Combining Agents and Ontologies for Building an Intelligent Tutoring System

Stamatis Panagiotis[1], Panagiotopoulos Ioannis[1], Goumopoulos Christos[1,2], Kameas Achilles[1]
[1]*Educational Content, Methodology and Technology Laboratory (e-CoMeT Lab), Hellenic Open University, Patras, Greece*
[2]*Information and Communication Systems Engineering Department, Aegean University, Greece*
panagiotis.stamatis@gmail.com, gpanagiotopoulos@ecomet.eap.gr, goumop@tutors.eap.gr, kameas@eap.gr

Keywords: Agent-based systems, Intelligent Tutoring Systems, Ontologies, Personalized Learning, Distance Learning, Dynamic Courseware Generation.

Abstract: In this paper an approach for building an intelligent tutoring system is presented, based on a multi-agent architecture and combined with ontologies for knowledge representation. The system developed is focused on a bottom up, reactive generation of an active sequence of knowledge units regarding a set of adjustable, high level learning goals. The learning process begins with a set of simple learning goals that require a few learning objects and as the educational process proceeds, the student has to achieve higher learning outcomes that combine other low level outcomes which have been already achieved. The system is able to adapt to student's learning profile and progress by applying proper learning tactics to prioritize through a weight calculation scheme the sequence of the learning outcomes to achieve. The main components of the system consisting of ontological models of the learner and the subject under study, gateway agents and tutor agents with their core modules (learning space management and learning tactics control) are explained and a detailed description of their interaction is given in the context of an example application. Finally, the advantages of the proposed approach are laid out, especially in the setting of a distance learning education system.

1 INTRODUCTION

The term Intelligent Tutoring Systems (ITSs) refers to complex tutoring systems that can be adapted to the needs, characteristics and learning progress of the individual learner (Polson & Richardson, 1988). These systems exploit a large amount of educational knowledge and usually they employ pedagogical methodologies. Especially in the case of agent-based architectures, the interaction between the different components of the ITS is achieved through the communication of the intelligent agents assigned to each component. A typical architecture of an ITS consists of four models: (a) the domain model which contains all the knowledge and problem-solving strategies to be learned, (b) the student model which is an overlay of the domain model; it is the core component of an ITS and stores all the data about student's characteristics and progress, (c) the tutoring (or pedagogical) model which contains all the information about the various pedagogical decisions and methodologies and (d) the user interface (UI) which enables the communication between the user and the system (Nkambou et al., 2010).

In this paper we introduce a pilot educational system that enhances personalized learning of students in the context of selected courses. We propose an agent-based intelligent tutoring system, able to adapt to student's characteristics by employing learning tactics based on the student's learning profile and progress. More specifically, our proposed multi-agent system architecture employs a set of homogenous student-dedicated *tutor agents* for each course. Each agent builds an internal learning model based on the domain and available resource semantic representation while during the educational process the agent updates the model based either on the student's learning profile and interaction or by accessing the student's progress with respect to a given group. The tutoring system is not domain specific while the pedagogical module is versatile, allowing tutors to experiment on different learning tactics in order to engineer more domain-

specific or student profile-oriented agents. Finally, the system self organizes student groups based on overall group progress indicators and without any tutor interference. To the best of our knowledge, this is the first indirect approach towards self-organized learning.

In the context of our work we have chosen to model the main components of the proposed system through ontologies. Ontologies have been widely used especially in the field of education and specifically in tutoring systems for three main reasons: (i) to support the formal representation of abstract concepts and the relations between them in a reusable and extendable way, (ii) to allow the extraction of new knowledge by applying inference mechanisms and (iii) to provide rich semantics for humans to work with and the formalism for computers to perform mechanical processing. Furthermore, ontologies facilitate the reuse and the integration of services and thus e-learning systems are able to provide better applications (Peña & Sossa, 2010).

Agent technology is a well-accepted approach to address the challenges of technology enhanced learning. In our case, by using intelligent agents in a distance education system it is possible to obtain adaptivity to each individual student's learning capabilities, particularities and learning progress.

The proposed tutoring system follows a 3-tier architectural style. In the presentation tier users connect to the system through a web interface; the logic tier consists of a multi-agent system; agents connect with a semantic repository in order to access the domain related reusable learning objects and student profiles (data tier). The multi agent system is implemented using the Java Agent Development Framework (JADE), a middleware for the development and execution of peer to peer applications following the agent-based development paradigm (Bellifemine et al., 2003).

