Dynamic e-Learning Content Selection with BDI Agents

João de Amorim Jr.(✉), Thiago Ângelo Gelaim, and Ricardo Azambuja Silveira

PPGCC – UFSC, Florianópolis, Brazil
{joao.amorim, thiago}@iate.ufsc.br,
ricardo.silveira@ufsc.br

Abstract. This paper presents an e-learning content selection model, based on multi-agent paradigm, aiming to facilitate the learning material reuse and adaptability on Learning Management Systems. The proposed model was developed according to a BDI multi-agent architecture, as an improvement of the Intelligent Learning Objects approach, allowing the dynamic selection of Learning Objects. A prototype was implemented to validate the proposed model, using the JADEX BDI V3 platform, and allowing to build improved learning experiences.

Keywords: Dynamic learning experience · Intelligent Learning Objects · Intelligent learning environments

1 Introduction

The distance learning plays an important role on the educational process worldwide. Several learning institutions have adopted e-learning as one of their acting strategies, and new ways of online learning are becoming more and more common [1].

Teaching aid systems must be geared to enhance the educational experience. Two aspects contribute to it: adaptability and reuse. The former is related to different students' needs and styles. An adaptable system increases the student understanding, in a personalized way [2–4]. The latter avoids the need of developing a new resource if exists another one with the same learning intention [3, 5].

Intelligent Tutoring Systems (ITS), Learning Management Systems (LMS) and Learning Objects (LO) are some computational tools that enrich the learning process. ITS are applications created for a specific educational domain, usually with some few adaptability and interoperability resources [6]. Thus, if a change in the learning domain is needed, the ITS must be reconfigured. Following, LMS are learning environments used to build online courses (or publishing material), to monitor the student progress and to manage educational data [1, 7, 8]. Finally, LO are digital artifacts for supporting the teaching-learning process, promoting reuse and adaptability [9].

Although LO and LMS allow reusability, they have a limited level of adaptability. The kind of adaptation they offer needs to be pre-programmed by the instructional designer or teacher, that is, this systems usually are no dynamic adaptable [8, 9].

J. Bajo et al. (Eds.): PAAMS 2015 Workshops, CCIS 524, pp. 299–308, 2015.
DOI: 10.1007/978-3-319-19033-4_25

This article presents the first results of our research that seeks the convergence of these three different paradigms for the development of intelligent learning environments. The first step, presented in this paper, was to develop a search engine, delivery and presentation systems to get learning objects obtained from repositories.

1.1 Related Work

There are several similar studies developed to provide adaptability to educational systems. Some of them are built as an extension of a LMS using conditional jumps [8], Bayesian networks [4] or data mining technics [7] as their adaptive strategy. Other researches are not integrated with a LMS, and use distinct ways to adapt the learning to the students' style, such as ITS [6], Recommender System [2], etc. [10, 11].

In addition, there are also some works based on the Multi-Agent System (MAS) approach to produce smarter applications. MAS are composed by autonomous entities (agents) which percept the environment they are situated in and act on it to achieve their objectives. The communication and cooperation of individual agents make possible to solve complex problems, which they cannot solve individually [12, 13].

An important MAS architecture for intelligent agents is the BDI model. BDI agents are composed of mental attitudes (Belief, Desire, Intention), and act following the practical reasoning process (goal deliberation and means-end reasoning) [12, 14].

Some analyzed works combine LMS and MAS to make the former more adaptive. An example is the use of data obtained from a MOODLE [15] forum to show resources and activities to the students [16]. There is also an intelligent, dynamic and adaptive environment, based on agents that are able to identify the student cognitive profile [17].

However, all this related studies have two features that can be improved. The former is the way as the student's learning style is determined. The analyzed works use questionnaires in the beginning of the course to do it. This extra step can be considered intrusive and distracting [2]. The exception is the work that clusters students in profiles based on their assessments performance (grades) [17]. The latter is the possibility of coupling new learning resources (LO) dynamically to the environment. The teacher (or instructional designer) need to configure previously all the possible course paths for each student style, what could be hard and take so much time [4, 18]. Further, the attaching of a new LO to the course involves modifying the course structure.

