Proceedings of the IASTED International Conference
Computers and Advanced Technology in Education (CATE 2012)
June 25 - 27, 2012 Napoli, Italy

PROPOSAL OF AN INNOVATIVE ARCHITECTURE FOR WEB BASED EDUCATION SYSTEMS

Rubén Peredo Valderrama[1], Iván Peredo-Valderrama[1]
[1] Superior School of Computer Science of National Polytechnic Institute
Av. Juan de Dios Bátiz S/N esquina Miguel Othón de Mendizabal, Col. Lindavista, 07738
México
rperedo@ipn.mx

ABSTRACT
This paper presents a proposal of an innovative Architecture for Web Based Education Systems based on: Software Design Patterns, Components, and Multi-Agent System (MAS). The approach improves the Learning Technology Standard Architecture (LTSA), proposed by IEEE. Maintaining compatibility with: Advanced Distributed Learning Initiative - Sharable Content Object Reusable Model (ADL-SCORM). The proposal implements several Software Design Patterns provide solutions for common problems in the development of Web Based Education Systems, focusing on maintenance and change. Another key part of the proposal are IRLCOO components, which allow: conquering complexity, managing change, and maximizing reusability. The Web Based Education Systems are: complex and dynamic, the MAS can model complex and dynamic systems.
The IEEE 1484 LTSA has four processes, which are implemented as agents in our proposal: Learner, Evaluation, Coach, and Delivery.

KEY WORDS
Architecture, Software Design Patterns, Components, MAS, Web-Based Education

1. Introduction

The Software Design Patterns are tools that allow us to manage constant change in software design and development, allowing managing changes and maximizing reusability of the software developed.

Designers and developers need to build software that is flexible and easy to maintain, supporting constant change, developing software for the real world.

The Web and more specially the Web applications must have capacity to manage change, maximizing reusability, and maintainability. The Software Design Patterns offer designs that can handle the change, allowing to make changes and reuse of the software developed. The Software Design Patterns take a set of patterns to resolve recurrent problems.

The fundamental principles of design patterns were established by the Gang of Four [1]:

- "Program to an *'interface*', not an '*implementation*'."
- "Favor '*object composition*' over '*class inheritance*'."

The flexibility in the software development is linked to the necessity to manage dependency; an inadequate handling leads to unexpected results. The interfaces decouple the code from the implementation, allowing variation in the implementation; the interfaces reduce implementation dependencies between subsystems, resulting in changes that can be done at Run-Time. The object composition adds new functionality bringing new objects; it hides the internal details, maintaining encapsulation.

The IEEE 1484 LTSA is a standard for Learning Technology designed by the Learning Technology Standards Committee (LTSC) of the IEEE. It defines an architecture for recording descriptive information linked to the learning process [2].

Sharable Content Object Reference Model (SCORM) is a standard integrated with a set of related technical standards, specifications, and guidelines. SCORM is a specification of the Advanced Distributed Learning (ADL) [3].SCORM defines a communication channel between the client side content and the Run-Time Environment (RTE).

A system can be based on a single agent working within an environment, interacting with its users, but usually are made up of multiple agents. The Multi-Agent System (MAS) can model complex social systems and introduce the possibility of agents having common or conflicting goals. These agents may interact with others directly or indirectly. Agents may decide to cooperate for mutual benefit or may compete to serve their own interests [4].

This paper is organized as follows: in Section 2, the General Architecture based on patterns is described; in Section 3 the MAS architecture is described; in Section 4, the Web applications are presented; finally, the conclusions are discussed.

DOI: 10.2316/P.2012.774-074

2. General Architecture based on Patterns

The OOP and Software Design Patterns help to develop Web applications with the goals of flexibility and reusability, keeping the Web application healthy, to regularly update it, and to expand it, when needed. The Software Design Patterns are a set of architectural designs to keep the Web applications dynamic.