The system has been designed in the context of the Hellenic Open University (HOU). HOU has a mission to offer university level education using distance learning methodology and to develop the appropriate material and teaching methods. Currently, HOU offers 31 undergraduate and postgraduate Study Programs with a total of approximately 30,000 students, coached by 1,700 tutors in 1,550 groups (20 students per group on average). Students of the HOU usually live in disparate locations all over the country. Besides being students they usually have families and working obligations so they have pressing time constraints for studying. Given the special characteristics of an adult distance learning education system, the provision of tools, such as the one presented here, that can facilitate the learning process and enhance the learning experience are of great importance.

The rest of the paper is structured as follows. Section 2 provides related work on agent-based e-learning systems. In the following section we elaborate on the ontological models we have implemented in order to represent the learners and generally the knowledge of the domain to be taught. Section 4 presents our agent-based ITS architecture focusing on the tutor agent organization and logic. A detailed system usage example is provided in order to demonstrate system's functionality. The next section contains a discussion of the developed system and provides future directions of work for its improvement. Finally our conclusions are given.

2 RELATED WORK

Multi agent system (MAS) is a technology where its application came into existence during 1980's. A number of e-learning systems use the multi agent scheme to create sophisticated environments in order to achieve maximum effectiveness in learning by implementing different technologies and using different methodologies (Bokhari & Ahmad, 2014). An example in the domain of multi-agent e-learning systems is (Ali et al., 2010) where the authors present a multi agent approach for designing an e-learning system architecture. The proposed architecture consists of four tier layers, namely Interface layer, Middle layer, Database Controller layer and Database layer. The middle layer is based on MAS and supports any information communication, login, logout and new user sessions creation. Another example of a multi agent system that exploits ontologies for describing the educational material as well as the learners and their learning styles is presented in (Dung & Florea, 2011). The authors here present an architecture to support a multi-agent e-learning system, where intelligent agents are capable of providing personalized assistance according to learner's learning style and knowledge level. A study by (Hammami et al., 2009) describes an architecture composed of four multi-agent system levels interacting with each other using intelligent blackboard agents; blackboard agents facilitate the cooperation and coordination among interacting agents. Each level consists of different agents specialized on interfacing, authoring and learning aspects depending on the human user role. The system is connected to a number of databases modelling the student profile, the learning process, the learning domain, teaching material and practices.

The authors in (Acampora et al., 2010) apply a Memetic Computing methodology into a hierarchical multicore multi-agent system while formalizing memetic agents' exploration of taxonomic knowledge as an optimization problem in order to compute personalized learning experiences. Their approach includes building a set of knowledge highways whose paths connect information sources, earner's requirements and cross feasible learning contents. Memetic agents explore the available learning knowledge taking into account hardware details of the available computing resources. The domain model employs a semantic representation of the educational domain including a set of teaching preferences; the learning presentation generation algorithm uses a predefined learning path of concepts to be covered and generating the best sequence of learning activities to best satisfy the concept path. In (Yaghmaie & Bahreininejad, 2011) the authors suggest a framework for building an adaptive Learning Management System (LMS). The proposed architecture is based upon multi-agent systems and uses both Sharable Content Object Reference Model (SCORM) 2004 and Semantic Web ontology for learning content storage, sequencing and adaptation. Moreover, they provide a way to adapt course topics according to learners' experiences whose learning style is similar to the current learner.

3 ONTOLOGICAL MODELS

Ontologies are used for modelling the learners and the knowledge of the learning domain. In order to develop the ontologies we have followed a widely-adopted methodology, described in (Noy & McGuiness, 2001) and for their representation we adopted the Web Ontology Language (OWL) (McGuinness & van Harmelen, 2004). The implementation process was done by the Protégé tool which is the most widely used and offers a complete development environment. Below we give a more detailed description of the ontological models we have implemented.

3.1 Learner Model

In order to implement the learner model we were based, on the one hand, on student modelling standards (Smythe et al., 2001, LTSC Learner Model Working Group of the IEEE 2000, 2000) and, on the other, on empirical studies that were conducted by social scientists among students of HOU. The proposed learner model is thus a combination of stereotype and overlay techniques. A fully stereotype-based profile, as the information derived from student's descriptions or questionnaires is not accurate for every knowledge domain and the system would adapt to student's needs very slowly. Dynamic attributes related to the learning process are represented with an overlay model. From the empirical studies we extracted information about the dimensions/characteristics of the learner profile that could affect his/her academic performance. A few examples of these dimensions are: the learning style, previous experience, reasons for education, computer literacy, etc. The values for these attributes (i.e. stereotypes) are used for the initialization of the learner's profile and then, after the initialization phase the profile is dynamically modified as the overlay model is updated through the interaction of the user with the system.

