APLe: Agents for Personalized Learning in Distance Learning

Stamatis Panagiotis[1], Panagiotopoulos Ioannis[1], Goumopoulos Christos[1,2(✉)], and Kameas Achilles[1]

[1] Hellenic Open University, Patras, Greece
panagiotis.stamatis@gmail.com, gpanagiotopoulos@ecomet.eap.gr, goumop@tutors.eap.gr, kameas@eap.gr
[2] Information and Communication Systems Engineering Department, Aegean University, Lesbos, Greece
goumop@aegean.gr

Abstract. This work presents an intelligent tutoring system that supports personalized learning especially in a distance learning setting. The proposed architecture is based on multi-agents which facilitate the communication between the different components and on ontologies that provide a sound and complete representation of the knowledge domain. The operational procedure of the multi-agent system is described and the overall functions of its fundamental components are illustrated. The prototype, called APLe, provides dynamic learning path 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. The preliminary evaluation of our system indicated positive feedback by the users in terms of the usability of the system, completeness of the educational content and ability of the system to adapt to the individual's educational needs and preferences and informed for further developments required.

Keywords: Agent-based systems · Intelligent tutoring systems · Ontologies · Personalized learning · Distance learning · Dynamic courseware generation

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 [1]. 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

S. Zvacek et al. (Eds.): CSEDU 2015, CCIS 583, pp. 37–56, 2016.
DOI: 10.1007/978-3-319-29585-5_3

about the various pedagogical decisions and methodologies and (d) the user interface (UI) which enables the communication between the user and the system [2].

In this work we introduce a pilot educational system, called APLe, which 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 homogeneous 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 [3].

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 [4].

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 text is structured as follows. In Sect. 2 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 3 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 provides a preliminary evaluation of our current accomplishments. The advantages of the proposed approach are laid out, especially in the setting of a distance learning education system and future directions of work for its improvement are discussed. Section 5 provides related work on agent-based e-learning systems. Finally our conclusions are given.

2 Ontologies

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 [5] and for their representation we adopted the Web Ontology Language (OWL) [6]. 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.

2.1 Learner Model

In order to implement the learner model we were based, on the one hand, on student modelling standards [7, 8] 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. Table 1 shows the stereotypic profiles we have used and their corresponding values.

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.

Table 1. Stereotypic profiles and their corresponding values.

Dimensions	Values
Learning style	visual – auditory – read/write – kinesthetic
Use of technology	adaptable – adaptive
Computer literacy	novice – beginner – advanced
Previous experience	novice – beginner – advanced
Time for study	no time – little – much
Reasons for education	career development –career change – general knowledge
Academic literacy	poor – good – excellent
Socialization style	social – solitary

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. Table 2 lists the subclasses that exist under the upper level class `StudentPersonalInformation`. The table gives also a brief description of the entities that are represented by these classes. Figure 1 depicts the student model ontology, as displayed in Protégé.

Learning Style. The VARK model [9] argues that for practical reasons, the processing of information is done through the senses. The model describes how human senses are being used for knowledge and skills acquisition. Defines the following four categories of learners: (a) Visual, (b) Auditory, (c) Read/Write and (d) Kinesthetic.

We have chosen to use the VARK model in order to categorize learners by their learning style for two main reasons: (1) there is already in the official website of VARK a scored questionnaire that can help individuals to identify their learning style and (2) the different learning styles can be directly linked with the metadata of the learning objects as we have defined them (e.g. the file format of the object – video, document, audio, etc.).

In order to keep a track of student's interaction with our ITS (learning outcomes achieved, learning objects viewed, evaluation of the learning objects) we have

Table 2. Description of the `StudentPersonalInformation` class.

Class name	Class description
`Accessibility`	The overall set of features that characterizes the student's behavior during his interaction with the e-learning system.
`Disabilities`	The set of student's disabilities that could affects the educational process.
`DemographicData`	Student's static demographic information
`InteractionPreferences`	Student's preferences regarding interaction with the e-learning system
`MediaPresentation`	Student's preferences regarding the presentation of learning objects
`Language`	Student's preferences regarding the language of the learning objects
`LanguageSpoken`	Student's native languages
`LanguagePreferred`	Language that the student prefers for the presentation of learning objects
`Aesthetics`	Aesthetic factors such as the use of highly interactive sensory and visual communication
`Color`	Student's preferences regarding the coloring scheme of learning system's environment
`Fonts`	Student's preferences regarding the fonts used by the learning system's environment
`LearningStyle`	Student's learning style - This class is further divided to the sub classes according to the VARK model
`MotivationState`	Student's motivation during the educational process
`LearningGoals`	Overall goals set by the student
`ReasonsForEducation`	The reasons why the student desires to engage in the educational process
`AcademicLiteracy`	Student's previous formal educational experiences
`Interests`	Student's interests
`TimeStudy`	The average time per day that the student can use for studying

