E-LEARNING SYSTEM: A SERVICES ORIENTED ARCHITECTURE APPROACH

Marin Lungu, Dan-Ovidiu Andrei, Gabriel Toma-Tumbar, Lucian – Florentin Barbulescu

University of CraiovaFaculty of Automation, Computers and Electronics
Computer and Communications Engineering Department

Abstract: The article presents a new approach to e-learning systems, the architecture based on services, or as it is well known today: Services Oriented Architecture. This architecture best fits an e-Learning system because the prresentation and workflow are constructed from multiple shared services, data and business function are encapsulated in services. Many functions within an e-learning system can be „exported" to services offered by the environment to applications.

Keywords: Services oriented architecture, SOA, e-learning.

1. OVERVIEW

The project's goals are implementing the latest techniques (Services Oriented Architectures) in design, implementation and exploatation of an application inside a distributed environment of resources and users. The services oriented architectures are using object description languages and their interactions having interfaces that are to be used for delivering the services. Each object is offering independent services and the calling mechanisms are platform independent. Such a complex architecture is used for implementing distributed systems; in particular, the e-learning applications are well fit for this context. This projects combines a modern approach for services oriented complex systems with an application in a high priority domain of current society evolution - permanent information and learning by electronic means. The final product will be a framework for creating and validating educational modules, learning the students and testing their knowledge. The means of data transport for the proposed application will be the Internet; this environment's vulnerabilities are well known and there rises the need for security mechanisms for user access to the system resources as well as for the data stream. The final phase of the system will be built from the following 'bricks': a component responsinble for creating and editing educational modules, the educational modules validatidation component, the access component, the testing and automatic evaluation component.

In 2005 the world software market in general and the e-learning systems available in Romania in particular offers products that cover partly the advancements of the new design, implementation and management technologies and of the corresponding equipments (KnowledgePresenter 2005 Professional, MindVision e-learning software, Kallidus, TopClass Publisher, Cognitivity etc.)

The e-learning presents come unique characteristics, for example the use of collaborative instruments like chat or conferencing, navigational options that can put learning into an explanatory mode, and the use of simulation for training of problem-solving skils. (Ruth Colvin Clark, E-Learning and the Science of Instruction : Proven Guidelines for Consumers and Designers of Multimedia Learning, John Wiley & sons, 2003, ISBN: 0-7879-6051-9)

Many e-learning applications are strictly oriented to some specific directions and have pure commercial goals. In this moment it is needed a new application built on modern technologies and which can be easily customized for a variety of learners.

The majority of existing e-learning applications are in fact web sites where the main goals are customize the e-learning process, acce3pt learners in a pay-per-

use mode, implement some security elements etc. The technologies used are in most of the cases old. For example, a php web-site is not a good solution any more mainly because of the security problems it has and of the performance which decreases with the number of users .

Even though we have decades of experience in software production, there are still challenges and unsolved problems in the management of their complexity. As complexity grows, researchers try to find new technologies to solve this problem. SOA combined with web-services represents the newest solution. SOA is the best scalable solution for a complex application. (Sayed Hashimi, Service-Oriented Architecture Explained, O'Reilly Network, 2003).

The new research and implementation directions of complex applications in Internet are based on service oriented architectures, those being available to clients regardless of the platform they use.

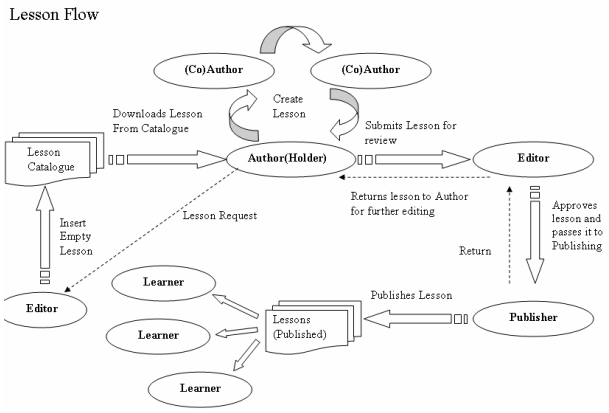

Fig. 1. Lesson Flow

The system is based on the concept of lessons and their versions. A Lesson is in fact a discussion topic. The peoples that work on a lesson will provide several implementations of that specific topic.