The proposed student model is partially based on the standards we mentioned above, but they also have limitations as they reflect different perspectives on the attributes of a learner (e.g. classic CV notion based or student's performance as the most important information). On the other hand, as resulted from the study of other similar student models, there is no approach that satisfies all the attributes of an adult learner within a distance learning environment.

The learner's model: (1) is a dynamic model that can change over time as the system collects information about the user, (2) is a long-term model that keeps generalized information about the user and not only for the current interaction with the system and (3) combines "active" and "passive" user modelling techniques, i.e. in the beginning user provides direct information about him/her and then the system collects data indirectly. The proposed ontology defines the following four upper level classes: (a) *Student* which represents any student, (b) *StudentCourseInformation* which holds information about learner's academic performance during the entire educational process, (c) *StudentCurrentActivity* which captures learner's activity for the current academic year and (d) *StudentPersonalInformation* which is the most compact class of the proposed ontology, representing not only learner's static data, such as demographics, but also more complex characteristics that concern his/her interaction with the system. A detailed description of the ontology is provided in (Panagiotopoulos et al., 2013).

3.2 Learning Objects and Outcomes

In order to represent the domain knowledge we have used the notion of Learning Objects. A learning object (LO) is defined as "*a self-contained and independent unit of digital educational content,*

which is associated with one or more learning objectives and it has as primary aim the ability of reuse in different educational contexts" (Nikolopoulos et al., 2012b). The ontological representation and description of LOs has been based on the metadata schema proposed in (Nikolopoulos et al., 2012a).

The system also needs to keep a record of the learner's performance. To achieve that, we have used the notion of Learning Outcomes. According to the Bologna project (Bologna Working Group, 2005) a learning outcome is a statement of "*what a learner is expected to know, understand and be able to demonstrate after completion of a learning process (a lecture, a module or an entire program), which are defined in terms of knowledge, skills and competence*". For the classification of the learning outcomes in different level skills we have applied the Revised Bloom's taxonomy (Krathwohl & Anderson, 2001), as it is the most widely used. The detailed description of the ontological representation for the learning outcomes is given in (Kalou et al., 2012).

It is worth noting here that the process for the cognitive domain representation from which we construct the corresponding learning objects and also the definition of the learning outcomes for the different cognitive domains is realized within a well-defined and applied collaborative methodology between domain experts (tutors) and knowledge engineers, described in (Panagiotopoulos et al., 2012, Nikolopoulos et al., 2013).

4 AGENT-BASED PLATFORM

4.1 System Architecture Overview

In this section, we provide an overview of the ITS prototype, called APLe (Agents for Personalized Learning), whose system architecture is depicted in Figure 1. Students interact with the platform through a web based interface; a servlet keeps track of all available Gateway Agents (GA) and acts as dispatcher in order to route each student request/action to the proper GA, based on user and selected course information. GAs are special purpose agents that interface between agents of a remote agent environment and the servlet. GAs maintain and utilize student to Tutor Agent (TA) mappings in order to transfer request/action messages between students and corresponding agents (inside their particular agent environment). In Figure 1 two such environments are depicted representing two different courses (e.g. Structured Programming in C and Software Engineering). For different students attending this course the system will spawn different TAs; with red color we depict a student UI-TA mapping example.

In case a GA has no mapping for a particular user (e.g. on user login), it creates a TA. Then, requests/actions are transformed into specific data structures (we refer to these as Blackboard Beans or simply beans) which encapsulate request/action specific information such as session id, user id, course, action type and action data. GAs are triggered either by incoming beans or FIPA ACL (Foundation for Intelligent Physical Agents (FIPA), 2002) compliant messages. If the agent receives a bean, the agent translates the data into an ACL message which is transmitted to the corresponding agent. On the opposite, when the agent receives an ACL message from a TA then the agent a) finds the corresponding bean, b) attaches the agent action/response data to the bean and c) sends the bean back to the servlet. TAs and GAs are grouped inside a set of agent environments (JADE containers) which can be distributed in different physical places of the service provider infrastructure. Each agent environment contains at least one GA which is created on system startup. On the other hand, TAs are generated and terminated dynamically.

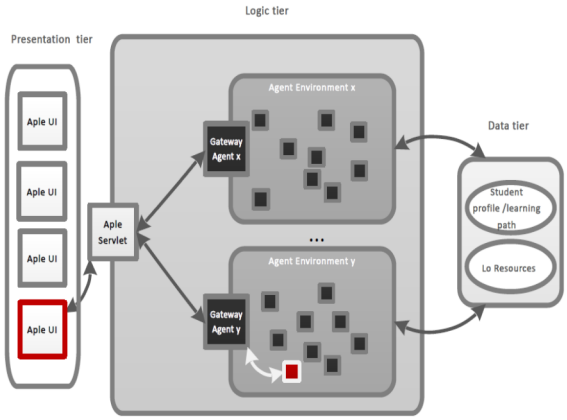

Figure 1: System Architecture.