introduced the following classes: (a) `_Course` which is the class where are included the courses attended by the student, (b) `_LORecord`, which keeps all the records for the relations between a user and a particular learning outcome, (c) `_LearningTrack` which represents a low level tracking during the learning procedure. It is connected to a course, a student and a `_LORecord`, with a student, (d) `_StudentLMap` which keeps all the learning paths of the users.

Furthermore, datatype properties, that link individuals to data values, have been set in order to define more effectively the classes. In the proposed model the stereotypic profiles (Table 1) have been expressed as datatype properties.

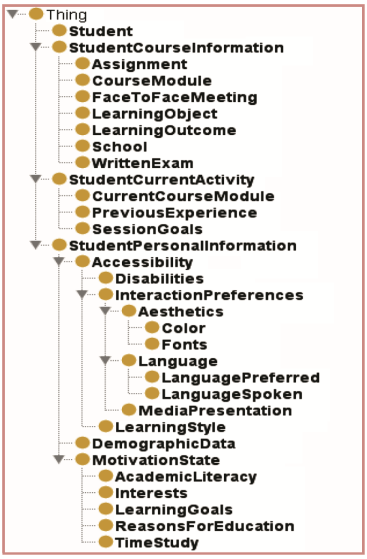

Fig. 1. The student model ontology as displayed in Protégé.

2.2 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" [10]. The ontological representation and description of LOs has been based on the metadata schema proposed in [11].

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 [12] 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 [13], as it is the most widely used. The detailed description of the ontological representation for the learning outcomes is given in [14].

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 [15, 16].

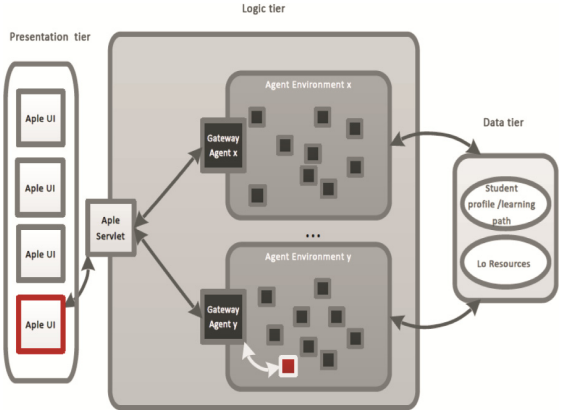

Fig. 2. System architecture.

3 Agent-Based Platform

3.1 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 Fig. 2. 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 customized JADE gateway wrapper 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 Fig. 2 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 asynchronous incoming beans or FIPA ACL [17] compliant messages via a cyclic ACL message tracking behavior. 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. In some edge cases such as the termination of a TA or some user actions irrelevant to the TA's tutoring process, the GA responds directly by populating the bean with the corresponding information. The GA processing algorithm is depicted in Fig. 3.

