A SOA-based Middleware to Integrate Chatterbots in e-Learning Systems

Manuel Caeiro, Jorge Fontenla, Fernando Mikic, Roberto Perez, Martin Llamas, Juan C. Burguillo
Department of Telematic Engineering
E.E. Telecomunication, Universidade Vigo
Vigo, Spain
Manuel.Caeiro@det.uvigo.es

ABSTRACT

In recent years, artificial intelligence conversational agents, usually known as chatterbots, have become very popular in the Internet. In this paper we show how chatterbots can be integrated in e-learning systems. To perform such integration the Service Oriented Architecture programming paradigm is adopted. A middleware is provided for enabling the integration and reuse of chatterbots by e-learning systems supporting a tight control of their operation. Such middleware takes into account several issues such as user authorization, instance creation, data transfer to and from the chatterbot, permission assignment to users and subscription to events. Our approach is applied to the specific case of TQ-Bot, which is used to track and supervise the progress of the students, and to provide answers orienting them to the more appropriate course contents.

1. INTRODUCTION

The adoption of new technologies to support education is continually increasing. The Internet's functionality and capability is being applied to support an increasing number of courses at different levels (from K-12, to higher education and lifelong learning), in a broad range of disciplines, and in different contexts (e.g. distance learning, blended learning, traditional in-class education). Artificial Intelligence (AI) is usually considered as a key technology domain in the development and adoption of e-learning systems. Since the 1980's many research projects have been devoted to the development of Intelligent Tutoring Systems (ITSs) [8], intelligent agents, and more specifically the use of conversational agents, usually called chatterbots, which allow the communication with users in natural language.

A.L.I.C.E. (Artificial Linguistic Internet Computer Entity) [2] has been one of the most ground-breaking projects in the field of AI during the last years. A.L.I.C.E. is the project that produced the AIML (Artificial Intelligence Markup Language) through which is possible to develop software chatterbots [21]. A.L.I.C.E. has won the "Loebner Prize in Artificial Intelligence Contest" (based on the Turing test) several times. An A.L.I.C.E.-like chatterbot can be used as a tutor in an e-learning system to provide tutoring and evaluating support. In this paper we use an A.L.I.C.E. based chatterbot named TQ-Bot, which is used to track and supervise the progress of the students, and to provide answers orienting them to the more appropriate course contents.

A main issue in the use of chatterbots is their integration in e-learning systems. Chatterbots are usually developed ad-hoc and with no interoperability support. Today we can find many bots in the literature [6, 2, 25, 18], but it is very difficult to use them in contexts different from the one they were conceived for. This can be seen as a reusability problem that should be solved.

In this paper we show a solution based on the Service Oriented Architecture (SOA) programming paradigm that enables the integration of chatterbots into e-learning systems. This work extends some standardization initiatives in the e-learning domain for the integration of third-party tools [1]. Our solution comprises a middleware, interfaces and protocols to achieve a hard integration of third-party tools and e-learning systems involving transparency and privacy requirements key for final users. As a result, it is provided an infrastructure that can be used to support the integration of chatterbots in e-learning systems. In this paper we show how a specific chatterbot (TQ-Bot) is integrated into a SOA-based LMS using this infrastructure.

2. BACKGROUND

Nowadays, the most common e-learning systems are Learning Management Systems (LMSs) [32][33]. LMSs are deployed as holistic platforms intended to manage all the issues involved in distance learning. These issues comprise authoring, assessing and delivering tools to provide specific functionalities (e.g. profile management tools, productivity tools, communication tools). In their first attempts, LMSs were essentially repositories with lots of documents but very basic functionality. However, these platforms evolved into rich environments where students can communicate, collaborate, access to multimedia files, participate in virtual worlds, subscribe to podcasts, writing wikis, playing games, etc. The EduTools [31] review analyzes 39 different LMSs.

In spite of the advantages of LMSs, there exist some important drawbacks that should not be overlooked. The lack of a tutor figure to pay specific attention to a individual students is one of these drawbacks. Here is where a chatterbot can play an important role. A chatterbot can be dedicated to tutoring students, taking advantage of AI techniques, and to offer a kindly interface to the users. This bot can help students at any time of the day, any day of the week. It does not get bored or loses its patience due to the students' attitude, and it can attract and keep students' attention because it supposes a technological innovation. Even to some degree, a chatterbot can make the student feel more comfortable than just surfing through the learning resources and tasks.

Attending to the development model, current LMSs can be grouped into two main categories [10]. The first category is about open source initiatives (such as Moodle, .LRN, Sakai, dotLRN, ATutor, Whiteboard), which are build over extensible frameworks that let implementers adjust and modify the systems to match their specific needs. The other category involves proprietary solutions (such as Blackboard, Gradepoint, Desire2Learn, Learn.com). These systems support extensions by providing software developers with "hooks" to tie third-party software into the LMS. Nevertheless, there is not any solution that can be applied in a general way.