To produce more intelligent LO, previous researches proposed the convergence between the LO and MAS technologies, called Intelligent Learning Objects (ILO) [19, 20]. This approach makes possible to offer more adaptive, reusable and complete learning experiences. An ILO is an agent capable to play the role of a LO, which can acquire new knowledge by the interaction with students and other ILO (agents information exchange), raising the potential of student's understanding.

The LO metadata permits the identification of what educational topic is related to the LO [9]. Hence, the ILO (agents) are able to find out what is the subject associated with the learning experience shown to the student, and then to show complementary information (another ILO) to solve the student's lack of knowledge in that subject.

The next section presents a different model to handle the two issues pointed out in the analyzed works, and to improve the ILO model as well.

2 ILOMAS

The new proposed model is based on MAS approach integrated to a LMS, resulting on the improvement of the analyzed related works, allowing that LO can be included to the learning experience dynamically, adding intelligent behavior to the system. The solution's adaptability is based on the possibility of new LO be attached to the LMS without previous course configuration, as soon as the system finds out that the student needs to reinforce its understanding on a specific concept. This is automatically identified through the verification of the student assessment performance (grade), on each instructional unit, or by student choice, when interacting with the LO. Moreover, the course structure becomes more flexible, since it is unnecessary to configure all the possible sets of learning paths for each student profile.

The proposed model achieves reuse by the combination of pre-existed and validated LO whose concept (subject) is the same of that the student needs to learn more about, avoiding that teachers or instructional designers need to build new materials.

The objective of the new model called Intelligent Learning Object Multi-Agent System (ILOMAS) is to enhance the framework developed to create ILO based on MAS with BDI architecture [21], extending this model to allow the production of adaptive and reusable learning experiences. The idea is to select dynamically ILO in the LMS according to the student performance, without previous specific configuration on the course structure. The ILOMAS is composed by agents with specific goals, and capable of communicating and offering learning experiences to students in a LMS course, according to the interaction with these students.

3 Analysis and Design

The modeling of the ILOMAS framework uses an Agent-Oriented Software Engineering methodology. The Prometheus methodology defines a detailed process to specify, design and implement intelligent agent systems based on goals, plans and beliefs (BDI) [22]. It is composed of several activities and phases to generate specification and design documents. Another reason to choose this methodology is the existence of Prometheus Design Tool (PDT), an Eclipse IDE plug-in that supplies an iterative development of the Prometheus diagrams [23].

Following the Prometheus methodology, on the system specification phase, it was identified the goals, roles, agents and their respective interfaces with the environment (perceptions, actions and external data). It was detected two kinds of agents (Fig. 1):

- LMSAgent – Represents the LMS. It is responsible for: (1) receiving the student learning experience request (*Learning Experience Request* perception); (2) finding out the subject that the student must learn about (*Learning Experience Subject Identifier* role), according to the information provided from the LMS data base (external data); and (3) for passing the control of the interaction with the student to a new agent of the kind ILOAgent (*Inform Learning Object Subject* message).

Its beliefs are information provided from the LMS database, such as the students, the courses and the respective subject related to the current learning experience (that is, the topic that the student must learn about), for each student enrolled on the LMS's courses. The communication between the agent and the LMS can be made through the calling to data base access API functions (such as Java JDBC, PHP PDO, etc.) or Web Services. The LMSAgent goals are *GetStudentCourseGoal* and *GetStudentSubjectGoal* (sub-goals of *IdentifyLearningExperienceSubject* goal), which allow the agent to identify the information of the current learning experience associated with a specific student.

- ILOAgent – Agent responsible for showing the learning experience to the student. This is made through the presentation of a LO related to the subject that the student needs to learn about, as determined by the LMSAgent. The function of an ILOAgent agent is to search a correspondent LO on the LO repository (*Learning Object Searcher* role) and show it to the student (*Learning Experience Show* action), furthermore, this kind of agent must keep itself aware to the interaction between the student and the system (*Learning Object Player* role).

  The beliefs of this agent are the information about the current learning experience (obtained from LMSAgent), the chosen LO, and the status of the interaction with the student. The goals are *GetLORepositoryGoal*, *GetRelatedLOGoal*, and *SelectLOGoal* (sub-goals of *SearchLearningObject* goal), which enable the agent to search for a LO (related to the specific subject) in the repository.