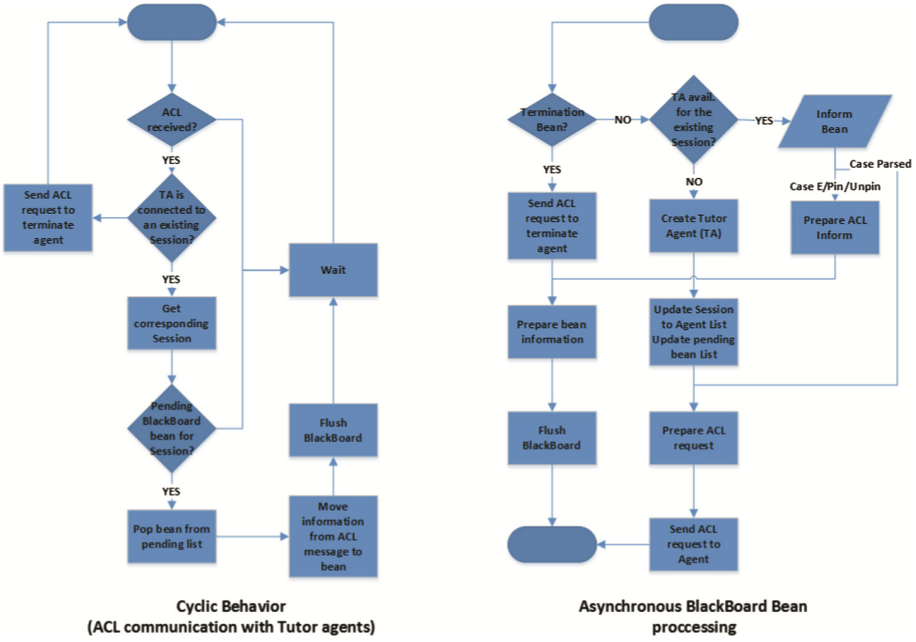

Fig. 3. Gateway Agent processing for incoming ACL messages and Blackboard Beans.

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.

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, robustness and security reasons. No matter of system traffic, the servlet and GAs do not consume resources on data processing while the communication with the repositories is minimal and exposed through a lightweight API which is limited on credential, logging, student and course general information. If for some reason some TAs fail, these agents are terminated while the system continues to operate for other connected users. If for some reason the session with the student terminates, the corresponding Tutor Agent is terminated as well. 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 [18]. 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.

3.2 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 Fig. 4: Learning Space Management (LSM), reflecting the internal representation and Learning Tactic Control (LTC), reflecting the learning tactic decision. Sesame refers to persistence/metadata, whereas Learning Procedure Updates refer to the user feedback.

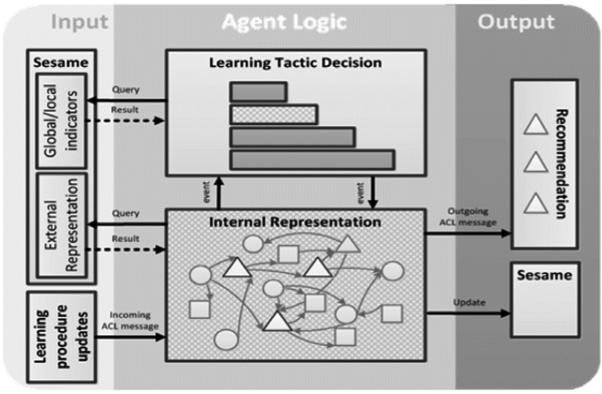

Fig. 4. Tutor Agent architectural design.

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.

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. The agent is able to generate the learning space using a (Learning Outcome – oriented) algorithm. According to this, an initial (disconnected) graph of learning outcomes is obtained from the educational plan. Next, a recursive series of queries is applied towards the combined ontology according to a breadth first strategy, populating and connecting the initial graph with links. Each link type represents a relation which is inferred from the original ontology using a set of predefined queries. The link set is created in regard to the learning tactic set described in the next section. The advantages on using the particular learning space construction algorithm are: (a) flexibility to create a vast variety of different complexes, based on the original set of goals, the set of well-defined links between learning goals and finally the depth of each link; (b) the algorithm can apply for any connection link between learning outcomes, as far as the link is feasible, based on the available query set. On the other hand, the authors recognize two issues that must be dealt with: a) due to the heavy time complexity, the algorithm may be unpractical if applied by each (computationally limited) agent. Instead, it is recommended for each agent to get an offline version instead of creating it; (b) Agent memory capacity bounds the actual size of the learning space; (c) Finally, the algorithm is not computationally optimal due to its generic purpose. If computational resources are limited and the relation set is predefined, it is then highly recommended to optimize the algorithm by taking advantage of particular structural and relational properties of the ontologies.

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). There are two types of rules: *reactive* and *chain transition* rules. A reactive rule describes a state transition based on a particular learning event related with a particular node (e.g. the user confirmed to have read a learning object thus meaning a state transition of the learning object node from *unknown* to *studied*). A chain transition rule is applied in the direct vicinity of a node which has an updated state. A chain transition rule describes a particular node state transition (current state to a new state) based on quantitative criteria that apply on the neighbor node states (e.g. the former node state change of the learning object node may trigger a state change to all neighbor learning outcome nodes). The current version of APLe uses a simple transition model, described in Fig. 5. Learning Objects have two states (*unknown*, *studied*), Concepts have two states (*unvisited*, *visited*) and Learning Outcomes have four states (*unvisited*, *parsed*, *covered* and *finished*). Only a single reactive transition rule exists to update the Learning Object state while every other transition is triggered in chain reaction if the quantitative criteria are satisfied. As an example,

when a new learning object is studied, then (a) all connected Concepts with status *unvisited* will become *visited* recursively; (b) all connected Learning Outcomes with status *unvisited* will become *parsed* recursively; (c) for the set of all the Learning outcomes updated or Learning Outcomes connected to the updated Concepts, the transition rule will evaluate (and trigger) recursively; (d) for each node on the set of Learning outcomes with state *covered* connected to the initial Learning Object or having an updated state *covered* the transition rules will evaluate (and trigger if satisfied).

