An Animated Pedagogical Agent playing the role of a Learning Management System

Beatriz Wilges¹, Ricardo Silveira¹
Universidade Federal de Santa Catarina, Centro Tecnológico, Departamento de Informática e Estatística, Campus Universitário, Florianópolis – SC, Brasil

{beaw, silveira}@inf.ufsc.br,

Abstract
This work presents a learning environment model based on Animated Pedagogical Agent (APA) manipulating Intelligent Learning Objects (ILO). For this purpose, a Learning Management System (LMS) was implemented in the form of an animated character with agent characteristics capable of interacting with determined ILOs, also implemented in this work, and constructed according to the proposed architecture. The results demonstrate the possibility of constructing more flexible and adaptable learning environments according to the ILO approach which proposes the convergence of the Learning Objects technology with the multi-agent systems paradigm.

KEYWORDS: learning environment, Intelligente Learning Objects, models.

1 Introduction
This work was motivated by the proposal of Intelligent Learning Objects approach presented by Silveira et al. (2004). Our approach claims that pedagogical agents can be used more efficiently, taking into account their interoperability within educational environments, aggregating the idea of implementation Animated Pedagogical Agent playing the role of a Learning Management System (LMS).

Johnson et al. (2000) and Jaques et al. (2003) state that animated pedagogical agents are a new paradigm in teaching environments because they can demonstrate complex tasks, employ locomotion and gestures to focus the student’s attention on the most important aspects of the learning activities. In addition of expressing emotional responses, they have the role of assisting the students in their interaction with the educational environment, being able to guide them in their actions and rendering the teaching – learning process more efficient, more pleasant and motivating.

According to Bradshaw (1997), an agent is a software entity which works continuously and autonomously in an educational environment generally inhabited by other agents. An agent is able to interfere in this environment in a flexible and intelligent way without requiring human intervention. Ideally, an agent should have the capacity to learn through past experiences and, if he/she inhabits an environment with other agents, he/she should be able to communicate and cooperate with them.

The main motivation for use of a Multi-agent System in the construction of this learning environment is the fact that one single agent requires vast knowledge to solve complex problems and, in some cases, as in the case of modeling of the learning process in educational environments, the problem is so complex that a single agent is unable to solve it (Brenner et al., 1998).

According to Ferber (1999), a Multi-agent System is a distributed computation application composed of a group of autonomous, heterogeneous, asynchronous and intelligent processes, which are called ‘agents’. These are able to cooperate with each other to solve complex problems, which would be impossible to solve individually.

Multi-agent Systems are most common employed to solve the kind of problems with distributed characteristics requiring independent entities which work toghether in order to guarantee the that the problem can be solved. That is a system composed of more than one agent working cooperatively in order to reach a common result.

According to Russell and Norvig (2004), an agent is an entity capable of perceiving its environment through sensors and by acting upon this environment through actuators. A Multi-agent System, however, is a system
constituted by a group of agents working to achieve a common objective.

In order to establish standards of communication and to make the agents compatible amongst themselves, the standards of the Foundation for Intelligent Physical Agents – FIPA (2006) were used in this work. This organization defines several characteristics that a Multi-Agent System should have in order to be considered interoperable. In this work, the implementation of this standard was obtained through the use of the FIPA-OS framework.

The FIPA-OS provides a set of components used to build agents compatible with the FIPA standards. According to Laukkanen (2000), the main objective of the FIPA-OS is to reduce the barriers in the adoption of FIPA technology through the supply of documents and technical specifications with source codes.

The architecture of FIPA-OS has three kinds of components: obligatory, optional and permutable. The obligatory components are necessary for the agents’ execution. The optional components, however, can either be used or not. The permutable components possess more than one implementation, which allows the selection of the most adequate implementation for the developer’s needs.

Some important considerations regarding the FIPA-OS are in relation to its capacity to divide an agent into Tasks, which are managed by the Task Manager. Execution of the tasks is based on events, which allow more than one to be executed at the same time. This is because they execute independent threads and can send and receive messages.

Within this context, it is believed that providing a Learning Object with typical agent characteristics, such as autonomy, knowledge about oneself and objectives, and with the possibility of interaction between the agent and the LMS through its capacity for communication, can give value to the objects and the educational environments in which they are inserted, making these environments more autonomous, dynamic and adaptable (Gomes et al., 2004).

