will be as follows (Fig. 8):

(1) An instance of the class for making the call must be created. Some parameters must be passed to this instance, such as: (i) a reference to a class for accessing the ontology; (ii) an identificator of the learning object we want to access; and (iii) the identificator of the student for whom the process it is carrying out.
(2) The templates, the rules, and the facts will be loaded (see Section 5.4.3). The templates allow to define facts, so the inference engine will take into account the information offered by these templates, and will be able to apply over these facts the appropriate rules.
(3) When all the facts are assigned to the inference engine, they will be engaged with the rules and the functions (process known as "executing the evaluation of the inference engine").
(4) At last, the evaluation of the facts within the system is carried out (in the working memory), and other new ones will be created by the application of rules and functions, so the requested information is obtained.

In short, the desired functionality will be the search of the next educational activity to offer to a specific student, or to know if a requested learning object can be showed to the student or not.

5.4.3. Templates and rules

Templates allow to define facts, so the inference engine takes into account the information they offer and it applies over them the appropriate rules. The templates implemented in our prototype have the following statements:

- An educational object is a Course, a Module, an Unit, or an Element (these are all the types of educational objects our system works with).
- An educational object has been passed.
- An educational object is "part of" or "prerequisite of" another one.
- It is a specific number of days (threshold days) now since an educational object has been passed.
- An educational object is candidate to be the next one to be shown to the students.
- An educational object must be deleted from the list of candidates.
- In the other hand, rules allow to engage facts to infer new information. The rules implemented in our prototype have the following statements.
- If the educational object "idx" is prerequisite of the educational object "idy", and "idy" is prerequisite of the educational object "idz", then "idx" is prerequisite of "idz".
- If the educational object "idx" is prerequisite of the educational object "idy", and "idz" is part of the educational object "idy", then "idx" is prerequisite of "idz".
- If the educational object "idx" is prerequisite of the educational object "idy", and "idz" is part of "idx", then "idz" is prerequisite of "idy".
- If the educational object "idx" is a Unit, it has not been passed, and it is part of the educational object "idy", then "idx" is candidate to be the next educational object to be showed to the student.

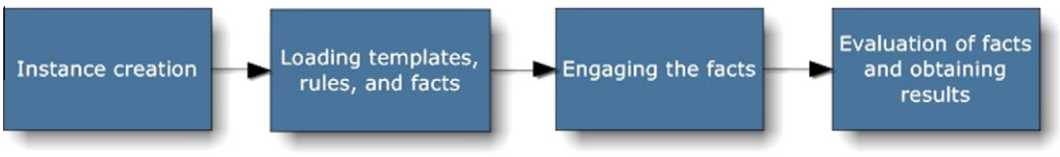

Fig. 8. Process of making a call to the inference engine.

- If the educational object "idx" is a Unit, it has been passed more than the threshold days ago, it is part of the educational object "idy", and "idy" has not been passed, then "idx" is candidate.
- If the educational objects "idx" and "idy" are candidates and "idx" is prerequisite of "idy", then has a preference over "idy" to be candidate.
- If the educational objects "idx" is not yet a candidate it will be deleted from the list of candidates.

6. Conclusions and future work

The main contribution of this paper is the presentation of a functional prototype of an intelligent learning platform (called INES), which includes capabilities related to LMSs, LCMSs, and ITSs, such as: management of contents and users, and recommendation of specific and suitable learning tasks to the students. More specifically, we have addressed the intelligent tutoring module of INES, and its BDI agents over the platform Jadex (ISMAEL, which acts as the actual heart of the system, and EMMA, which acts as a messenger between ISMAEL and a conversational agent to communicate with the students). The main task of the tutoring module is to decide at every moment what educational content is the most appropriate to offer to the students, and if some requests from the students are accepted or rejected. To make these decisions, the virtual tutor will take into account the information from several sources, such as the defined learning paths, the situation of each student, the last time a student has used the system, etc. At last, we have pointed out the use of an inference engine based on JESS to analyze the structure of the learning paths and, through a set of rules, to offer valuable information to the virtual tutor. So, we offer a general purpose system design and development, capable to adapt itself to specific needs of students and teachers, using intelligent support to provide reasoned decisions at every moment.