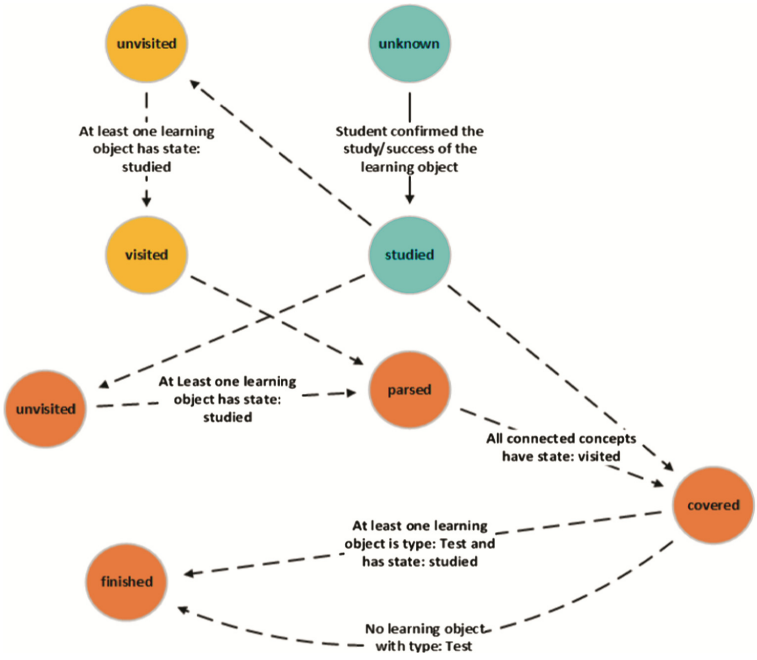

Fig. 5. Transition model supporting chain transition rules in APLe.

The LSM 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 [19]. 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 neighborhood (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 [20] which specifies that 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.

3.3 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 Sect. 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 Fig. 6. 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.

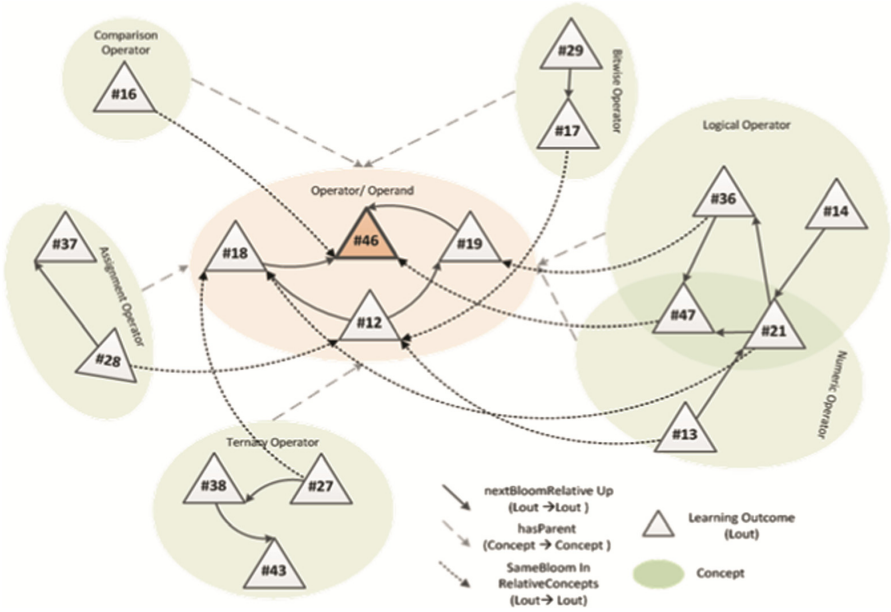

Fig. 6. Simplified representation of the learning space in the context of the example course.