The need for extensibility solutions in e-learning systems has led many organizations to develop and publish several standards and recommendations. Some standards regard the definition of layered and decoupled architectures [10]. Example of this are the E-Learning Framework (ELF) [14], the IMS Abstract Framework (IMS-AF) [27] and the Open Knowledge Initiative (OKI) [22]. Among the targets of these specifications we can find the modularization of functionality in e-learning systems by the identification of well-defined core components, interfaces and APIs. These elements are defined to support the interoperability with the other elements via Web Services, and grouped according to their functionality [10]. However, the practical adoption of these works is very limited, and therefore they are regarded just as theoretical frameworks. Other kind of specifications (IMS General Web Services [26], IMS Tool Interoperability [1] and IMS Common Cartridge [29]) are related to the extension of the functionalities of current e-learning systems by means of their interconnection with third-party components during runtime, using broadly-accepted Web technologies and paradigms such as SOAP, WSDL, UDDI [30], Ajax and Comet [9], Saas [28], IaaS [13], and Cloud Computing [15]. Despite their heterogeneity, these solutions present well-known advantages in terms of interactivity and scalability.

3. OUR E-LEARNING SYSTEM

3.1 The Educational Scenario Concept

The educational scenario is the fundamental unit for constructing complex courses. The most relevant elements for defining an educational scenario are participants, which are enrolled into scenarios; goals, which declare learning objectives; environments, which aggregate learning resources and tools (in which bots are included); and temporal deadlines, which indicate the temporal limit for fulfilling goals. Therefore, an educational scenario encapsulates a fully functional unit of learning.

The life-cycle of educational scenarios can be divided into the following stages: design time, instantiation time and runtime. The concept of scenario is, therefore, twofold: it can be whether the model created during design time, or a concrete instance, with concrete participants enrolled into, and with certain temporal constraints as well. These concepts are illustrated in Figure 1. In design time, the author creates the model of the scenario using an authoring tool. In the example, a scenario with human participants and chatterbots is depicted, as well as a lab environment with some tools: a microscope and some books on inorganic chemistry. In instantiation time, a new instance of the educational scenario is created from the model in order to handle a particular case. In the example of the figure, three participants are grouped and enrolled into the first educational scenario instance: Arthur, having a teacher's role; Bob and Carol, having a learner's role; and Bob_Chatterbot and Carol_Chatterbot, having a consultant role. In the same way, Dan, Ernst, Frank, Dan_Chatterbot and Ernst_Chatterbot are grouped and enrolled into the second educational scenario instance. The creation of a new scenario instance entails creating instances of its containing elements: a new environment instance as well as instances for tools and chatterbots into the environment. Finally, in runtime, participants access to environment instances and make use of tools and chatterbot instances. Notice that every group of participants use its own scenario instance.

In the following subsection we detail a general architecture to support the life-cycle of educational scenarios, from design time to runtime.

3.2 Technologies

The e-learning system follows a typical three-tier architecture: presentation, business logic and database.

The presentation layer of our LMS is inspired in Moodle [20], programmed in PHP.

The Business Logic Layer is based on the PoEML [7], which is an Educational Modeling Language and, as such, it allows to describe scenarios, groups of participants, tools, resources, and the rest of elements in educational scenarios. This layer enables the definition and execution of learnflows [24] involving participants, learning goals, temporal constraints, etc. This layer is implemented as a Java Web Application running on Tomcat [3].

The Business Logic Layer is integrated in the overall system through a well-defined interface that is based on Web Services. This approach provides the maximum level of interoperability in web-based scenarios. In order to make Web Services accessible to presentation modules, we use the functionalities provided by a SOAP engine, Axis [4]. The functionalities that the Business Logic Layer provides are published in a WSDL file. The service methods serve for passive information retrieval, communication of events, and ad-hoc changes in instances. The JavaToWSDL tool provides for automatic WSDL generation from Java code. The WSDL file is automatically generated from the Java class containing the declaration of Web Service methods as a Java interface