In this work, the Animated Pedagogical Agent is responsible for the administrative and pedagogic tasks involved in a learning environment. In other words, the agent represents the Learning Management System (LMS), proposed in (LTSC, 2006). In a general way, the Animated Pedagogical Agent represents an interface for the management of learning objects through its metadata and interaction of these objects with the student.

2 Intelligent Learning Objects

For the purpose of this work, two Intelligent Learning Objects were developed: a simple arithmetic calculator and a set of learning content showed to the student in a typical blackboard. Both learning objects were developed according to the ILO approach to achieve interaction with the LMS animated agent.

The motivating idea behind the so-called Learning Objects is the assumption that development of new instructional material is not necessary for every new course created, as long as the already prepared material can be shared. The main objective of the Learning Objects is the reduction in time and money spent on the development of educational materials. A Learning Object is a piece of reusable educational material, i.e. Part or educational content which can be reused on several courses, several times.

In order to a Learning Object be reused, it must be compatible with different educational environments. This characteristic is obtained through the employment of well-defined standards. It must also be sufficiently modular to fit in to different contexts. Furthermore the Learning Objects must be stored in a safe place where they can be accessed and discovered. These places are named Learning Objects Repositories. The educational environments in which the Learning Objects are delivered to the students are called Learning Management Systems – LMS (LTSC, 2006).

The ability for the Learning Objects to be discovered is related to the formal description obtained through the metadata. The metadata are used to describe and categorize the learning objects, enabling a Learning Object to be discovered. In this work, the Learning Object Metadata standard (LOM) was selected to be used in this work. This standard was developed by the LTSC of the IEEE (LTSC, 2006) and it defines a hierarchy of data elements for the exposition of Learning Objects metadata.

A set of Learning Objects which constitute a learning environment is usually controlled by a Learning Management System – LMS. According to the LTSC IEEE (LTSC, 2006), a Learning Management System is a computational system, which can include the capacity to register students, control and guide the learning process, analyze and report on the student’s progress, plan the presentation of learning resources and track students.

Some LMS need to exchange information with their learning resources. The LTSC of the IEEE developed a model of data whose objective is to standardize the information, which is passed between the LMS and the content objects.

The standard used for this is the Data Model for Content Object Communication – DMOCOC, managed by workgroup number 11 of the LTSC IEEE. It describes a data model for the exchange of information between a content object and an LMS. Its structure is designed to support client-server environments in which an educational system using a LMS delivers digital content, a content object to the students.

In this work, we assume that an Intelligent Learning Object (ILO) is an agent capable of fulfilling the role of a
Learning Object (Gomes et al., 2004). In the same way, an ILO can be considered as a Learning Object that possesses an agent as a base.

Operationally, an ILO is an agent, which can generate learning experiences in the same way the learning objects do, that is, with reusability as the objective.

The use of Learning Objects concepts to create pedagogical agents is justified by several possibilities they can offer and methods of communication stand out as an example of this. An agent is capable of communicating through the exchange of messages using a high level of communication language named Agent Communication Language (ACL). In the more complete model of learning objects, SCORM (ADL, 2006), communication is made by passing through parameters and method calls, in the spirit of object orientation. This approach results in a very static communication where all possibilities should be predicted during the design stage. The use of an ACL extrapolates this limitation and provides greater dynamism to the process. This is due to the fact that the ACL are based on theories capable of providing more semantics to the communication. In addition, the content of the messages can be represented through a Content Language (CL), which is heavily based on logical formalisms. The result of the communication through the union between the CL and the ACL is potentially better than communication through the object orientation approach, as is the case with the current learning model objects.

A learning object designed as an agent is equipped with the ability to acquire new knowledge and behaviors throughout their existence through interaction with other students and even with other learning objects. With this idea, the learning object may evolve as it is no longer as static as in the current models. The learning possibilities are enormous, for example: acquire new educational materials that can help the student and complement the task to be completed; acquire information on the student, such as their preferences and cognitive styles, in order to adapt to them; change their educational content in the sense of adapting to the student; amongst other possibilities. Many works related to the implementation of learning in agents exist and these can be useful.