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: $f(x) = 1/(1 + dist) * \#nodes_{in,finished} / \#nodes_{in} * \left(\#nodes_{out} - \#nodes_{out,finished}\right) /$

$\#nodes_{out}$ 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, $g(x) = 1$

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, 0,1/3, ¼ 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 1/3, 1/8, ¼ for the former and 1, 0, 1 for the latter learning tactic. Thus, 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 (Fig. 7) (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.

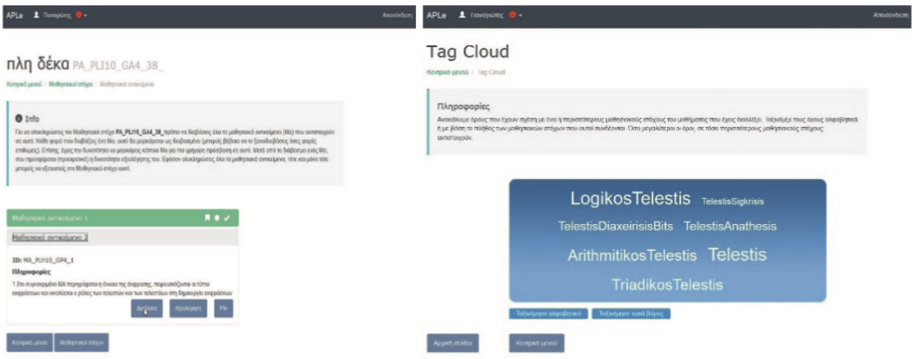

Fig. 7. Screenshots of APLe system user interface: choosing learning objects for a specific learning outcome (on the left); tag cloud service (on the right).

In addition, to the main screen, the student can see a tag cloud on the concepts which appear in the current educational path, as given in Fig. 7. Furthermore, the student can classify concepts according to their weight or alphabetically. Selecting a specific tag concept, the student is driven to all learning objects associated with the specific concept. It should be noted that in this way the user overrides the tutor agent; the later will be able, however, to adapt to the new context (learning object selected by the student).

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.

4 Evaluation

For a preliminary evaluation of the proposed system we conducted a small-scale evaluation, where the participants were members of the Educational Content Methodology & Technology Laboratory at HOU. Fifteen volunteer members of the Lab participated in the evaluation (4 female, 11 female, average age 23.7). All the participants had a computer engineering background and were familiar with the educational content and methodologies of the HOU.

For the evaluation of the proposed agent-based system, our approach involved evaluating the system through user's experiences to find out the usability and impact of the ITS, finding learning rates and achievements level. The first step for the evaluation process was to create a number of new user accounts in the repository with respective credentials (username/password) which were provided to each participant. Along with the credentials an instruction manual of the system was sent to all the participants.

The next step of the preparation of the system in order to be used, was to import an adequate number of learning objects with different file formats and resource types, appropriately created by Hellenic Open University's tutors, corresponding to the learning outcomes of the course "Structured programming in C", as described in Sect. 3.3). By using the property "resource type" of the learning objects, we categorized also a learning object according to its content (e.g. theory, example, exercise, activity etc.). The learning outcomes were updated weekly, according to the curriculum of the course. The evaluation process lasted one month, during which each participant had to use the system and finally complete a questionnaire (which we will describe below).

4.1 Questionnaire

For the completion of the evaluation process, a questionnaire was provided on-line to the participants. The criteria for the evaluation of the APLe, have emerged from the study of different system evaluation methodologies (such as Technology Acceptance Model 2, TAM2) and are represented through scored questions. More specifically we have used the likert scale (5-point scale). There is also a number of open questions about general comments and remarks.

The questions were divided in three categories: (1) Usability, (2) Educational content and (3) Adaptivity. Briefly, in the first category there are questions about the easiness of using the system, the interface, navigation features, functions and menus. The second category includes questions about the relevance of the learning objects with the learning goals, the validity of the educational content, how the knowledge acquisition through APLe is facilitated and what are the main flaws regarding the content and types of the learning objects. Finally, questions in the last category concern the completeness of the forms for the creation of the student profile, if there are any other parameters besides those included in the profile, that affect the adaptivity of the system that we should take into account and finally the creation of the educational paths.

4.2 Results

In this section, we briefly present and discuss the results of the evaluation process as they were captured through the questionnaire that the participants had to complete after the use of the APLe system at the end of the evaluation period of one month.