There are several types of users found within the e-learning system. Some of those users can be considered “producers” while others can be called “consumers”. In the following all the users found within the system are presented:

1.1 Authors

The author is the creative force behind the lesson system. He can browse the lesson catalog then select and download an empty lesson for further development. He also has the possibility to request the editor for the holder right of a particular lesson. In case there is a lesson he wants to develop that does not appear in the catalogue, the author can submit a request to an editor to have it added to the lesson catalogue. If he desires to collaborate with some of his Author colleagues, he can choose to select those persons by designating them as Co-Authors. He cannot however pass the Holder right to one such Co-Author. An author can upload a lesson to the Collaboration Area – a lesson pool where he (and the Co-Authors) can share (upload and download) intermediate lesson versions. If the Author has Holder right, he can also submit his version for revision to the Editor as Request to Edit Version (REV). Alternatively, he can select one of the versions in the Collaboration Area as the Request to Edit Version and submit it to the Editor. Note: each lesson has its own Collaboration Area for the authors working on it. Once the lesson is submitted to the Editor for review, the Request to Edit Version is locked until an Editor either qualifies the lesson for the next level (Publishing) or returns feedback to Author on how to further improve the content. Lesson sharing is also freely available outside the Collaboration Area – the Author can export the local lesson to one file and send it by e-mail or disk to other Authors.

1.2 Holders

A Holder is an Author with additional Holder rights granted on a lesson basis. As such, a Holder can perform all the Author actions described in the above section. In addition below are described the tasks that are specific to a Holder

1.3 Editors

The Editor is responsible with issuing and revoking authorization for Authors; he grants and revokes the Holder right to an Author for a specific lesson. The Editor also assures the quality control for all new lessons and re-created lessons. He analyzes the requests submitted by authors to add new lessons in the catalog and reviews a lesson submitted by an Author as Request to Edit Version. Upon successful revision of such a lesson, the Editor recommends it for publishing to the Publisher as a Request to Publish Version. Upon negative revision, he can provide feedback to the Authors of a lesson by making suggestions and comments to in order to have the lesson better match the requirements. In completing all these tasks the Editor mainly communicates with Authors, Holders and Publishers.

1.4 Publisher

The Publisher covers the last phase of the lessons generation. He receives the lessons approved by the Editor and performs a final verification on the content. In this he mainly checks for violations of the Law of Author Rights. If such violations are encountered, he informs the Editor and provides him

with feedback on the reasons. In case the lesson is approved for publishing, the lesson is marked as 'Currently Published' and thus available to be chosen in the learning path of students. The Publisher also supervises the updates on the lesson catalogue. He receives from the Editor the lessons approved to be entered in the catalogue and fills the required information for an empty lesson. Also when the Editor designates a lesson as no longer needed in the system, the Publisher marks it as 'Out of Publish' in the catalogue. A lesson with this status can still be downloaded by students that already have it assigned in the learning path. It cannot however be assigned in new learning paths. When the 'Out of Publish' lesson is no longer used by any student, the lesson should receive a new status from the Publisher i.e. 'Archive'.

1.5 Learner

The Learner is the ultimate target of the System. Upon registration within the system a learner can download the Learner Kit; he follows interactive lessons and takes administered exams and tests. He has an assigned Tutor, who creates his Learning Path and monitors his progress. Tools are provided for seamless communication between the Learner and the Tutor. The System extracts and uploads relevant data from Learner's study patterns and exams, allowing the Tutor insightful performance reviews.

1.6 Tutor

The Tutor is the direct supervisor of a Learner. He creates (or selects from a system list) the Learner's Learning Path, evaluates his performance and provides the Learner with feedback on the results and progress. The Tutor uses tools to freely communicate with his Learners and their Parents.