Although this approach introduces some extra communication overhead (there exist user actions/requests that do not affect the learning process and thus could be handled in a central fashion), we argue on transferring the information load to a dedicated tutor agent for scalability, error tolerance and security reasons: No matter of system traffic, the servlet and GAs do not consume resources on data processing and communication with the repositories. If for some reason some TAs fail, these agents are terminated while the system continues to operate for other connected users.

Moreover, in order to deal better with any occurring bottleneck delay, multiple gateway agents may exist in each agent environment; finally, JADE agent mobility can be applied to allocate agents in more remote containers in respect of a particular grouping policy, e.g. container traffic. The current implementation employs a simple agent grouping policy: TA grouping/placement is applied in regard of the course that is currently attended while each group (environment) contains exactly one GA.

The data tier consists of a semantic repository and a content repository. The content repository is a data storage facility for the available educational content that is available for presentation. The semantic repository contains semantic representation and instances for the students (student profile and action log), Learning Objects, Learning Outcomes and finally the Domain concepts for each available course. As a semantic repository, we use OWLIM-Lite, a high-performance semantic repository implemented in Java and packaged as a Storage and Inference Layer (SAIL) for the Sesame openRDF framework (Bishop et al., 2011). Each TA is able to interact with the semantic repository through the respective OWLIM-Lite by using a set of predefined SPARQL query and update patterns. Each pattern is defined in respect of the structural and relational properties of the ontologies used for the semantic representation.

4.2 Tutor Agent

Each Tutor Agent is allocated to a student that attends a particular course at the current point of time. The Tutor Agent (TA) architecture consists of two modules, as depicted in Figure 2: Learning Space Management (LSM), reflecting the internal representation and Learning Tactic Control (LTC), reflecting the learning tactic decision. Sesame refers to persistence/metadata, whereas LP Updates refer to the user feedback.

LSM is triggered either when the agent receives a learning request/action message either on an LTC-generated internal event. LTC may be triggered in a three way fashion: periodically, on message arrival or on LSM-generated internal event.

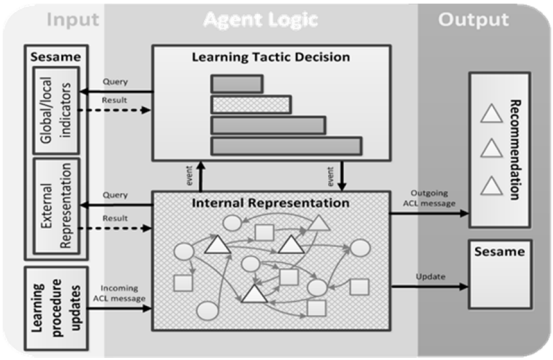

Figure 2: Tutor Agent architectural design

Taking under consideration the agent architecture, courseware generation is affected not only as a result to a student's learning events but also in accordance to the relative distance between the student's indicators and the group's indicators. According to this, the TA reacts to direct and indirect stimuli:

- *Direct stimuli* refer to student learning events (or student learning event chains) which affect directly the node state set of the learning space. The agent reacts to the state change by guiding the student towards a set of (most fitted) learning outcomes. The set is produced after applying the dominant learning tactic to the personal learning space.
- *Indirect stimuli* refer to indirect interaction between students through specific learning indicators which are estimated globally during the progress of the group of agents for a particular course. The agent reacts (through LTC) to indicator changes by switching the (most fitted) learning tactic.

Learning Space Management

This module creates and operates upon a complex graph structure which represents the personal learning map of a student. The learning space is modeled as a 3-color graph using a variety of links based on explicit/implicit properties which are extracted from the combined educational ontology. Each node is described by its type, current state and current value. The state of each node is determined according to the state of its connections. For that purpose, a set of well defined, non-recursive, non-overlapping transition rules are applied on initialization or after an educational update (incoming message).

The module is able to parse the graph and extract filtered information based on a particular learning tactic and the student profile. The outcome may be either high level (objective) recommendation (based

on the learning tactic) or low level (learning object) recommendation based on a student-selected objective and the student profile. Finally, the module updates the ontology repository (the student profile in particular) upon each update.

Learning Tactics Control

In educational context, a learning tactic is the way a student is attempting to learn something (Popham, 2011). We define an agent learning tactic as the way an agent selects the next Learning Outcome for a student to learn (to persist). More specifically, a learning tactic is a set of connection types and corresponding weights that apply to each node based on the status of its very local neighbourhood (directly connected nodes).