In the question whether the APLe system is easy to learn, 100 % said that they agreed or strongly agreed. When the participants asked whether the navigation functions are easy to understand, 80 % answered positively, while 20 % disagreed or remained neutral. In the question if the overall user interface is friendly, more than 87 % agreed and strongly agreed and only 12.5 % disagreed. We also received some remarks about some of the functionalities which are mainly front-end issues like the use of natural language in the description of the learning objects that could be easily solved.

Regarding the educational content, in the question whether the content is relevant to the learning goals, 90 % agreed while 10 % remained neutral. In the question whether the educational content is valid, 89 % agreed and 11 % remained neutral. In the question whether knowledge acquisition in facilitated through the use of the APLe system, 72.5 % agreed while 27.5 % remained neutral. Some of the major shortcomings related to the educational content integrated in the APLe system, as remarked by the participants, was the lack of more rich educational content types such as self-assessment activities, self-evaluation tests, etc.

Finally, regarding the adaptivity of the system, in the question whether the forms that the APLe uses for the creation of the learning profile of the student are complete and easy to understand, 87.5 % agreed and strongly agreed, while 12.5 % remained neutral. In the question whether the APLe system is capable to support the student's learning profile (by adding and using more and different learning formats of learning objects), 87.5 % agreed and strongly agreed, while 12.5 % remained neutral. Regarding the creation of the learning paths and if they meet the user's learning preferences, while there was a positive feedback (80 % agreed and strongly agreed) highlighting as a positive remark the capability offered by the system to override the system suggestions through the tag cloud service, there was also a critique that it was not clear to the user how the learning paths are created (e.g. a diagram) and more feedback on that would be helpful.

From the above brief presentation of the results, it becomes clear that, regarding the usability of the system there was a positive assessment for both navigation and learning

of the system. In relation to the educational content the majority of the participants were positive towards the content (learning objects), and were encouraging on whether the APLe system is able to enhance the traditional way of learning with the remark that more types of educational content should be added. Regarding the adaptivity of the system, almost all participants found complete and fully understandable the forms for the creation of the student profile, they consider that APLe is adapted to the student learning profile and preferences, while on the creation of the learning paths there were requests to provide more feedback on their creation.

In the next section we present some of the next steps and suggestions in order to improve the functionality and reliability of the system.

4.3 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.

According to the results of the small-scale evaluation we present in the previous section, the next step is to integrate to the APLe system 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.). We also plan to prepare some tests which will form the basis for the fulfillment of a learning goal.

Regarding the creation of the learning paths, we plan to integrate a module to the system which will provide to the student an overall picture of his/her learning progress and the next steps he/she has to follow. This would be in a form of a diagram, for example, which will be accessible at any time by the student.

Finally, and after we have made the necessary updates we plan to evaluate the proposed system during the time-period of a semester with students of HOU. Our final goal is to provide to the community of the HOU the APLe system as an integrated system that will assist students during the educational process.

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.

5 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 [21]. An example in the domain of multi-agent e-learning systems is [22] 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 [23]. 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. In [24] a study 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 [25] 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 [26] 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.

6 Conclusions

In this work an approach for building an intelligent tutoring system was 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) were explained and a detailed description of their interaction was given in the context of an example application. Finally, the advantages of the proposed approach were laid out, especially in the setting of a distance learning education system.

Acknowledgements. 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

1. Polson, M., Richardson, J.: Foundations of Intelligent Tutoring Systems. Lawrence Erlbaum Associates, Mahwah, New Jersey (1988)
2. Nkambou, R., Mizoguchi, R., Bourdeau, J.: Advances in Intelligent Tutoring Systems. Springer, Heidelberg (2010)
3. Sossa, H., Peña Ayala, A.: Semantic representation and management of student models: an approach to adapt lecture sequencing to enhance learning. In: Sidorov, G., Hernández Aguirre, A., Reyes García, C.A. (eds.) MICAI 2010, Part I. LNCS, vol. 6437, pp. 175–186. Springer, Heidelberg (2010)
4. Bellifemine, F., Caire, G., Pogg, A., Rimassa, G.: Jade a white paper. Telecom Italia EXP Mag. **3**(3), 6–19 (2003)
5. Noy, N., McGuiness, D.: 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 (2001)
6. McGuinness, D.L, van Harmelen, F.: OWL web ontology language overview W3C recommendation (2004). http://www.w3.org/TR/owl-features/
7. Smythe, C., Tansey, F., Robson, R.: IMS Learner Information Package Information Model Specification, IMS Global Learning Consortium. http://www.imsglobal.org/profiles/lipinfo01.html
8. LTSC Learner Model Working Group of the IEEE 2000. Draft Standard for Learning Technology - Public and Private Information (PAPI) for Learners (PAPI Learner), p. 1484.2/d7. IEEE, 28-11-2000
9. Fleming, N.D., Mills, C.: Helping students understand how they learn. The Teaching Professor, vol. 7(4), Magma Publications, Madison, Wisconsin, USA (1992)
10. Nikolopoulos, G., Solomou, G., Pierrakeas, C., Kameas, A.: Modeling the characteristics of a learning object for use within e-Learning applications. In: 5th Balkan Conference in Informatics, pp. 112–117. ACM, New York (2012)