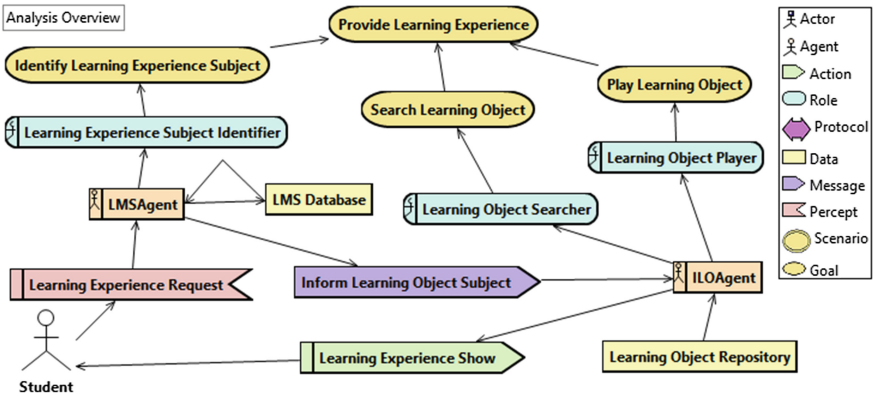

Fig. 1. Analysis overview diagram

The Agent Overview diagram (Fig. 2) presents the agent's plans (the concrete way to achieve the agent's goals [12]), perceptions, actions, internal messages, and capabilities. The idea of using capabilities is to module the agent features, allowing the reuse of commons characteristics (such as plans, goals, etc.) among distinct agents [22].

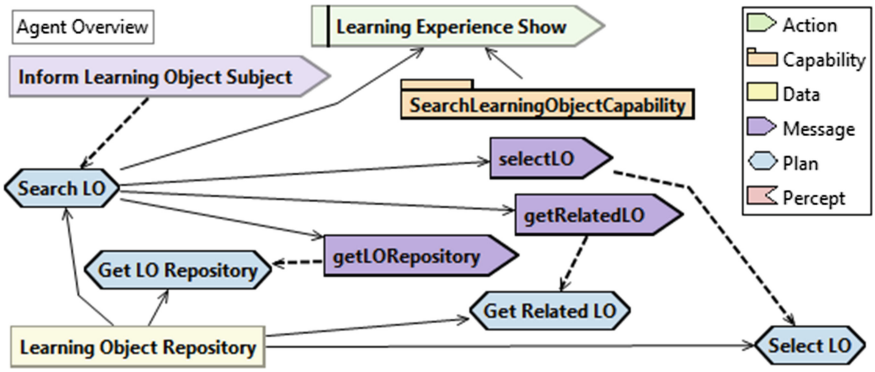

Fig. 2. Agent overview diagram (i.e.: ILOAgent)

4 Implementation

After the modeling phase, the ILOMAS framework and a prototype instance were implemented to validate the proposed model. The JADEX framework [24, 25] – an extension to JADE platform (FIPA compliant) with support to develop intelligent agent systems – was chosen to implement the agents based on the BDI architecture [12, 14].

The JADEX platform permits the creation of active components, an approach that gets benefits from the association of two distinct technologies: Agents and SCA. The SCA model was proposed by IT companies (i.e.: IBM, ORACLE) with the intention of promoting the interoperability among distributed applications, according to concepts of components and service oriented architecture (SOA) [24, 25].

The newest JADEX platform version is JADEX BDI V3 (V3). Before V3, the agents were built in ADF files (XML tags for beliefs, desires, etc.) and Java classes (plans) [26]. However, on V3 the agents are developed with pure Java classes (without XML files) and the annotations mechanism (beliefs, plans, etc.) [14, 25]. Further, on V3 the recommended way of agent communication is through service invocation. The BDI agents can be service providers (declaring specific annotations and implementing specific interfaces) and can request services of other agents, components, etc. [24, 25].

It is necessary to point out the implementation limitations. Instead of putting emphasis on visualization issues (such as formats, rich graphical user interfaces, etc.), it was emphasized the MAS development (with the agents and their interactions).