The Learning Tactic Control (LTC) is a reactive selection mechanism which uses global and local (internal) indicator updates in order to select one learning tactic to apply to LSM. Each time the LTC is triggered, a series of queries is sent to the ontology repository concerning some quantitative data about the class (or group of students). Next, using a formula that is solely based on indicator data, the available learning tactics are hierarchically checked to take the control of the LSM. The hierarchical winner-takes-all mechanism is based on Brook's subsumption architecture: when triggered, top level behaviours (in our case, learning tactics) suppress lower level behaviours from triggering. When the dominant learning tactic switches, LTC triggers an internal event which forces LSM to comply with the dominant learning tactic by resetting the connection weights of the learning graph according to the new learning tactic. The result of this action is a rebalanced learning graph.

4.3 System Usage Example

In this section we describe the execution phase of the tutoring system each time a student connects to the system, based on the following assumptions: a tutor of the "*Structured programming in C*" course, has created an initial learning plan of a single learning goal (objective): "*PA_PLI10_46_*". For that purpose, we employ the educational ontology discussed in Section 3, consisting of Learning Outcomes, Learning Objects, the Bloom taxonomy schema, an educational domain schema and finally a student profile schema. More particularly, the combined ontology contains 124 classes, 55 object property types, 45 data property types and 737 individuals, including 109 Learning Outcomes, 128 Learning Objects and 208 C programming specific Learning Concepts. A Learning Outcome has a natural language description, an assigned Bloom level, a number of connections with relative Concepts and a number of connections with relative Learning Objects. For example, "*PA_PLI10_46_*" refers to "*combining operators and operands in a program to form expressions*", it is related with C concepts like "*operator*", "*operand*" and "*expression*" and it is satisfied with Learning Object "*MA_PLI_25*". The latter is titled "*Common mistakes on using operators*" related with Concepts "*operator*" and "*operand*", it refers to a document-formatted example (resource type). The "*operator*" concept is connected with parent concepts like "*expression*" and a number of child concepts like "*Logical Operator*", "*Bitwise Operator*" and "*Numeric Operator*".

Also, we consider a set of two learning tactics: a) a *rapid-advance strategy* which focuses on selecting learning objectives towards higher goals as far as at least one sub-objective is fulfilled and b) a *greedy strategy* which focuses on achieving all sub-objectives before moving toward a higher goal. The first learning tactic is triggered by using two indicator sets: student versus mean class quantity of learning goals achieved multiplied by the mean class versus student self-evaluation score. The latter tactic is triggered by using a formula of two indicators: student quantity versus mean class quantity for the successful learning objectives. Also, the rapid advance learning tactic suppresses the greedy tactic.

When a student connects to the course, a Tutor Agent spawns inside the multi agent "*Structured Programming in C*" container; next, the agent initializes the learning space using the initial set of objectives according to the learning plan. Next, the graph is populated and connected recursively with learning objectives, objects and concepts according to a breadth first strategy using a defined set of connection types. Currently, the exploited connection types are five: "*satisfies*" between a Learning Object and an Objective; "*subject*" between a Learning Object/Objective and a Concept; "*hasBloomLevel*" for Learning Objectives and the Bloom level; finally "*hasParent*" / "*hasChild*" between Concepts. The learning space generation algorithm is set to expand uniformly all possible connection chains with maximum Concept distance 2 from each initial learning objective *(#46* in our case). Using the initial set of the learning plan, the generated learning space graph involves 46 Learning Outcome, 52 Learning Object and 93 C programming Concept nodes. Next, the learning space synchronizes according to the student relevant data from the student log. In our scenario, the course has just started so there is no relevant data in the student log. At this point, the student is able to use the recommender. A graphical representation of the

learning space is depicted in Figure 3. This graph represents a fraction of the learning map that is built to support the goal of learning the semantics of C operators. Learning Objects are not shown for clarification reasons.

When the student selects the recommendation button, the event is passed to the tutor agent who calculates and returns back a list of the most valued objectives of the learning space, according to the dominant learning tactic. Each learning tactic applies to each Learning Objective (node) of the learning space as follows:

- rapid-advance: each node *x* estimates its score based on the formula: $f(x) = 1/(1 + dist) * \#nodes_{in,finished}/\#nodes_{in} * (\#nodes_{out} - \#nodes_{out,finished})/\#nodes_{out}$ where $dist$ is the distance of a node from the closest learning goal, $\#nodes_{in/out}$ is the number of connected incoming/outgoing nodes and $\#nodes_{in/out,finished}$ is the number of finished incoming/outgoing nodes. If there are no incoming nodes, $\#nodes_{in,finished} = nodes_{in} = 1$. If there are no outgoing nodes, $(\#nodes_{out} - \#nodes_{out,finished})/\#nodes_{out} = 1$
- greedy: each node *x* estimates its score based on the formula: $g(x) = \#nodes_{in,finished} - \#nodes_{in,pending}/\#nodes_{in}$, where $\#nodes_{in,pending}$ is the number of (incoming) nodes that are not finished. If there are no incoming nodes, $g(x) = 1$.