11. Kameas, A., Pierrakeas, C., Kalou, A., Nikolopoulos, G.: Creating a LO metadata profile for distance learning: an ontological approach. In: Dodero, J.M., Palomo-Duarte, M., Karampiperis, P. (eds.) MTSR 2012. CCIS, vol. 343, pp. 37–48. Springer, Heidelberg (2012)
12. Bologna Working Group, A Framework for Qualifications of the European Higher Education Area. http://ecahe.eu/w/images/7/76/A_Framework_for_Qualifications_for_the_European_Higher_Education_Area.pdf
13. Anderson, L.W., Krathwohl, D.R., Bloom, B.S.: A Taxonomy for Learning, Teaching, and Assessing: A Revision of Bloom's Taxonomy of Educational Objectives. Longman, London (2001)
14. Kalou, A., Solomou, G., Pierrakeas, C., Kameas, A.: An ontology model for building, classifying and using learning outcomes. In: 12th IEEE International Conference on Advanced Learning Technologies, pp. 61–65. IEEE, New York (2012)
15. Panagiotopoulos, I., Pierrakeas, C., Kameas, A., Kalou, A.: An ontological approach for domain knowledge modeling and management in e-learning Systems. In: Iliadis, L., Maglogiannis, I., Papadopoulos, H., Karatzas, K., Sioutas, S. (eds.) Artificial Intelligence Applications and Innovations, Part II. IFIP AICT, vol. 382, pp. 95–104. Springer, Heidelberg (2012)
16. Nikolopoulos, G., Solomou, G., Pierrakeas, C., Kameas, A.: An instructional design methodology for building distance learning courses. In: 7th International Conference in Open and Distance (2013)
17. Foundation for Intelligent Physical Agents (FIPA). FIPA ACL Message Structure Specification (2002). http://www.fipa.org/specs/fipa00061/
18. Bishop, B., Kiryakov, A., Ognyanoff, D., Peikov, I., Tashev, Z., Velkov, R.: OWLIM: a family of scalable semantic repositories. Semant. Web **2**(1), 33–42 (2011)
19. Popham, W.J.: Transformative assessment in action: an inside look at applying the process. ASCD, Alexandria, USA (2011)
20. Brooks, R.A.: Intelligence without representation. Artif. Intell. **47**, 139–159 (1991)
21. Bokhari, M., Ahmad, S.: Multi-agent based e-learning systems: a comparative study. In: 2014 International Conference on Information and Communication Technology for Competitive Strategies. ACM, New York (2014)
22. Ali, A.P., Dehghan, H., Gholampour, J.: An agent based multilayered architecture for E-learning system. In: 2nd International Conference on E-Learning and E-Teaching (ICELET), pp. 22–26. IEEE, New York (2010)
23. Dung, Q.P., Florea, M.A.: An Architecture and a domain ontology for personalized multi-agent e-Learning systems. In: 3rd International Conference on Knowledge and Systems Engineering, pp. 181–185. IEEE, New York (2011)
24. Hammami, S., Mathkour, H., Al-Mosallam, E.A.: A multi-agent architecture for adaptive E-learning systems using a blackboard agent. In: 2nd IEEE International Conference on Computer Science and Information Technology, pp. 184–188. IEEE, New York (2009)
25. Acampora, G., Loia, V., Gaeta, M.: Exploring e-Learning knowledge through ontological memetic agents. IEEE Comput. Intell. Mag. **5**(2), 66–77 (2010)
26. Yaghmaie, M., Bahreininejad, A.: A context-aware adaptive learning system using agents. Expert Syst. Appl. **38**(4), 3280–3286 (2011)