The interaction interface between the student and the agents' environment was implemented based on the Java Servlets and JSP technologies, getting benefits of the JADEX BDI V3 services communication structure. The Servlets technology allows the execution of services and Java classes at the server side from Web requests.

On the prototype, the servlet layer delegates the handle of the student's browser request to a class (non-agent) based on the Facade design pattern. This pattern provides a unified and simplified interface to a sub-system, promoting low coupling [27]. The ILOMASFacade class offers to the servlet classes the access to agents' services (the agents' capabilities, plans, etc.) keeping the separation between the MAS layer and the external items (front-end and servlets), avoiding unnecessary coupling (Fig. 3).

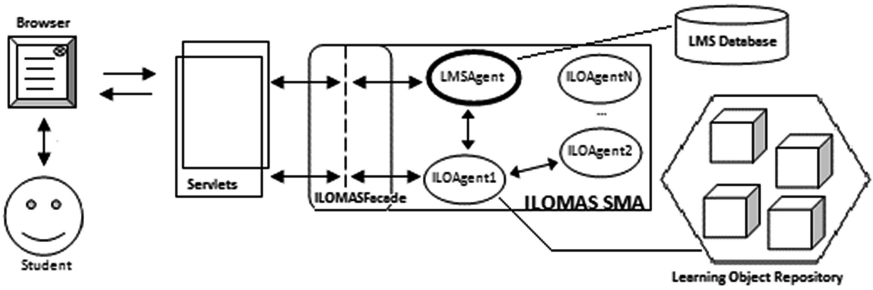

Fig. 3. ILOMAS Web prototype architecture

4.1 Developed Classes

Following JADEX BDI V3 framework structure, the developed agents were declared with the @Agent annotation and with the BDI suffix in their class name. Besides, it was required to specify a BDIAgent type attribute (annotated with @Agent). This attribute type provides the necessary methods to execute the reasoning and behaviors on the BDI model, such as dispatch an agent goal (leading to plan execution).

Other annotations can be used in agent classes to declare BDI attributes (@Belief, @Capability, etc.) and methods. The method annotations define agent behavior to specific states of the agent's life cycle, such as @AgentCreated (after agent start up), @AgentKilled (before agent die), and @AgentBody (agent run) [14, 25, 28], i.e.: when the ILOAgentBDI starts running, its goal of search what LO will be shown to the student is dispatched, according to the subject identified by the LMSAgentBDI. Additionally, the agents were specified as service providers, using the annotations @Service and @ProvidedService, and implementing the methods of the Java Interface corresponding to each respective service, whose return is asynchronous [25, 28].

The BDI elements (beliefs, goals and plans) related to the functionality of determining the subject associated with the learning experience were grouped on the IdentifyLearningExperienceCapability class. This capability class has beliefs that are instances of another class, which keeps information obtained from the LMS database (annotated with @Belief).

The IdentifyLearningExperienceCapability has also the goal *IdentifyLearningExperienceGoal* (and its sub-goals *GetStudentCourseGoal* and *GetStudentSubjectGoal*), declared with the @Goal annotation. The execution of a goal is made through the triggering of a specific plan represented by a method, using the @Plan and @Trigger annotations, (i.e.: *identifyLearningExperienceSubject* method).

Another developed capability was SearchLearningObjectCapability, whose main goal *SearchLOGoal* is composed of sub-goals: (1) find out the repository, (2) get the list of LO associated with the subject within the repository, and (3) choose one LO. Some metadata elements declared in IEEE-LOM [9] are used in this process. The ILOAgentBDI has the SearchLearningObjectCapability, which uses the element "keywords" of the LOM metadata to define the list of related LO.

The SearchLearningObjectCapability class declares one belief which references the LO repository found. The dispatching of the main goal *SearchLOGoal* results in the execution of the *searchLearningObject* plan method (annotated with @Plan and @Trigger).

4.2 ILOMAS Validation

The prototype was deployed to an Apache Tomcat Server (7.0.57) to test the proposed model. On the test beginning, a student accessed the system and the LMSAgent has identified that the student needed to learn about photosynthesis (Biology course). After, the student asked for the learning experience, leading to a new servlet requesting. This servlet has forwarded it to ILOMASFacade, which has waited for the end of ILOAgentBDI's deliberation. Then, this agent has discovered a LO related to the subject within the repository. Finally, this LO was shown to the student successfully.