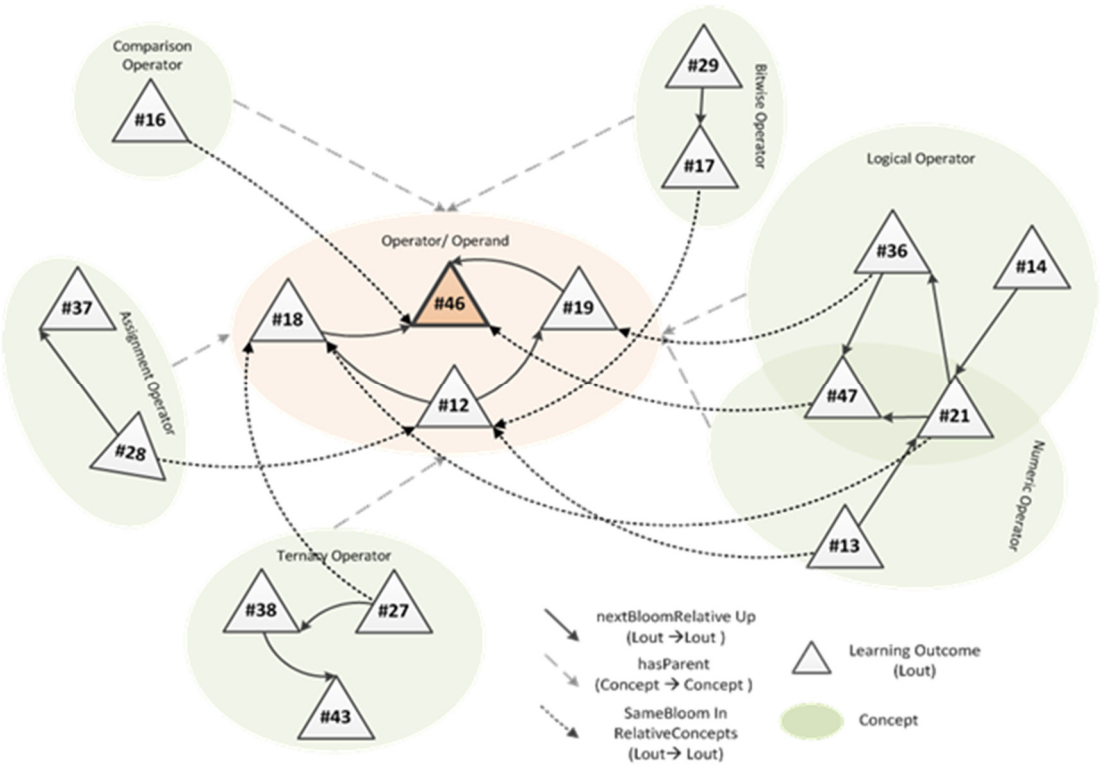

Figure 3: Simplified representation of the learning space in the context of the example course

To better understand how a learning tactic affects the recommendation, consider Learning Objective nodes 16, 19, 36, 27, 13 and 47: assuming there are no (visited/finished) nodes, the former learning tactic will estimate values ½, 0, 0, ⅓, ⅓ and 0 respectively. According to this, the rapid-advance tactic will recommend the sequence 16, 27 and 13. The latter tactic will estimate 1, -1, -1, 1, 1 and -1 respectively, leading to a random recommendation sequence for 16, 27 and 13, since all nodes have the same weight. If we assume nodes 16 and 21 as visited, the values for 19 and 27 are not affected (neighbours are unchanged). Nodes 36, 13 and 47 are affected, giving estimations ½, ⅙, ½ for the

former and 1, 0, 1 for the latter learning tactic. According to this, the rapid-advance tactic will recommend the sequence 36, 27 and 13, whereas the greedy tactic a random sequence between 36, 27 and 47.

When the student selects an objective to attain, the selection is passed to the agent who calculates and returns back a list of the existing learning objects with respect of the selected objective and the student preferences, located in the student profile (Figure 4) (the language used in the user interface is currently greek). For example, if the student prefers visual content and the objective concerns the topic "*recursive functions*", a video learning object will be selected, if available explaining this topic. It is noted that the agent sorts instead of excluding learning format/types. Thus, the student is able to select a learning object of his/her choice.