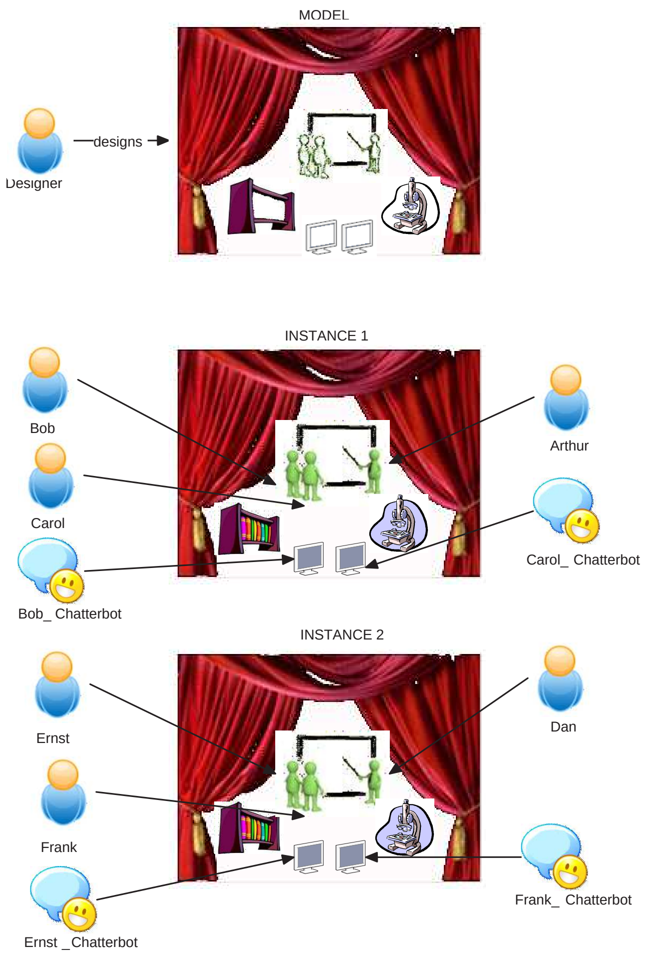


Figure 1: Design time and runtime of an educational scenario.

definition.

In the Presentation Layer we use the NuSOAP [5] library, which facilitates the consumption of Web Service methods. After retrieving the WSDL file containing the definition of Web Service methods, the Presentation Layer is able to declare a client and request service methods from the Business Logic Layer.

The Database Layer is implemented on Oracle [23]. We have chosen Oracle because of its good out-of-the-box scalability support, which is an important concern in big e-learning deployments, as those of universities supporting distance learning courses.

4. SEAMLESS INTEGRATION OF CHATTERBOTS IN E-LEARNING SYSTEMS

Given the previous architecture of a generic e-learning system, our objective in this section is to describe an extension mechanism in order to complement the basic features of the system with the aid of third-party tools, in this case chatterbots. We consider that the integration of new functionalities must be as tight as possible, and must be carried out with minimum changes in the legacy systems. In the following sections we give some definitions concerning the level of integration of a third-party tool in an e-learning system, and then we provide a close look to the architecture we have developed for integrating of chatterbots.

4.1 Soft and Hard Integration

At this point we consider two opposite alternatives for integrating third-party tools in e-learning systems, which are also considered in [17]:

- Soft integration of third-party tools. The e-learning system functionality can be extended through a hyperlink to an (external) third-party component. When the user clicks on it, the graphical user interface of the tool is displayed. From this point, users are operating a tool that the e-learning system cannot control by any means. Therefore, a new functionality is included but it does not work in coordination with the core system, resulting in a very “soft” integration.
- Hard integration of third-party tools. It includes soft integration, but providing the e-learning system with a more comprehensible control over the integrated tools. We describe in the next paragraphs our proposal for such a comprehensible control.

Hard integration allows the e-learning system not only to link the application, but also to supervise and alter the workflow of the tool as required, in order to adapt it to the concrete requirements and limitations of the course and its users.

As discussed in [7], the control of the operation of a learning tool (a chatterbot in this paper) to achieve hard integration in e-learning systems involves the following issues:

1. Creating a chatterbot instance for each user. For example, in an “Chemical” subject a chatterbot instance can be created for helping a learner in the course.
2. Transferring from the e-learning system to the chatterbot all those data that the user may need in order to carry out his/her tasks. In the previous example the student can obtain additional content asking the chatterbot. Previously, the chatterbot received such content from the e-learning system.
3. Establishing some access permissions over these data and the chatterbot functionality. In our example, the student may be assigned a configuration permission to change certain features of the bot, for example its name or background image.
4. Subscribing to events result of the work with the chatterbot. For example, the e-learning system may be interested in knowing when the student access to some specific contents provided by the bot.
5. Authorising the user to access the chatterbot instance. In our example, the student may not have access credentials at the chatterbot, in whose case the e-learning system has to grant him/her access as guest user.

6. Activating an action in the chatterbot according to the information provided by the events triggered. For example, the LMS activate a message in the chatterbot to inform the learner that 5 minutes remain to finish the task.

4.2 The Generic Tool Adapter

The Generic Tool Adapter has been posed as a software component to extend the functionalities of an e-learning system by enabling the integration of third-party tools in a hard way. In the context of this research work a chatterbot is considered as a special kind of third-party tool. This adapter has been developed at our research group to allow e-learning systems to import, control and manage external tools that complement the functionalities of the LMS. The aspects covered by this adapter involve:

1. Authorization granting. A single sign-on mechanism, named *Reverse OAuth* [11], included as part of the Generic Tool Adapter, has been developed in order to authorize users (e.g. learners and teachers) to access the tool without requiring additional sing-ins. This is especially interesting when users have already authenticated after the e-learning systems and, from their point of view, additional authentications after the tool should not be necessary.
2. Instances management. The Generic Tool Adapter includes resources devoted to control the instances of the tool. We understand by instance of a tool a working environment along with a graphical user interface, associated to several files to manipulate, and a set of users allowed to access it. Several methods are included to control the creation and deletion of concrete tool instances, and to add and remove users to tool instances.
3. Data transfer. A mechanism to exchange data between the LMS and the tool, either single data values or full backups of user data. This functionality allows the e-learning system, for example, to submit configuration files to a chatterbot and to get a log of conversations in the chatterbot.
4. Permissions assignment. A functionality is included in order to set access permissions to specific users over concrete parts of the tool. This functionality provides an straightforward mechanism to differentiate the different roles of teachers and students (e.g. students may be allowed to communicate with a chatterbot and teachers, additionally, may have permissions to change its configuration).
5. Event subscription. This feature allows the e-learning system to subscribe to particular events triggered by the tool in response to specific actions carried out by its users. This feature is specially useful in e-learning environments, where the external system must be "in touch" with what happens inside the tool in order to track, evaluate and help students.
6. Specific methods management. Finally, the Generic Tool Adapter provides mechanisms to alter the workflow of the tool. This category includes all those methods that do not fit in the previous five categories for providing functionalities that are very specific and dependent of the type of tool.

The Generic Tool Adapter features a standardized syntax to invoke its methods, i.e. it implements the Generic Tool Interface. This interface is further decomposed into six sub-interfaces, according to the six aspects of hard integration enumerated above. Table 1 summarizes some of the methods of the Generic Tool Interface, and classifies them according to the sub-interface they belong to.

4.3 Generic Tool Adapter protocol stack

The internal architecture of the Generic Tool Adapter is based on the well-accepted approach to software design of protocol stacks. Figure 2 depicts a representation of the Generic Tool Adapter as a refinement of the TCP/IP protocol stack where the Application layer has been further divided into three sublayers, and the Generic Tool Adapter corresponds to the "Integration Manager" and "Integration Protocol" sublayers. As in the standard TCP/IP protocol stack there is a (virtual) direct communication between analogous (sub)layers, so that Integration Managers communicate with Integration Managers and Integration Protocols with Integration Protocols.

The Integration Managers implement the methods of the Generic Tool Interface (see Table 1) and, together with the Integration Protocols, form the Generic Tool Adapter (see Section 4.2). There are six Integration Managers and six Integration Protocols altogether. These Managers and Protocols are grouped in pairs, dealing with a specific issue of hard integration (see Section 4.1). When a method of the Integration Manager is invoked it serializes the call and forwards it to the corresponding Integration Protocol, which in turn submits it to the remote Integration Protocol. At this point, the remote Integration Protocol passes the call to the remote Integration Manager, which executes the action requested.

5. INTEGRATING TQ-BOT

In order to prove the usefulness of the Generic Tool Adapter in extending the functionality of an e-learning system we decided to apply it to integrate TQ-Bot [19]. TQ-Bot is a chatterbot based on AIML and dedicated to tutoring students, taking advantage of AI techniques and offering an appealing interface to users. This section introduces the functionalities and underlying architecture of TQ-Bot, and provides a thorough description (both static and dynamic) of the different elements of the system resulting from the combination of TQ-Bot and a generic e-learning system.

5.1 TQ-Bot

TQ-Bot is a virtual assistant designed for tutoring tasks, helping students in the e-learning process within an e-learning system. More specifically, using TQ-Bot students are able to auto-evaluate their knowledge and skills and to ask for specific course contents. It can attract and keep students' attention because it supposes a technological innovation. Even to some degree, TQ-Bot can make the student feel more comfortable than just surfing through the learning resources and activities.

Sub int.	Method	Input parameters	Output parameters	Description
1	grant	resourceURI, expirationTime, username	authID	Grants access to a resource given its URI, the expiration time and the username of the beneficiary of the authorization. Returns an identifier for future references to the authorization.
1	revoke	authID	result	Revokes a previous authorization given its authID. Returns an error code, if any.
2	createInstance	name	instanceURI	Creates a new instance given its name. Returns its URI.
2	deleteInstance	instanceURI	result	Deletes an instance given its URI. Returns an error code, if any.
3	getDataElement	dataURI	data	Requests a data element by its URI. Returns its value.
3	setDataElement	dataURI, data	result	Overwrites the current value of the data element given by the parameter dataURI with the value contained in the parameter data. Returns an error code, if any.
3	getBackup	instanceURI, incremental	data	Requests a backup copy of the data of an instance given its URI. It can be a complete or an incremental copy. Returns the backup copy.
4	grantPermission	permission, username, dataURI, expirationTime, instanceURI	result	Grants the given permission to a user over a particular resource. If the parameter dataURI is not present, it applies to all the resources of the instance given by the parameter instanceURI. Returns an error code, if any.
4	resetToDefaults	username, dataURI, instanceURI	result	Resets the permission of the given user of the given instance over the given data element to their default values.
5	subscribe	event, instanceURI, compact	result	Subscribe to the given event. If the parameter instanceURI is present, the subscription only affects to the events that take place within the given instance. If the parameter compact is present, similar events are grouped and sent in a single message. Returns an error code, if any.
5	notify	event, instanceURI, username	result	Given a username and an instance URI he belongs to, notifies an event to the user. Returns an error code, if any.
6	invoke	methodName, parameterList	data	Invokes the given remote method with the given parameters list. Returns the result in a serialized format.