Besides, it was simulated a student's request for a complementary learning experience (by pressing the corresponding button). As result, a new ILOAgentBDI was created on the system, which searched for and found a different LO related to the same subject (photosynthesis) within the repository. It was not explicitly defined in the database that the student should have watched this new LO (only the subject was required, no specific LO), so the MAS obtained the related LO dynamically (Fig. 4).

Fig. 4. ILOMAS Web prototype

5 Conclusions and Future Work

This paper presented a model to build more adaptive and reusable educational experiences, compared to related works. The ILOMAS framework was designed to allow the dynamic LO selection on LMS courses, as an improvement of ILO's previous approach. The agents are modeled based on the practical reasoning paradigm (towards goal achievement). The MAS was developed following the JADEX BDI V3 framework, which permits that the agents' functionalities can be accessed as services. The use of Servlet technology provides the integration of front-end and intelligent layer.

A prototype was implemented to verify the proposed model, and some evaluation tests were executed. As result, the ILOMAS has received the learning experience requested by the student, and has identified dynamically a LO associated with the subject that the student must have learned about (according to the LMS database).

As future work, the ILOMAS framework will be extended to supply the integration with SCORM [29] in order to raise reuse, dynamic sequencing, and interoperability. This improvement on the system will make possible to identify accurately the need of reinforcement by the student, according to the information about the interaction between the student and the object, provided by the messages received from and sent to the SCORM API (the data model elements, such as success, fail, latency, weighting, objectives, etc.) [30].

Another enhancement on the framework would be the use of some recommendation mechanism, made by specialized agents, to better choosing a LO among the list of objects associated with the specific subject within the repository. Currently, when there are more than one LO related to the desired subject within the repository, a random choice is made to select the next LO to be presented to the student, considering only the objects not shown. The overall process can be smarter by using a multiagent based recommender system for indexing and retrieving LO [31, 32]. The ILOMAS framework can consult this system before accessing the repository.

Finally, future works involve also the testing of the model with different learning situations and real students.

References

1. Allison, C., Miller, A., Oliver, I., Michaelson, R., Tiropanis, T.: The Web in education. Comput. Netw. **56**, 3811–3824 (2012). Elsevier
2. Vesin, B., Klasnja-Milicevic, A., Ivanovic, M., Budimac, Z.: Applying recommender systems and adaptive hypermedia for e-learning personalization. Comput. Inform. **32**, 629–659 (2013). Institute of Informatics
3. Mahkameh, Y., Bahreininejad, A.: A context-aware adaptive learning system using agents. Expert Syst. Appl. **38**, 3280–3286 (2011). Elsevier
4. Bachari, E., Abelwahed, E., Adnani, M.: E-learning personalization based on dynamic learners' preference. Int. J. Comput. Sci. Inf. Technol. (IJCSIT) **3**(3), 200–216 (2011)
5. Caeiro, M., Llamas, M., Anido, L.: PoEML: modeling learning units through perspectives. Comput. Standards Interfaces **36**(2), 380–396 (2014). Elsevier