Selecting the best design patterns for a particular situation is complex, because Web application development which is our case may have different approaches. Our proposed architecture uses the following Software Design Patterns: Factory Method, Singleton, Composition, Observer, Template Method, and Model-View-Controller (MVC). The key Software Design Patterns are: MVC and Composition, the first on the Server side, and the second on the Client side.

The MVC is a compound pattern, with patterns embedded working together to build complex Web applications, been used in several previous papers [5-8]. The MVC pattern makes the separation of: Model, View and Controller, without overlap in each of their responsibilities. The MVC pattern is shown in Fig. 1, using Software Design Patterns on the Server side: Observer and Singleton, while on the Client side: Composition, Template Method, and Factory Method. The MVC pattern needs to notify all associated Views that a change has occurred, ignoring specific details about the Views, being a recurrent problem, resolved implementing an Observer pattern. The Observer pattern keeps updated Views of the MVC pattern. The Singleton pattern has one and only one instance of the class at any one time, with a global access point, this pattern is implemented with a Java Bean, to manage the Model of the MVC pattern.

The Composition pattern is fundamental to the Client side, allowing us to build composite components based on primitive/composite components, also simplifies the interface, allowing us to manage primitive/composite components in the same way. The Intelligent Reusable Learning Components - Oriented Object (IRLCOO) were implemented with the Adobe Flash platform. Adobe Flash is a multimedia platform with a powerful Object Oriented Programming (OOP) language denominated ActionScript 3.0 [9]. IRLCOO were developed by Peredo et al [5], the components were used to develop: Virtual Laboratory, Problem Based Learning (PBL), Virtual Desktop, VoiceXML, Authoring, and Evaluation Systems. The components load media objects at Run-Time and offer a programmable and adaptive environment to the Learner's needs, using the Composition pattern. The pattern provides a solution to assemble complex systems that are made up of several smaller components.

The system consists of components that may be individual multimedia components or containers with collections of multimedia components. The Composition pattern improves the construction and management of multimedia components composed of related pieces. The IRLCOO can be primitives/indivisible components or composite, containing a collection of other components, a container allowing to the clients to manage primitives/indivisible components and composite components to use a common API, simplifying the interface [10]. This pattern has particular utility in our development, allowing us building and manipulating complex multimedia components. IRLCOO Components have their communications capabilities, grouped in a communication API. The middleware for the components use different frameworks as: Flex [11], AJAX [12], Web Services [13], Hibernate [14], Struts [15], etc.

Fig. 2 shows the class diagram with the composition pattern, allowing us to separate the control/ navigation of the IRLCOO components, maximizing reusability with IRLCOO components based on specialized/indivisible components, also been implemented reusable multimedia components for: animation, sound, image, and video, obtaining a common API for specialized/indivisible components.

Another pattern used on the client side was the Template Method; it is a set of steps for getting something done. The pattern uses inheritance for behavior between classes, allowing to the subclasses to provide with implementation in some operations. The parent class invokes the operations of the subclasses, blocks the order of operations to be called, but not implementations of operations. The IRLCOO components for: Animation, Sound, and Video, sharing actions in common, but with specific differences among multimedia, the Template Method was implemented with the IRLCOO components: IRLCOO_Animation, IRLCOO_Image, IRLCOO_Sound, and IRLCOO_Video. The components focus in the functions of getting a filename and playing, placing them into a set of generic steps, but taking into consideration the differences among multimedia. The next block of code shows the implementation of the pattern:

```
//ActionScript 3.0
package
{
import flash.display.Sprite;
...
//Abstract Class
class Multimedia extends Sprite
{
...
//Template method
public final function mediaPlayer(
):void
{
//Abstract Method
selectMedia();
//Abstract Method
```

```
play();
//Concrete Method
IRLCOO();
}
...
}
}
```

To do changes that were not planned in the design can require changes to existing code. The Factory Method pattern eliminates coupling caused by instantiating concrete classes. It lets to separate the creation of objects from their use. The Factory Method pattern introduces an intermediary between the client and the concrete class that is instantiated, the intermediary is called creator class, the client does not specify the class name, the creator class encapsulates that information, it allows loosely coupled designs.