Table 1: Generic Tool Interface method summary.

TQ-Bot is an AIML-based chatterbot, a type of conversational agent (a computer program) designed to simulate an intelligent and natural-language conversation. It processes the users' inputs and consults its knowledge base to make a response that imitates the human's one.

AIML is an XML based programming language and it is widely used in the development of software agents that communicate with their users in natural language (the programming language AIML was developed by Dr. Richard Wallace and the A.L.I.C.E.bot open source community among 1995 and 2000). AIML is a text file with a specific structure, which constitutes the knowledge base of the chatterbot. The "categories" are the fundamental knowledge basis, and they consist of at least two elements: the "pattern" and the "template". In general, the performance of AIML is based on a stimulus-response model, in which the stimulus (the user's input) corresponds with the "pattern", and the response (which the chatterbot will show to the user) will be its associated "template". All these actions, about looking for the adequate pattern and showing the related template, will be carried out by a data treatment engine, of which there are many versions (Program D, Program E, etc.).

TQ-Bot has been developed as a PHP application based on Program E [16], which is the PHP implementation of the AIML interpreter. TQ-Bot also uses AJAX (Asynchronous JavaScript And XML) technology, that enables to make interactive applications or RIA (Rich Internet Applications). This technique enables our bot to maintain an asynchronous communication with its server in the background, and so, it is possible to make changes on the chatterbot interface. This means a significant improvement of the interactivity.

Students interact with the bot through the BUI (Bot User Interface), which consists in a pop-up window with a text area reflecting the conversation and a text box to introduce new requests. The bot obtains input data from this BUI and searches into its knowledge source appropriate content to reply. This content is provided during the configuration of the chatterbot instance.

If the bot does not detect any input related to the content of a course, it replies to the student with an expression taken from its general knowledge base. Once the bot detects a reply from the student, where he/she has used a special keyword (related to a learning resource of the course), the bot retrieves the previously established association and processes the learning path. All needed information is found at the database tables, and TQ-Bot shows an answer consisting of:

- The resource's abstract.
- Extra information about the resource: a link to all the content of the course related to the concept that the student was asking for.
- Related information: a set of links to any type of information related to the resource that the bot has found.
- Scoring the answer: the bot offers to the student the possibility of ranking the given answer.

TQ-Bot also enables to auto-evaluate and monitor student progress. While a student is talking to the bot, he/she can request several activities (see Figure 3):

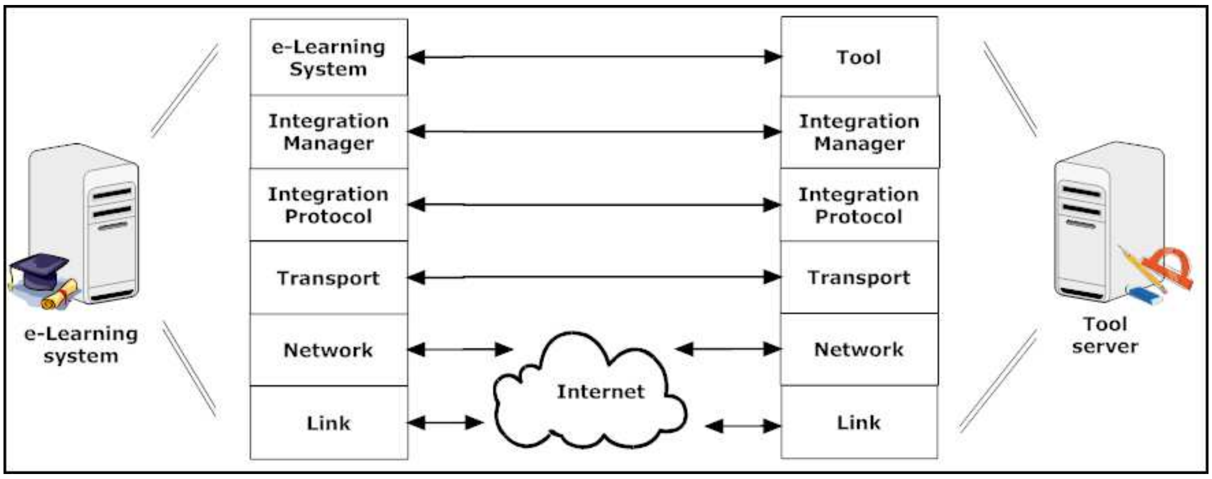
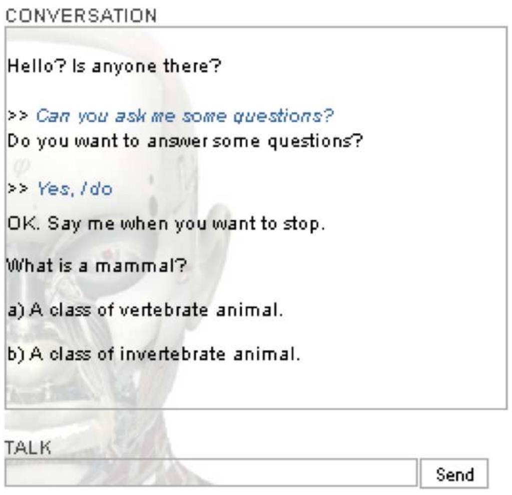


Figure 2: Representation of the Generic Tool Adapter as a protocol stack.