6. Santos, G., Jorge, J.: Interoperable intelligent tutoring systems as open educational resources. IEEE Trans. Learn. Technol. **6**(3), 271–282 (2013). IEEE CS & ES
7. Despotovic-Zrakic, M., Markovic, A., Bogdanovic, Z., Barac, D., Krco, S.: Providing adaptivity in moodle LMS courses. Int. Forum Educ. Tech. Soc. **15**(1), 326–338 (2012). International Forum of Educational Technology & Society
8. Komlenov, Z., Budimac, Z., Ivanovic, M.: Introducing adaptivity features to a regular learning management system to support creation of advanced eLessons. Inform. Educ. **9**(1), 63–80 (2010). Institute of Mathematics and Informatics
9. Barak, M., Ziv, S.: Wandering: A web-based platform for the creation of location-based interactive learning objects. Comput. Educ. **62**, 159–170 (2013). Elsevier
10. Chen, C.: Intelligent web-based learning system with personalized learning path guidance. Comput. Educ. **51**, 787–814 (2008). Elsevier
11. Kurilovas, E., Zilinskiene, I., Dagiene, V.: Recommending suitable scenarios according to learners' preferences: An improved swarm based approach. Comput. Hum. Behav. **30**, 550–557 (2014). Elsevier
12. Wooldridge, M.: An Introduction to MultiAgent Systems, 2nd edn. Wiley, New York (2009)
13. Weiss, G. (ed.): Multiagent Systems: A Modern Approach to Distributed Artificial Intelligence. The MIT Press, Cambridge (1999)
14. Pokahr, A., Braubach, L., Haubeck, C., Ladiges, J.: Programming BDI agents with pure Java. In: Müller, J.P., Weyrich, M., Bazzan, A.L. (eds.) MATES 2014. LNCS, vol. 8732, pp. 216–233. Springer, Heidelberg (2014)
15. MOODLE – Modular Oriented-Object Dynamic Learning Environment. http://moodle.org
16. Alencar, M., Netto, J.: Improving cooperation in virtual learning environments using multi-agent systems and AIML. In: 41st ASEE/IEEE Frontiers in Education Conference, Session F4C, pp. 1–6. IEEE (2011)
17. Giuffra, P., Silveira, R.: A multi-agent system model to integrate virtual learning environments and intelligent tutoring systems. International Journal of Interactive Multimedia and Artificial Intelligence (IJIMAI) **2**(1), 51–58 (2013)
18. Brown, E., Cristea, A., Stewart, C., Brailsford, T.: Patterns in authoring of adaptive educational hypermedia: a taxonomy of learning styles. Edu. Technol. Soc. **8**(3), 77–90 (2005). International Forum of Educational Technology & Society
19. Silveira, R., Gomes, E., Vicari, R.: Intelligent learning objects: an agent-based approach of learning objects. In: van Weert, T., Tatnall, A. (eds.) Information and Communication Technologies and Real-Life Learning. IFIP, vol. 182, pp. 103–110. Springer, Boston (2006)
20. Silva, J., Silveira, R.: The development of intelligent learning objects with an ontology based on SCORM standard. In: Seventh International Conference on Intelligent Systems Design and Applications, pp. 211–216. IEEE (2007)
21. Bavaresco, N., Silveira, R.: Proposal of an architecture to build intelligent learning objects based on BDI agents. In: XX Informatics in Education Brazilian Symposium (2009)
22. Padgham, L., Winikoff, M.: Developing Intelligent Agent Systems – A practical guide. Wiley, New York (2004)
23. Prometheus Design Tool (Eclipse Plug-in). https://code.google.com/p/pdt-plugin/
24. Pokahr, A., Braubach, L., Jander, K.: The jadex project: programming model. In: Distributed Systems and Information Systems, Chap. 1, pp. 1–34. University of Hamburg (2012)
25. JADEX Active Components. http://www.activecomponents.org/
26. Braubach, L., Pokahr, A.: Jadex Active Components Framework – BDI Agents for Disaster Rescue Coordination. University of Hamburg (2011)
27. Gamma, E., Helm, R., Johnson, R., Vlissides, J.: Design Patterns: Elements of Reusable Object-Oriented Software. Addison-Wesley, New York (1995)

28. Braubach, L., Pokahr, A.: Developing Distributed Systems with Active Components and JADEX. University of Hamburg (2012)
29. SCORM 2004. Advanced Distributed Learning. http://www.adlnet.org/scorm
30. SCORM Run-Time Reference. Rustici Software. http://scorm.com/scorm-explained/technical-scorm/run-time/run-time-reference/
31. Vian, J., Campos, R., Palomino, C., Silveira, R.: A multiagent model for searching learning objects in heterogeneous set of repositories. In: 2011 11th IEEE International Conference on Advanced Learning Technologies (ICALT), pp. 48–52. IEEE (2011)
32. Campos, R.L.R., Comarella, R.L., Silveira, R.A.: Multiagent based recommendation system model for indexing and retrieving learning objects. In: Corchado, J.M., Bajo, J., Kozlak, J., Pawlewski, P., Molina, J.M., Julian, V., Silveira, R.A., Unland, R., Giroux, S. (eds.) PAAMS 2013. CCIS, vol. 365, pp. 328–339. Springer, Heidelberg (2013)