The main future work related to the tutoring module is to improve its intelligence. The functional prototype developed has a basic intelligence and it is able to carry out the tasks mentioned in this paper for purposes of checking its viability. Nevertheless, it is essential the improvement of its capabilities to deal with a lot of situations which can arise in a real scenario.

Acknowledgements

The authors want to thank Spanish Ministerio de Ciencia e Innovacion for its partial support to this work under grant "Methodologies, Architectures and Standards for adaptive and accessible e-learning (Adapt2Learn)" (TIN2010-21735-C02-01). Also, we want to thank to David Fernández Hermida for his partial support of this work.

References

Bai, X., Black, J. (2006). Real: an agent-based learning environment. In *Proceedings of the Agent-based Systems for Human Learning (ABSHL) Workshop at AAMAS-2006* (pp. 13–17). Hakodate, Japan.

Bratman, M. E. (1999). *Intention, plans, and practical reason*. CSLI Publications.

Braubach, L. (2009). *Jadex, BDI Agent System. Distributed Systems and Information Systems Group, University of Hamburg*. Available at <http://jadex.informatik.uni-hamburg.de/xwiki/bin/view/About/Overview>. Accessed on 04/2011.

Brusilovsky, P. (2004). KnowledgeTree: A distributed architecture for adaptive e-learning. In *Proceedings of the 13th international World Wide Web conference on Alternate track papers & posters, WWW Alt. '04* (pp. 104–113). New York, USA.

Brusilovsky, P. Schwarz, E., Weber, G. (1996). ELM-ART: An intelligent tutoring system on World Wide Web. In *Intelligent Tutoring Systems* (Vol. 1086, pp. 261–269). Springer Verlag.

Burguillo, J. C., González, C., Llamas, M., Mikic, F. A. (2008). A case-based peer-to-peer framework for managing student models in intelligent tutoring systems. In *38th ASEE/IEEE Frontiers in Education Conference – FIE 2008. Saratoga Springs, New York, USA*.

Burguillo-Rial, J. C., Vázquez, E. (2004). X-Learn: an Intelligent Educational System oriented towards the Net. Current Topics in Artificial Intelligence. Lecture Notes in Artificial Intelligence (LNAI), 3040 (pp. 628–637). Springer-Verlag. X Conferencia de la Asociación Española para la Inteligencia Artificial (CAEPIA04), San Sebastián, Spain.

Chandrasekaran, B., Josephson, J. R., & Benjamins, V. R. (1999). What are ontologies and why do we need them. *IEEE Intelligent Systems, 14*(1), 20–26.

Corbett, A. T., Koedinger K., Anderson, J. R. (1997). Intelligent Tutoring Systems. In M. Helander, T. K. Landauer, P. Prabhu (Eds.), *Handbook of Human-Computer Interaction*, Completely Revised in Second Edition, Elsevier Science B. V. [Chapter 37].

FIPA Standard Specifications. (2002). Available at <http://www.fipa.org/repository/standardspecs.html>. Accessed on 04/2011.

Friedman-Hill, E. J. (2000). Jess, The Java Expert System Shell. Available at <http://web.njit.edu/all_topics/Prog_Lang_Docs/html/jess/>. Accessed on 04/2011.

Giraffa, L. M. M, Móra, M., Vicari, R. M. (1998). Modelling the MCOE tutor using a computational model. In *Advances in Artificial Intelligence, Lecture Notes in Computer Science*, Vol. 1515/1998 (pp. 135–142). Springer Verlag.

Giraffa, L. M. M., Móra, M., Zamberlan, A. (2001). Working on student models (really) based on mental states. In *Proceedings of the Fourth International Conference on Computational Intelligence and Multimedia Applications, ICCIMA 2001* (pp. 379–383).