The Factory Method pattern allowed us a loosely coupled design. The Classes/Objects in the class diagram representing the logic of the Exam, instantiated within the IRLCOO components, in order to evaluate the Exam.

Web applications developed have followed this general architecture, being the following: Authoring, Evaluation, PBL, Virtual Java Programming Laboratory, VoiceXML, Remote Desktop, Virtual Desktop, and Filtered.

3. MAS Architecture

WBE systems are complex, and dynamic. An autonomous agent finds, conveys, or manages information. Because of the nature of the environments, the agents must be long-lived (they should be able to execute unattended for long periods of time), adaptive (they should be able to explore and learn about their environment, including each other), and they should interact and coordinate to achieve their own goals, and the goals of their society; they should rely on other agents to know things so they do not have to know everything [16]. MAS are groups of agents, forming a society in which they play different roles. The group defines the roles, and the roles define the commitments associated with them. When an agent joins a group, he joins in one or more roles, and acquires the commitments of that role. The groups define the social context in which the agents interact [16].

.

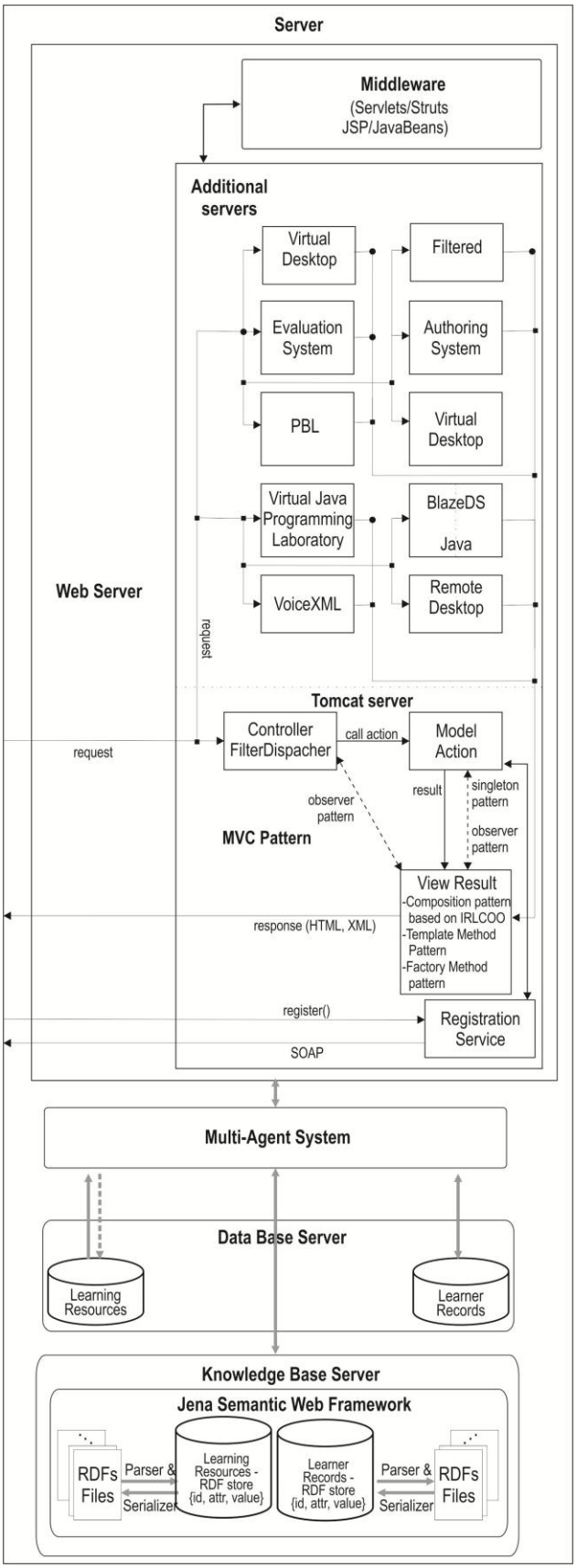

Fig. 1. System architecture of the Server-side.