CONVERSATION

Hello? Is anyone there?

>> *Can you ask me some questions?*
Do you want to answer some questions?

>> *Yes, I do*

OK. Say me when you want to stop.

What is a mammal?

a) A class of vertebrate animal.

b) A class of invertebrate animal.

Figure 3: Answer of the TQ-Bot.

- To ask for a test: the bot chooses the first from all available tests that the student has not done yet.
- To ask for a personalized test: the student must choose the number of questions to be included in the test and the bot composes it.
- To ask for questions that do not belong to any test (free questions): the bot starts to ask questions and keeps on doing it until the student wants to stop.

Finally, we would like to point out that the student can ask for a clue to answer a question, and that this fact penalizes his/her final score.

5.2 Global Architecture

In this section we describe the final architecture that allows the integration of TQ-Bot in an e-learning system. The architecture of TQ-Bot, the Engine, and the Generic Tool Adapter are glued together by means of the Chatterbot Binding Adapter and the Creational API. Therefore, this section is devoted to describe these two elements. The result is depicted in the UML component diagram of Figure 4.

The Creational API has been posed to allow a programmatic management of the bot. Originally, the logic of TQ-Bot had been designed together with a graphical user interface that allows its configuration and management by users (namely, a teacher). This approach proved to be tiresome when the teacher has to configure a large number of instances of TQ-Bot for its students. Therefore, we defined the Creational API to enable an automated configuration of the bot by the e-learning system.

Property	Value
name	TQ-BOT
gender	male
master	Fernando Mikic
birthday	January 1, 2007
birthplace	University of Vigo
favouritebook	I, Robot
favouriteband	Smashing Punkins
favouritesong	Stairway to Heaven
favouritemovie	Matrix
forfun	Surfin' the WWW
language	english
image	angel.jpg

Table 2: Vocabulary used to configure TQ-Bot.

The Creational API provides the following features, in accordance with the six aspects of hard integration described in Section 4.1:

1. Authorization granting: transparent access for users to the TQ-Bot server.
2. Instances management: automated creation and deletion of instances of TQ-Bot, and addition and removal of participants to specific instances.
3. Data transfer: methods to allow the e-learning system to read and post messages in a TQ-Bot instance. An excerpt of the vocabulary used to exchange data