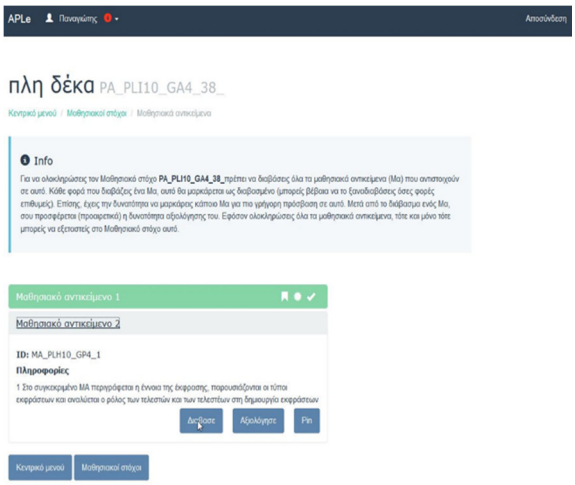

Figure 4: Screenshot of the APLe System: Choosing learning objects for a specific learning outcome

Finally, when the student finishes the study of a particular learning object, the student has to self-evaluate his/her understanding on the learning object. This action triggers the agent to update the learning space and the student data repository. Also, the agent updates its indicator data for the class.

5 DISCUSSION

The advantage of the agent-based platform derives from the fact that the tutor agents can provide recommendation on a sequence of learning outcomes that most fit the student profile, according to the properties of the learning objects. On the other hand, the use of ontological models for representing and storing the information regarding the learner and the learning material enhances reusability of this information and promotes interoperability with third-party systems.

The research work described in this paper acts as a proof of concept and there are still many challenges that are related with the particular approach for engineering an ITS. Further study and experiments will follow in order to verify well fitted agent configurations (learning tactics and LTC) under different domains. A research direction is to identify the (bipartite) application of student stereotyping into dynamic reconfigurations and vice versa.

According to these, the next step is to evaluate the proposed system with real data from the students of the Hellenic Open University (HOU). In order to evaluate the proposed agent-based tutoring system, our approach involves evaluating the system through user's experiences to find out the usability and impact of the ITS, finding learning rates and achievements level.

We already have prepared a number of learning objects with different metadata and for various knowledge domains. These objects have different file formats (video, document, presentation, etc.) and different resource type (activity, exercise, self-assessment, etc.). The criteria for the evaluation of the APLe, have emerged from the study of different system evaluation methodologies (such as TAM2) and are represented through scored questionnaires that will be given to the students of the HOU. For example, there are questions about the usability of the system, the interface and knowledge acquisition.

In HOU, we are developing (in a collaborative effort among ontology experts and course tutors) educational ontologies for the majority of the 600 courses we offer. Our aim is to gradually introduce these ontologies to the platform and deploy the respective agents for each course. Currently, about 40 courses are in the "pipe-line". The platform will eventually become a component of the HOU educational portal, which will offer a personalized learning environment to our students. In its first deployment, the course ontologies will be independent, thus a different instantiation of the platform per course is planned. This approach will also help us sidestep scalability issues and allow us measure system's performance, so as to plan the next deployment phase.

6 CONCLUSIONS

This paper presented an integrated intelligent tutoring system in order to support distance learning, especially for adult learners. The proposed architecture is based on a multi-agent system which

facilitates the communication between the different components of the ITS and provides personalized learning to the individual students. The operational procedure of the multi-agent system has been described and the overall functions of its fundamental components have been illustrated. The prototype provides dynamic curriculum sequencing in a bottom up fashion using direct information about the student preferences or learning styles and relative information about the student learning process as part of a group.

ACKNOWLEDGMENTS

This research has been co-financed by the European Union (European Social Fund – ESF) and Greek national funds through the Operational Program "Education and Lifelong Learning" of the National Strategic Reference Framework (NSRF) (Funding Program: "HOU").

REFERENCES

Acampora, G., Loia, V., Gaeta, M., 2010. Exploring e-Learning Knowledge Through Ontological Memetic Agents. *IEEE Computational Intelligence Magazine*, *vol.5, no.2*, *pp. 66-77*.

Ali, A, P., Dehghan, H., Gholampour, J., 2010. An Agent Based Multilayered Architecture for E-learning system. *In Proceeding of E-Learning and E-Teaching (ICELET). Second International Conference on IEEE, pp.22-26*

Bellifemine, F., Caire, G., Pogg, A., Rimassa, G., 2003. Jade-A White Paper. *In EXP in search of innovation, vol. 3, no. 3*, 2003, *pp. 6-19*.