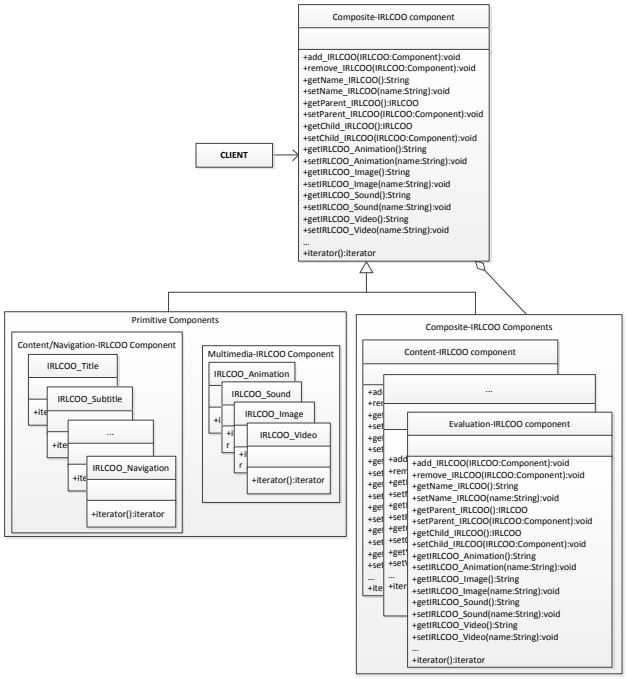

Fig. 2. Class diagram of the Composition pattern.

The Virtual Java Programming Laboratory implemented the first attempt of a MAS showed in Fig. 3. The MAS analyzes the Learner's metrics, dynamically reconfiguring: sequence, level and feedback according to the Learner's needs. The system was implemented using the framework JADE [17]. The package jade.wrapper.gateway is used, and more specifically the classes: JadeGateway and GatewayAgent. The event POST trigger a message from the Learner's Web browser to the system, later the message will be handled by the GateWayServlet, depending on the sendmessage (action type) received. The action creates a BlackBoardBean object, which will be our communication channel between the AgentConnection and the servlet. The AgentDB receives the send message by the AgentConnection and performs the query to the database. The response is obtained and sends it to the AgentConnection, which is in waiting of the mentioned response. The AgentConnection writes the response on the BlackBoardBean and sends a notification to the servlet, which prepares the response. Finally the servlet sends the response to the user's JSP.

The MAS Architecture later was improved based on layer 3 of IEEE 1484 – LTSA architecture specification [5] and implemented the MAS with the frameworks JADEX-Webbridge [18], using the Belief–Desire–Intention (BDI) model. The architecture consists of four processes: Learner, Evaluation, Coach, and Delivery; two stores: Learner Records and Learning Resources; and fourteen information workflows. The MAS makes use of knowledge bases: Learning Resources and Learner Records, Data Bases: Learning Resources and Learner Records, the knowledge bases have Learner's metrics collected via the IRLCOO components e.g., tracking learning, chat, discussion forums, completed activities, etc., the data bases have Learner's metrics collected via the system based on IEEE 1484 – LTSA architecture and made persistent via Hibernate [14], to tailor the: navigation sequence and reconfigure content at Run-Time according to Learner's needs [9-10], modifying the imsmanifest.xml, in order to maintain compatibility with SCORM [11]. The JADEX-Webbridge is to communicate transparently the platform of agents and the Web application [18].