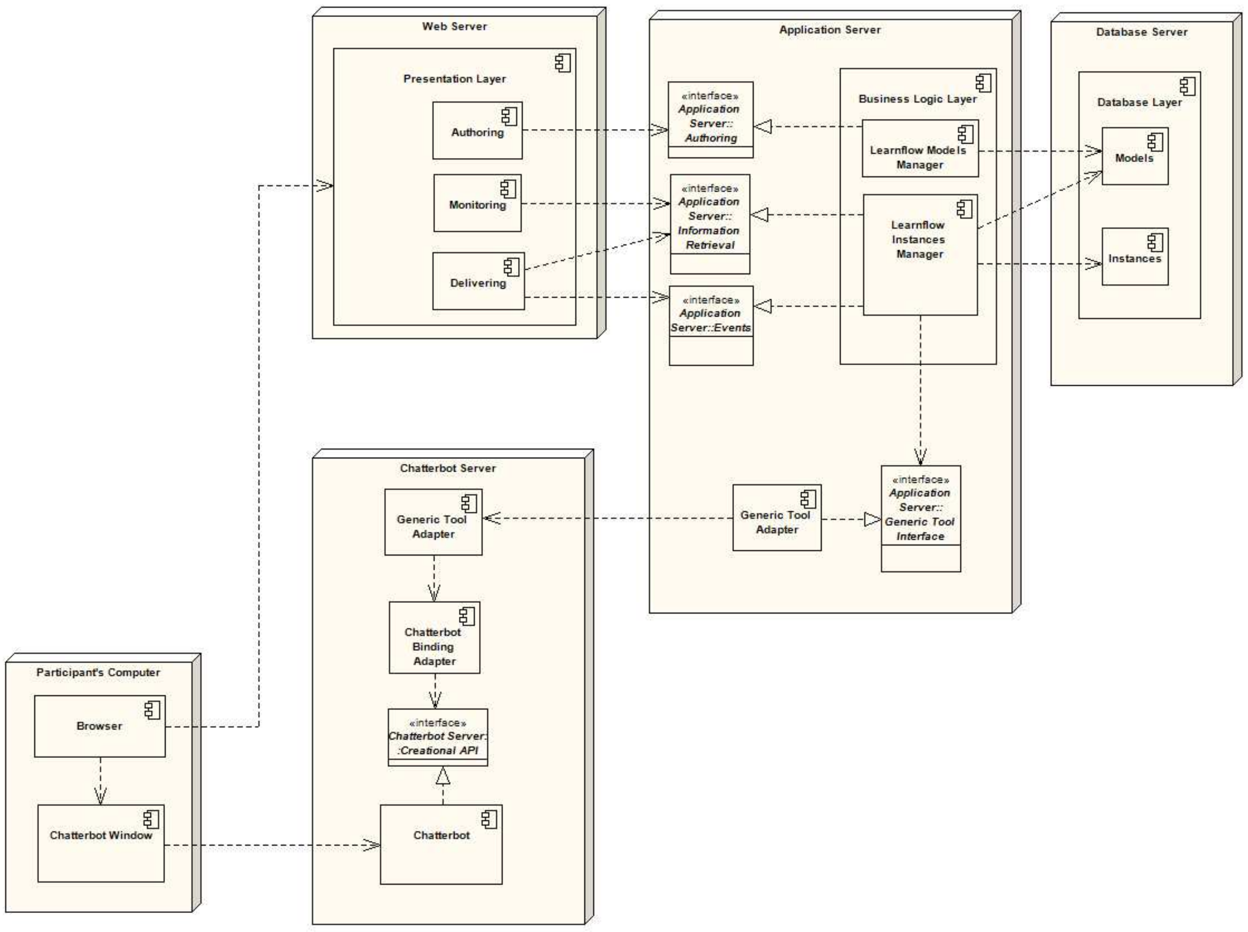

Web Server
Presentation Layer
Authoring
Monitoring
Delivering
Application Server
«interface» Application Server:: Authoring
«interface» Application Server:: Information Retrieval
«interface» Application Server::Events
Business Logic Layer
Learnflow Models Manager
Learnflow Instances Manager
Database Server
Database Layer
Models
Instances
Chatterbot Server
Generic Tool Adapter
Chatterbot Binding Adapter
«interface» Chatterbot Server: :Creational API
Chatterbot
Generic Tool Adapter
«interface» Application Server:: Generic Tool Interface
Participant's Computer
Browser
Chatterbot Window

Figure 4: UML deployment diagram of the integration architecture.

between the e-learning system and TQ-Bot is summarized in Table 2. This vocabulary also include terms to inform the bot about the course structure and organization.

4. Permissions assignment: methods to assign permissions to the participants of a TQ-bot instance.

5. Event subscription: methods that allow an e-learning systems to subscribe to events that take place in a TQ-Bot instance. This is especially useful in educational scenarios where the e-learning system must be "in touch" with the interaction between the user and a TQ-Bot instance.

6. Other methods: this category includes all those methods that do not fit in the previous five categories for providing functionalities that are very specific of the TQ-Bot system. We consider, for example, configuring TQ-Bot to display a message to the users of an instance (e.g. "By the way, I remind you that only 10 minutes remain to finish the test"). In addition, there are methos to inform the bot about the course structure and organization.

The Chatterbot Binding Adapter is an intermediate layer between the Creational API and the Generic Tool Interface. The reason of its existence is that, while the Generic Tool Interface has been designed for general-purpose tools (featuring generic methods such as `createInstance()`), the Creational API features a TQ-Bot-oriented syntax (e.g. `newTQInstance()`). Therefore, the purpose of the Chatterbot Binging Adapter is to perform a conversion between both syntaxes. This is in agreement with the Adapter design pattern [12].

The conversions between the Generic Tool Interface and the Creational API carried out by the Chatterbot Binding Adapter are actually one to one, because the latter has been designed to cover a set of common needs in learning tools. The output of the Chatterbot Binding Adapter is a request that can be appropriately processed by the Creational API.

6. CONCLUSIONS

During the last years LMSs have become very popular e-learning systems. They are used by academic institutions and companies to support learning programs and educational activities. Nevertheless, there are many problems and limitations that remain to be solved in LMSs. A main issue is related with the isolation of learners and the lack of a tutor figure that provides companion and guidance orienting the student to the more appropriate course contents. The use of an artificial intelligence entity (an AIML-based chatterbot) can provide this functionality.

The key contribution of this piece of research is a middleware to integrate chatterbots in e-learning systems. This middleware has been developed in a generic way, not just focused on chatterbots but also on other tools that can be used in e-learning: simulators, games, production tools, etc. Eventually, all these tools share some basic integration needs (managing instances, assigning permissions, etc.). Our middleware provides support to these needs following a modular approach as well as it supports specific issues on particular tools. In this paper it is shown how this middleware can be used to integrate a chatterbot in a LMS. The final integration of the TQ-Bot was achieved through the programming on a single software component: the Chatterbot Binding Adapter. Similarly, following the same approach a broad variety of tools can be integrated in the LMS. The difficulties are on the availability of a component implementing an interface with methods as the ones of the Creational API. If this component does not exist it needs to be provided.