Future works focusing on mechanisms and methods of coordination and cooperation between agents can bring the ability for self-organisation to the society of ILOs with a view to making richer learning experiences available. Together with the ability for communication, the use of coordination mechanisms and cooperation allow for the emergence of complex behaviours between the ILOs for more complete educational experiences.

By analyzing all these characteristics we can highlight that the potential for use of the ‘learning object agents’, the ILOs, is enormous. This is due to the flexibility and the dynamism that can be reached with them, which is greater than can be reached through use of current learning objects. As a consequence, learning environments based on these can be more flexible and more dynamic also.

3 The Architecture

For the development of this work, the architecture proposed by Gomes (2005) was employed in the construction of Intelligent Learning Objects. This section presents this architecture, as well as the related framework.

This architecture defines the internal processes that make the interaction of the agent with its environment possible. It involves three basic components.

A sensing component is responsible for perceiving modifications in the environment such as, for example, receiving messages and occurred events in a graphic user interface. A reasoning component is responsible for the behavior adopted by the agent related to the occurred event. And the actuator component, which is responsible for the execution of agent actions in the environment. Figure 1 illustrates these three components.

This architecture can be characterized as hybrid as it contemplates both cognitive and reactive aspects. These aspects were modeled in different layers where each one of the three related components corresponds to one of the layers of the process.

The Learning Environment proposed in this work was implemented from the framework presented by Gomes (2005), which, in turn, was implemented from a set of Java

![Figure 1. Components of the proposed architecture.](ART04_Wilges.pmd)
classes. The relationship among the agents is characterized by a client-server model in which the agent who initiates a dialogue (requests a service) is the client and the agent who responds to the dialogue (providing a service) is the server. For this reason, a pair of tasks exists for each one of the dialogues. One of them should be used by the server agent and the other by the client agent.

The behavior of each type of task is standardized as the flow of messages is the same between all the dialogues. For this reason, the same protocol always used: the called FIPA-Request protocol. This standardization generates an abstract class for each type. All the tasks that deal with dialogues are, therefore, extensions of one of these classes.

In this framework a relationship exists between the abstract class used for the implementation of the server part and of one of the tasks that extend it. In this case, the class that implements the HandleGet_Metadata dialogue. In this way, only the constructor methods startTask() and process() are necessary. The process() method is abstract and, in it, the behaviour of the Task should be inserted generating a large part of the reasoning of the task in question.

A relationship also exists between the abstract class that must be used in the implementation of the client part of a dialogue and of one of the classes that extend it, in this case the class that implements the ReqGet_Metadata dialogue.

In the ReqTask classe, the sendRequest() method is responsible for building the initial message from the dialogue. The argument for this method corresponds to an object from a class which implements the Action interface. This should be implemented by all the predicted actions in the ontology of this framework. That is to say, the argument of this method is the action to be requested.

In treating the ontology of this framework, it was established that the representation of the content of the messages is achieved through the FIPA-SL language. In this way, all the elements defined in the ontology should also possess a form of FIPA-SL representation. As a means of facilitating this codification, each ontology element is responsible for producing its own representation in FIPA-SL. This responsibility was inserted out of a need for all ontology elements to be implemented into an interface containing a toSL() method. For each type of element, action, concept or adjective, a specific base interface exists.

The ILOs from this framework can be divided into classes in accordance with their communicative possibilities. With this division in mind, an ILOAgent class and an IdleTask class are necessary for each ILO class. Three classes of ILO exist: the ILOAgentC1, which is responsible for the requests for metadata to be sent; the ILOAgentC2 class which is, in fact, an extension of an ILO from class 1 although it has the possibility of answering requests for information on the student; the ILOAgentC3 which is an extension of the ILO from class 2. The difference here lies in the possibility of this ILO accessing services by other agents.

In the part, which corresponds to an LMS Agent, only the classes which carry out the basic services for the agent to be available are implemented. In this way, the implementation of an IdleTask, the LMSIdleTask, which identifies actions relevant to the dialogues of this agent, and the classes which implement the server part of the dialogues, are available. In order to implement a more sophisticated LMS agent, all that is required is that the classes are extended.