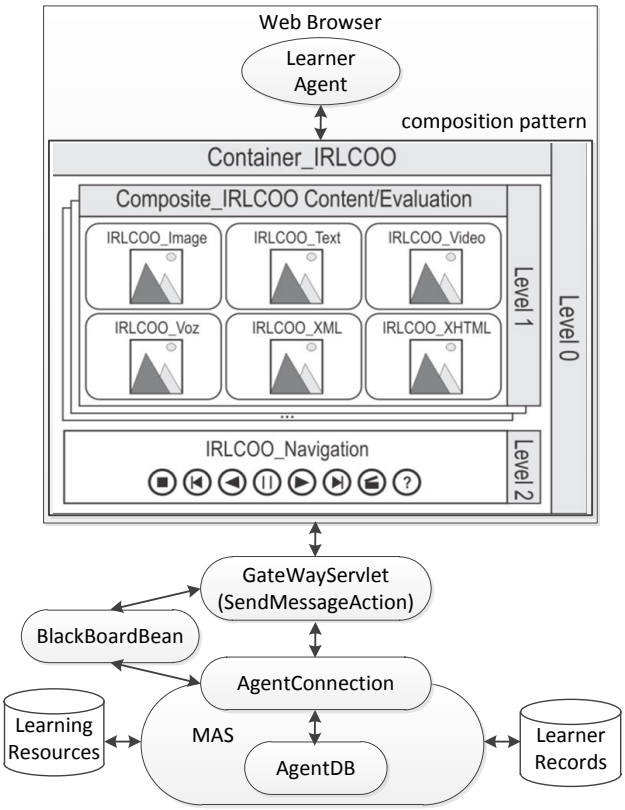

Fig. 3. Communication model between IRLCOO and MAS.

In this case, we centred in: Learner Agent, Coach Agent, Evaluation Agent, and Delivery Agent, which allow the system's adaptive behaviour, providing information about the Learner Agent. The Learner Agent is used by the system for:

-Adaptive support and navigation guide based on prioritized successors and Learner's needs,
-Context support,
-Dynamic support and feedback.

Fig. 4 depicts the MAS architecture of the Server-side of the proposal with the integrated MAS, highlighting the following parts: Middleware, different Web applications developed with patterns, and MAS. Middleware has the following main modules: LMS, Meta-labeled SCORM, Meta-labeled VoiceXML, Meta-labeled XML, Meta-labeled RDF, Uploadfiles, Dynamic sequencing, Dynamic composition, MAS, Dynamic Feedback, etc.

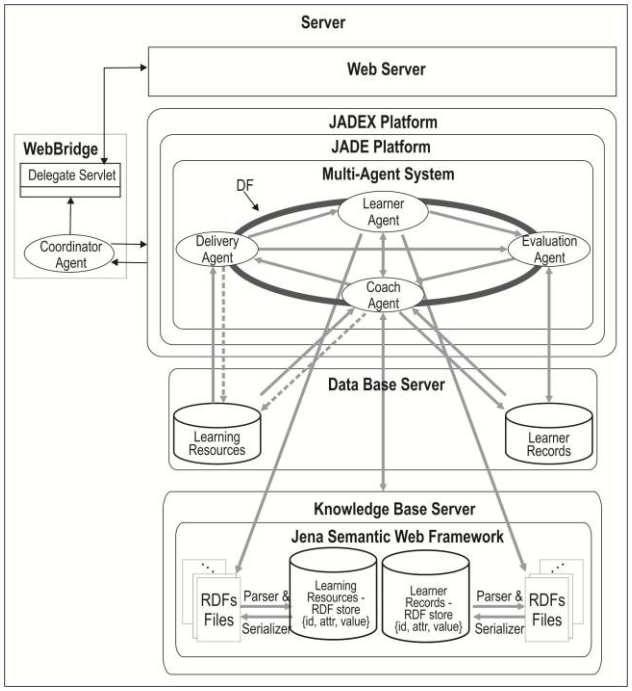

Fig. 4. MAS architecture of the Server-side.