7. ACKNOWLEDGMENTS

Authors want to thank Spanish Ministerio de Ciencia e Innovacion for its partial support to this work under grant “Methodologies, Architectures and Standards for adaptive and accessible e-learning (Adapt2Learn)” (TIN2010-21735-C02-01) and to CYTED Program under Coordinating Action 508AC0341 SOLITE.

8. REFERENCES

[1] B. Alcom, C. Vento, and et al. IMS Tools Interoperability specification. Web site: [Last accessed: March 2011], February 2006.

[2] ALICE A. I. Foundation. A.L.I.C.E. Artificial Intelligence Foundation. Web site: [Last accessed: March 2011], 1995.

[3] Apache. Tomcat. Web site: [Last accessed: March 2011], 2000.

[4] Apache. Axis. Web site: [Last accessed: March 2011], 2004.

[5] D. Ayala and S. Nichol. Nusoap. Web site: [Last accessed: March 2011], December 2009.

[6] J. C. Burguillo. Project Galaia. Web site: [Last accessed: March 2011], 2008.

[7] M. Caeiro. *PoEML: A separation-of-concerns proposal to instructional design.* IGI Global, 2007.

[8] A. T. Corbett, K. R. Koedinger, and J. R. Anderson. *Intelligent Tutoring Systems*, pages 849–874. Elsevier Sscience, 1997.

[9] D. Crane and P. McCarthy. *Comet and Reverse Ajax: The Next-Generation Ajax 2.0.* Apress, Berkely, CA, USA, 2008.

[10] D. Dagger et al. Service-oriented e-learning platforms: From monolithic systems to flexible services. *IEEE Internet Computing*, 11(3):28–35, 2007.

[11] J. Fontenla et al. Reverse OAuth - A solution to achieve delegated authorizations in single sign-on environments. *Computers & Security*, Forthcoming publication.

[12] E. Gamma, R. Helm, R. Johnson, and J. Vlissides. *Design patterns: elements of reusable object-oriented software.* Addison-Wesley Professional, 1995.

[13] M. Hamamo. Overview of IBM Information Server - Information Integration Solutions to realize Information as a Service (IaaS). Web site: [Last accessed: March 2011], December 2009.

[14] JISC and DEST. E-Learning Framework Web site. Web site: [Last accessed: March 2011], 2004.

[15] M. T. Jones. Cloud Computing with Linux. Web site: [Last accessed: March 2011], December 2009.

[16] A. Kootstra. Program E. Web site: [Last accessed: March 2011], 2002.

[17] M. Kyng. *Computers and Design in Context.* The MIT Press, 1997.

[18] M. D. Leonhardt, D. D. Castro, R. L. Dutra, and L. M. R. Tarouco. Elektra: Um chatterbot para uso em ambiente educacional. volume 1. Renote - Revista Novas Tecnologias na Educação, 2003.

[19] F. Mikic, J. Burguillo, and M. Llamas. TQ-Bot: An AIML-based tutor and evaluator bot. *Journal of Universal Computer Science*, 15(7):1486–1495, 2009.

[20] Moodle. Moodle Web site. Web site: [Last accessed: March 2011], 2002.

[21] A. M. M. Neves, I. Diniz, and F. A. Barros. Natural language communication via AIML plus chatterbots. In *V Symposium on Human Factors in Computer Systems*, 2002.

[22] OKI. Open Knowledge Initiative Web site. Web site: [Last accessed: March 2011], 2001.

[23] Oracle. Oracle. Web site: [Last accessed: March 2011], December 2009.

[24] R. Perez et al. Enabling Process-Based Collaboration in Moodle by Using Aspectual Services. In *Proceedings of the 2009 Ninth IEEE International Conference on Advanced Learning Technologies-Volume 00*, pages 301–302. IEEE Computer Society, 2009.

[25] O. D. Pietro and G. Frontera. Tutorbot: An application aiml based for web-learning. In V. Uskov, editor, *CATE*, pages 284–290. ACTA Press, 2004.

[26] C. Schroeder, J. Simon, and et al. IMS General Web Services specification. Web site: [Last accessed: March 2011], December 2005.

[27] C. Smythe. IMS Abstract Framework specification. Web site: [Last accessed: March 2011], 2003.

[28] Trumba Corporation. Five benefits of software as a service. Web site: [Last accessed: March 2011], 2004.

[29] E. Unjhem, D. Mills, and et al. IMS Common Cartridge specification. Web site: [Last accessed: March 2011], July 2008.

[30] A. E. Walsh, editor. *UDDI, SOAP, and WSDL: The Web Services Specification Reference Book.* Prentice Hall Professional Technical Reference, 2002.

[31] WCET. EduTools portal. Web site: [Last accessed: March 2011], December 2009.

[32] P. Weaver. Preventing e-learning failure. *Training and Development*, 56(8):45–50, 2002.

[33] R. Zemsky et al. Thwarted innovation: what happened to e-learning and why. A report for the Weatherstation Project of the Learning Alliance at the University of Pennsylvania. Web site: [Last accessed: March 2011], December 2009.