Bokhari, M., Ahmad, S., 2014. Multi-agent Based E-Learning Systems: A Comparative Study. *In Proceedings of the 2014 International Conference on Information and Communication Technology for Competitive Strategies*

Bologna Working Group, 2005 *A Framework for Qualifications of the European Higher Education Area*.

Bishop B., Kiryakov, A., Ognyanoff, D., Peikov, I., Tashev, Z., Velkov, R., 2011. OWLIM: A family of scalable semantic repositories. *In Semant. web*, *vol. 2, no. 1*, *pp. 33-42*.

Dung, Q, P., Florea, M, A., 2011. An Architecture and a Domain Ontology for Personalized Multi-Agent e-Learning Systems. *In Proceedings of the 2011 Third International Conference on Knowledge and Systems Engineering, pp. 181-185*

Foundation for Intelligent Physical Agents (FIPA). *FIPA ACL Message Structure Specification* (2002), viewed 27 May 2014, http://www.fipa.org/specs/fipa00061/.

Hammami, S., Mathkour, H., Al-Mosallam, E.A., 2009. A multi-agent architecture for adaptive E-learning systems using a blackboard agent. *2nd IEEE International Conference on Computer Science and Information Technology*, *pp. 184-188.*

Kalou, A., Solomou, G., Pierrakeas, C., Kameas, A., 2012. An Ontology Model for Building, Classifying and Using Learning Outcomes. *12th IEEE International Conference on Advanced Learning Technologies, pp. 61-65.*

Krathwohl, D., Anderson, L., (eds), 2001. *A Taxonomy for Learning, Teaching, and Assessing: A Revision of Bloom's Taxonomy of Educational Objectives*, Abridged Edition.

LTSC Learner Model Working Group of the IEEE 2000. *Draft Standard for Learning Technology - Public and Private Information (PAPI) for Learners (PAPI Learner)*, IEEE p1484.2/d7, 2000-11-28.

McGuinness, D.L, van Harmelen, F., 2004. OWL Web Ontology Language Overview, W3C Recommendation, http://www.w3.org/TR/owl-features/

Nikolopoulos, G., Solomou, G., Pierrakeas, C., Kameas, A., 2013. An Instructional Design Methodology for Building Distance Learning Courses. *7th International Conference in Open and Distance*.

Nikolopoulos, G., Kalou, A., Pierrakeas, C., Kameas, A., 2012a. Creating a LO Metadata Profile for Distance Learning: An Ontological Approach. *6th Research Conference, pp. 37-48*.

Nikolopoulos, G., Solomou, G., Pierrakeas, C., Kameas, A., 2012b. Modeling the Characteristics of a Learning Object for Use within e-Learning Applications. *Proceedings of the Fifth Balkan Conference in Informatics, pp. 112-117*.

Nkambou, R., Mizoguchi, R., Bourdeau, J., (eds), 2010. *Advances in intelligent tutoring systems*. Heidelberg: Springer, 2010.

Noy N., McGuiness D., 2001. Ontology Development 101: A Guide to Creating Your First Ontology. *Stanford Knowledge Systems Laboratory Technical Report KSL-01-05 and Stanford Medical Informatics Technical Report SMi-2001-0880.*

Panagiotopoulos I., Kalou, A., Pierrakeas, C., Kameas, A., 2012. Adult student modeling for intelligent distance learning systems. *In Special Issue on AIAI 2012 of the International Journal of Engineering Intelligent Systems for Electrical Engineering and Communications, vol. 21, nos. 2/3*.

Panagiotopoulos, I., Kalou, A., Pierrakeas, C., Kameas, A., 2012. An Ontological Approach for Domain Knowledge Modeling and Management in E-Learning Systems. *In 1st AI in Education Workshop: Innovations and Applications, pp. 95-104*.

Peña, A., Sossa, H., 2010. Semantic representation and management of student models: An approach to adapt lecture sequencing to enhance learning. *In Proceedings of the 9th Mexican international conference on Advances in artificial intelligence: Part I, pp. 175-186*.

Polson, M., Richardson, J., 1988. *Foundations of Intelligent Tutoring Systems*. Psychology Press.

Popham, W.J., 2011. *Transformative Assessment in Action: An Inside Look at Applying the Process*. ASCD, USA.

Smythe, C., Tansey, F., Robson, R., 2001. IMS Learner Information Package Information Model Specification, http://www.imsglobal.org/profiles/lipinfo01.html

Yaghmaie, M., Bahreininejad, A., 2011. A context-aware adaptive learning system using agents. *In Expert Systems with Applications*, *Vol. 38, Issue 4, pp. 3280-3286.*