The IRLCOO components take Learner's metrics, these metrics complement the IEEE 1484 – LTSA architecture specification e.g.: tracking learning, tracking evaluation, , evaluation chat, evaluation discussion forums, completed activities, etc., being serialized in RDF format, forming our Knowledge Base layer, it interacts with the Coach Agent and the object Tracker Learner, extracting Learner's metrics personalized via Jena Query Engine (JQE), using: Learning Resources - RDF Store and Learner Records - RDF Store, based on triples, Fig. 4 shows the previously described.

The Learner agent takes information about the Learner, serialized to RDF triples through the IRLCOO components. The Evaluation and Delivery agent collaborate together to reconfigure dynamically the sequence and level of learning materials, through the imsmanifest.xml SCORM, to tailor based on Learner's metrics personalized using the layers: Data Base and Knowledge Base. The Coach agent provides a dynamic and personalized feedback at Run-Time, based on the information taken on Knowledge bases and Data Bases (Learner Records and Learning Resources respectively), during the learning process, and in accordance with the specifications established by the real Coach. The initial states of our agents are established among other things by our: beliefs, goals, and the library of plans. Furthermore to the BDI components, the Directory Facilitator (DF) registers the service descriptions of our gents.

4. Web Applications

Web applications developed are: Authoring, Evaluation, PBL, Virtual Java Programming Laboratory, VoiceXML, Remote Desktop, Virtual Desktop, and Filtered. We will show only some next.

Authoring System facilitate the development of learning content, to facilitate the authoring content to the tutors who are not experts in developing Web applications with multimedia. The Authoring System has a metadata tool that supports the generation of IRLCOO components to provide online courses (see Fig. 5). The courseware takes Learner's metrics based on IRLCOO components, with the purpose to tailor learning experiences. Besides, the IRLCOO components provide a RIA interface, based on IRLCOO components and patterns, compatible with the SCORM Models (Content Aggregation, Sequencing and Navigation, and Run-Time Environment) [3].

The Evaluation System is based on IRLCOO components and patterns. The Evaluation System makes an analysis of the Learner's results, which is built during the Teaching/Learning Process. The results are based on metrics taken during the Learner's behaviour at Run-Time. These measures are stored into the database and knowledgebase named Learner Records under the Learner's results. The reconfiguration of new course/assessment sequences is according to the results obtained, and deposited within the system at: Databases and Knowledgebase, also adapting the level of course materials to Learner's needs, based on IRLCOO components and patterns, taking decisions the Coach Agent in accordance with its established goals.

Fig. 5. Learning content (Spanish) generated for the Authoring System.

MAS utilize the dynamic sequencing to change the course/assessment sequence. The dynamic sequencing is defined for the instructional strategy based on the Sequencing/Navigation SCORM. The Coach Agent invokes Dynamic Reconfiguration Plan, to generate dynamic feedback, according to the Learner's results, comparing the goals set by the Coach Agent, and executed the plans: Dynamic sequencing and Dynamic composition, according to the Learner's metrics.

5. Conclusion

The different Web applications were created based on: IRLCOO components, patterns and MAS. Our approach is based on: accessibility, durability, interoperability, and reusability of the learning contents, built Shareable Content Objects (SCOs) based on IRLCOO components and patterns.

The communication model proposed supports synchronous and asynchronous communication channels, incorporating technologies such as: AJAX, Struts, Hibernate, Web Services, and JADE/JADEX/Webbridge. This model integrated complementary technologies. The different Web applications were developed under this model to help in reducing the complexity to produce learning materials. The IEEE 1484 LTSA standard allows us to produce intelligent and adaptive Web applications with bidirectional communication, according to the Learner's needs at Run-Time, taken certain Learner's metrics.

The Web applications are used to develop educational materials for different courses online at the Institute, in our case applied to the so called POLILIBROS. The benefits of using these tools are: complexity reduction in development, reuse, and updates based on components. The costs are reduced significantly. A recurring factor in the IRLCOO components and patterns is the managing change. One of the most important limitations of the system is the dependency on the plugin Adobe Flash Player.