4 Modeling of the Learning Environment

In this section, the learning environment that was modeled according to the specifications of the presented framework is described. The Learning Environment used in this work for the validation of the architecture presented in the previous section, was implemented in previous work (Lucas et al., 2005). The environment is constituted by an auxiliary system for students in initial stages of learning of the basic mathematical properties of multiplication and addition. The environment is composed of two agents: an Animated Pedagogical Agent (APA) and a Calculator Agent. The Animated Pedagogical Agent (APA) was modeled in order to present the general agent characteristics in the role of a Learning Management System – LMS (LTSC, 2006). Its main function is the management of Learning Objects through its metadata and interaction with the student. The agent was developed in Java language, using the Swing library, which encompasses classes and interfaces capable of providing animated character visual features to the pedagogic agent. It also uses corporal expressions realized through animation of images and exhibits messages in a dialog box like a ‘balloon’ for cartoon characters in comic strips.

The Calculator Agent is a kind of Learning Object representing a calculator where the student carries out the calculations with the ability for communication of the results of the student interaction with the APA. This carries out an evaluation of what was done by him/her.

In the following section, some details of the modeling and implementation of these agents will be presented. Both the Pedagogical Agent and the Calculator Agent are compatible with the FIPA-OS platform and all the mechanisms of communication occur through this.

4.1 The Animated Pedagogical Agent

This agent was implemented in an architecture constituted by 5 classes: the Apa class, responsible for calling
the character in the `main()` method; the ApaGUI class, responsible for its graphic interface; the Animator class, responsible for the character’s movements; the Dialogue class, which constitutes the animated character’s dialog box and one more class which contains all the constants used by Apa. Such an agent has three tasks: ResultReasoning, InstructionReasoning and EvaluationReasoning. The ResultReasoning Task is triggered by a message sent by the SendResult Task, which belongs to the CalcAgent class of the Calculator. This is responsible for the reasoning to analyze which properties can be demonstrated, such as instantiation of another task, the InstructionReasoning. The Reasoning Task to which is presented two types of action, in turn, triggers this. The first action consists of an analysis of what was carried out by the student (from the parameters sent by the ResultReasoning Task) and printing, in the character’s dialog box, an instruction to the student, specifying which mathematical properties can be proven from the initial calculation. The second action consists of printing an instruction to the student from the parameters sent by the EvaluationReasoning Task, informing the student of the properties proven to be correct or, in the case where they were not proven, specifying a reason for the error. Finally, the EvaluationReasoning Task only evaluates if a property has been proven. Figure 2 displays a diagram of functions, which illustrates the standard roles of the tasks described above.

In order for the Pedagogical Agent to also work as a LMS, it was necessary for the Apa class to extend to the LMSAgent class of the framework. In addition, it was essential to add two tasks responsible for initiating the dialogue server: the get-learner-lms task and the put-learner-lms task, and a third which is responsible for initiating the client: the put-learner-ilos task. Also added were the tasks responsible for registration of the FIPA-OS platform. With this, the pedagogical agent can execute and operate with an LMS agent, in the characteristics proposed by this project, offering all the basic services of an agent of this type such as, for example, actions relative to the dialogues of this agent.

4.2 The Calculator Agent

The Calculator Agent is composed of the CalcAgent and CalcAgentGUI classes, responsible for the graphic interface of the calculator. This class possesses only one task, the SendResult, which is triggered at the moment in which the student presses the key with the equal sign on. This action corresponds to the sending of a message to the Apa agent in which the mathematical calculation carried out by the student is contained.

The CalcAgent class extends to the ILOAgentCI class of the framework, to which were also added the client of the dialogue task: put-learner-lms and the dialogue server task: put-learner-ilos. From there, the calculator then represents the general characteristics of an ILO in the characteristics of the proposed architecture.

Figure 2. Diagram of task functions.
4.3 The Animations Agent

In order to better evaluate the architecture proposed, a second ILO was implemented. This agent controls a set of animations created with flash technology with the aim of providing, in a pleasant way, a series of information to the student, which is necessary to facilitate the process of teaching-learning. The function of this agent is to teach to the student the demonstration of a determined mathematical property. The OPA/LMS can call this agent when it feels it is necessary to do so. This may be in the case of a student presenting difficulties with the demonstration of a determined mathematical property, for example.

The Animations Agent has an internal architecture similar to that of the Calculator Agent. For implementation of this, extension of the ILOAgentCI of the framework was necessary as well as the addition of the same tasks described in the adaptation of the Calculator Agent.