González, C., Burguillo, J. C., Llamas, M. (2005). Case-based Student Modelling in a Multi-agent Learning Environment. Lecture Notes in Artificial Intelligence, LNCS 3690 (pp. 72–81). Springer-Verlag. 4th International Central and Eastern European Conference on Multi-Agent Systems (CEEMAS05). Multi-Agent Systems and Applications IV. Budapest, Hungary.

Gosling, J., Joy, B., Steele, G., & Bracha, G. (2000). *The Java language specification*. Addison Wesley Publishing Company.

Grace, A., & Butler, T. (2005). *Beyond knowledge management: Introducing learning management systems*. Idea Group Publishing.

Graesser, A. C., Chipman, P., Haynes, B. C., & Olney, A. (2005). AutoTutor: An intelligent tutoring system with mixed-initiative dialogue. *IEEE Transactions on Education*, 612–618. 4-48.

Hodgins, W., Duval, E. (2002). Learning Object Metadata Standard, IEEE 1484.12.1-2002.

Horton, W. K. (2000). *Designing web-based training: How to teach anyone anything anywhere anytime*. John Wiley and Sons.

Howden, N., Ronnquist, R., Hodgson, A., Lucas, A. (2001). JACK intelligent agents. Summary of an agent infrastructure. In *Proceedings of the 5th International Conference on Autonomous Agents*.

Jenks, M. S., & Springer, J. M. (2005). A view of the research on the efficacy of cai. *Electronic Journal for the Integration of Technology in Education, 1*(1), 43–58.

Jill, H., & Larkin, R. W. C. (1992). *Computer-assisted instruction and intelligent tutoring systems: Shared goals and complementary approaches*. Lawrence Erlbaum Associates.

Mikic, F. A., Burguillo, J. C., Llamas M. (2010). TQ-Bot: An AIML-based Tutor and Evaluator Bot. Journal of Universal Computer Science (JUCS), 15(7), 1486–1495. Verlag der Technischen Universität Graz, Austria.

Mikic, F. A., Burguillo, J. C., Llamas, M., Fernández, D. (2009a). Using Semantics in INES, an Intelligent Educational System. In *Proceedings of the 39th Frontiers in Education Conference – FIE 2009. San Antonio, Texas, USA*.

Mikic, F. A., Burguillo, J. C., Llamas, M., Rodríguez, D. A., Rodríguez, E. (2009b). CHARLIE: An AIML-based chatterbot which works as an interface among INES and humans. In *Proceedings of the 20th European Association for Education in Electrical and Information Engineering – EAEEIE 2009. Valencia, Spain*.

Mowlds, F., Roche, B. J., & Mangina, E. (2005). ABITS: learning more about students through intelligent educational software. *Campus-Wide Information Systems, 22*(3), 131–139.

Murray, T. (1999). Authoring Intelligent Tutoring Systems: An analysis of the state of the art. *International Journal of Artificial Intelligence in Education, 10*, 98–129.

Neves, A. M. M., Diniz, I., Barros, F. A. (2002). Natural Language Communication via AIML plus chatterbots. In *Proceedings of the V Symposium on Human Factors in Computers Systems (IHC 2002)*.

Norton, M., Treviranus, J. (2005). IMS Learner Information Package. IMS Technical Report. Available at http://www.imsglobal.org/profiles/. Accessed on 04/2011.

Rao, A. S., Georgeff, M. P. (1995). BDI agents: From theory to practice. In *Proceedings of the 1st International Conference on Multi-Agent Systems, ICMAS-95* (pp. 312–319). San Francisco, USA.

Trella, M. (2006). MEDEA: MEtodologias y herramientas para el Desarrollo de entornos inteligentes de Enseñanza y Aprendizaje. Inteligencia Artificial, Revista Iberoamericana de IA, 32–10 (pp. 77–80).