OOP and Software Design Patterns help to develop Web applications with flexibility and reusability, keeping the Web applications healthy, upgradeable and expandable, maintaining our dynamic Web applications.

6. Future Steps

We must develop more IRLCOO components, is also important to implement more Software Design Patterns in the IRLCOO components. We must further improve the architecture of MAS. We must add more ontologies and improve the use of semantic Web technologies.

Acknowledgements

Authors of this paper would like to thank the Instituto Politécnico Nacional (IPN) and the Escuela Superior de Cómputo (ESCOM) for the support for this work within the project SIP-IPN 20121741. Thanking all students who participated in the development of systems, and particularly to Juan Carlos Caravantes.

References

[1] E. Gamma, R. Helm, R. Johnson & J. Vlissides, *Design patterns: elements of reusable object-oriented software* (Addison-Wesley, 1995).
[2] A Draft Standard for Learning Technology - Learning Technology Systems Architecture (LTSA). Retrieved January 10, 2007 from http://ieee.ltsc.org/wg1.
[3] Advanced Distributed Learning Consortium. Retrieved January 24, 2006 from http://www.adlnet.org.
[4] Fabio L. Bellifemine, Giovanni Caire, and Dominic Greenwood, *Developing Multi-Agent Systems with JADE*. Wiley (England, Series in Agent Technology. Wiley, 2007).
[5] R. Peredo, L. Balladares & L. Sheremetov, Development of intelligent reusable learning objects for Web-based education systems, *Expert Systems with Applications*, 28(2), 2005, 273-283.
[6] A. Canales, A. Peña, R. Peredo, H. Sossa & A. Gutiérrez, Adaptive and intelligent web based education system: Towards an integral architecture and framework, *Expert Systems with Applications*, *33*(4), 2007, 1076-1089.
[7] A. Canales and R. Peredo, *Adaptive and Intelligent Agents Applied in the Taking of Decisions Inside of a Web-Based Education System*, *Intelligent Agents in The Evolution of Web and Applications*. (Studies in Computational Intelligence. Springer Berlin/Heidelberg. Book Chapter, 2009, 87-112).
[8] R. Peredo, A. Canales, A. Menchaca & I. Peredo, Intelligent Web-based education system for adaptive learning, *Expert Systems with Applications*, 38 (2011), 2011, 14690–14702 .
[9] Adobe© Flash©. Retrieved February 26, 2007 from http://www.adobe.com.
[10] W. Sanders, C. Cumaranatunge, *ActionScript 3.0 Design Patterns: Object Oriented Programming Techniques* (Sebastopol, CA: Adobe Developer Library, 2007).
[11] M. E. Davis, J. A. Phillips, *Flex 3: A Beginner's Guide* (USA: McGraw-Hill Osborne Media, 2008).
[12] D. Crane, E. Pascarello, D. James, *Ajax in Action* (Greenwich, CT: Manning Publications, 2005).
[13] E. Newcomer, G. Lomow, *Understanding SOA with Web Services* (Upper Saddle River, NJ: Addison-Wesley Professional, 2004).
[14] N. Heudecker, P. Peak, *Hibernate Quickly* (Greenwich, CT: Manning Publications, 2005).
[15] J. Holmes, *Struts: The Complete Reference* (New York, NY: McGraw-Hill Osborne Media, 2006).
[16] Gerhard Weiss et al., *Multiagent Systems: A Modern Approach to Distributed Artificial Intelligence*. The MIT Press (England, The MIT Press, 2000).
[17] JADE version 3.6. Retrieved May 5, 2008 from: http://jade.tilab.com/dl.php?file=JADE-all-3.6.zip.
[18] JADEX - Overview (About.Overview) - XWiki, Retrieved August 21, 2009 from http://jadex-agents.informatik.uni-hamburg.de/xwiki/bin/view/About/Overview.