5 The use cases

The following section describes the interactions that occur in the system for its several use cases in order to make the specification of this project.

5.1 First use case

The use case that describes the principle of the application occurs when a student opens the application through the AgentLoader of the FIPA-OS. The animated character displays a welcome message to the student and sends a message to the Calculator Agent instructing this to unfreeze the keyboard for the student to type the first operation. Figure 3 presents this use case of the system.

5.2 Second use case

This use case corresponds with the first mathematical operation carried out. It is initiated as soon as the APA sends a message to the Calculator Agent requesting that the keyboard, present in the graphic interface of the calculator, be unfrozen to enable the student to carry out the first operation. Once this has been achieved, the Calculator Agent sends the results to the APA who then sends a request, asking for the Calculator Agent to lock the keyboard once again. Finally, the APA evaluates the operation carried out by the student. This use case in the learning environment is illustrated in Figure 4.

5.3 Third use case

This use case begins when the APA sends a message to the Calculator Agent for it to unfreeze the calculator in order to enable the student to enter the data for the second mathematical operation. Once this is done, the Calculator Agent sends the operation to the Animated Pedagogical Agent, which evaluates whether the properties were applied correctly and prints an instruction to the dialog box stating which mathematical properties were proven correctly. In cases where the properties were not proven, a specification for the error is given. Figure 5 illustrates this use case.

5.4 Fourth use case

A further use case of this system involves the presentation of animations, which always occur after completion of the second use case and in the case of error in the demonstration of a property. In this case, the Animated Pedagogical Agent sends a message to the Animations Agent for it to present the correct form of demonstrating a determined property. This case is exemplified in figure 6.

Figure 3. Illustration of how this first use case in the Environment is developed.

Figure 4. Execution of the first operation in the Learning Environment.

Figure 5. Illustration of the third use case.
5.5 Fifth use case

The last use case in this system aims to reflect on how the restart of the process occurs. This case occurs immediately following an evaluation by the APA on the results of the second operation. It sends a message to the student about the results obtained. In order to restart the process, the Animated Pedagogical Agent sends a message to the Calculator Agent for this to initialize all the data. Figure 7 illustrates this case.

6 Conclusions

The main objective of this work was to validate the framework proposed by Gomes (Brenner et al., 1998), through the implementation of a system, which works with the exchange of messages and defined ontology in the proposed architecture, so that the presented framework worked in a learning environment built with agent-based Learning Objects.

Another important defined characteristic is that an Intelligent Learning Object should, as well as having to present the characteristics exposed above, have a pedagogical purpose. That is to say, it should be used in the sense of generating learning experiences for a student. For this reason, the application presented was adopted as the Agent responsible for the animations, together with the Calculator Agent, as it seeks to generate learning experiences in a pedagogical sense.

For these reasons two types of agents were implemented: an LMS agent, an abstraction from a learning management system in the form of an animated character, and the ILOs themselves, correspondent with the Calculator Agent and the Agent responsible for the Animations.

One of the difficulties encountered in the development of this work is related to the implementation of an agent-based Intelligent Learning Object. Although the framework, and even the FIPA-OS itself, facilitate construction of these agents, it can not be denied that implementation of an ILO is far more complex than implementation of a simple learning object. However, this disadvantage is minimized by the possibility of being able to implement learning objects, which are more pedagogically advanced.

It can be concluded through this work that the architecture proposed was validated in terms of the potential of the efficiency of the proposed framework. Although only two ILOs were used to interact with the APA, it is possible to envisage the possibility that the architecture represents in the sense of increasing...
interoperability amongst the components in the learning environment. We believe that the introduction of the ILO responsible for animation within the environment guarantees greater efficiency to this environment in terms of adaptation. By aggregating direct characteristics from the agents, this ILO demonstrated a wider capacity for learning, as it would only be requested when a student presented a difficulty in the demonstration of any of the shown mathematical properties.

Through this work, verification of the potential to achieve a convergence between the Learning Objects approach and the architecture of agents was sought, thus making possible the construction of learning environments with instructional material that can be reused and with self-adapting capabilities.

New tests in real learning situations with more complex environments need to be carried out. It is hoped, however, that this work represents a first step in the evaluation of this proposal thereby contributing to its further development.

References