1.7 Parents

The Parent can view Learner's progress and performance. He also has the ability to communicate with the Learner's Tutor using ' tools.

1.8 User Manager

The UserManager is an administrative role that receives and evaluates the online requests for authorization from Author, Editor, Publisher, Learner, Tutor and Parent users. He can create, edit and delete user accounts for these roles. Furthermore, the UserManager is in charge with assigning Learners to Tutors. If for some reason the Tutor must be changed for a Learner, the UserManager has the ability to make such a change.

1.9 Administrator

The Administrator is able to by-pass the normal workflow so that in extraordinary circumstances the system does not stop from working properly. He can perform all the tasks assigned to the UserManager, with the addition that he is also permitted to create UserManager and Administrator accounts. Furthermore he can send notifications to users of all types and force-change the rights of an Author on a lesson (such as the Holder right). He is allowed to by-pass the normal Lesson flow by moving a lesson from any one area to another (ex. from Collaborative Area to Publishing).The Administrator also has access to the system logs, detailing the time-stamped actions of all users.

1.10 Automatic

The Automatic system of has the ability to send notifications on predetermined triggers (such as inactive users or special status changes for lessons). It also has the responsibility to log and timestamp user actions

1. SERVICES ORIENTED ARCHITECTURES (SOA)

Definition: SOA is an architectural style whose goal is to achieve loose coupling among interacting software agents.

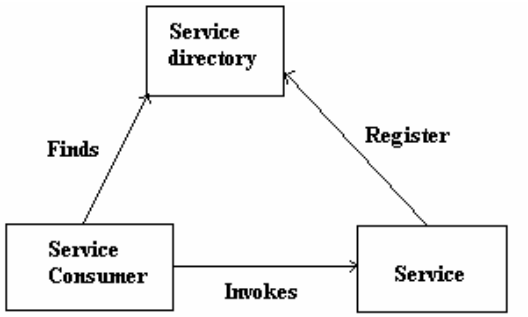

Figure 1 - Services Oriented Architecture

The participants in the Services Oriented Architecture are:

- Service (service provider)
- Service directory
- Service consumer

Service (Service Provider): The service provider is responsible for publishing a description of the service to the service registry. Normally, the service provider hosts the web service.

Service Directory: The service registry is a repository that provides the capability of discovering services by the service requestors.

Service Consumer: A software application that is responsible for discovering and invoking the service. The service requestor binds to the service obtained from the service registry.

A service is a unit of work done by a service provider to achieve desired end results for a service consumer. Both provider and consumer are roles played by software agents on behalf of their owners.

\

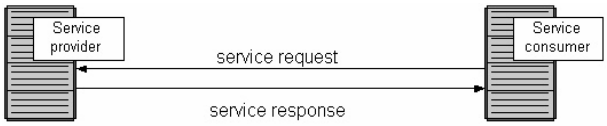

Figure 2 – Service Request/Response

SOA achieves loose coupling among interacting software agents so by employing two architectural constraints:

- A small set of simple and ubiquitous interfaces to all participating software agents. Only generic semantics are encoded at the interfaces. The interfaces should be universally available for all providers and consumers.
- Descriptive messages constrained by an extensible schema delivered through the interfaces. No, or only minimal, system behavior is prescribed by messages. A schema limits the vocabulary and structure of messages. An extensible schema allows new versions of services to be introduced without breaking existing services.

Conceptually, there are three major levels of abstraction within SOA:

- Operations: Transactions that represent single logical units of work (LUWs). Execution of an operation will typically cause one or more persistent data records to be read, written, or modified. SOA operations are directly comparable to object-oriented (OO) methods. They have a specific, structured interface, and return structured responses. Just as for methods, the execution of a specific operation might involve invocation of additional operations.
- Services: Represent logical groupings of operations. For example, if we view Customer Profiling as a service, then, Lookup customer by telephone number, List customers by name and postal code, and Save data for new customer represent the associated operations.
- Business Processes: A long running set of actions or activities performed with specific business goals in mind. Business processes typically encompass multiple service invocations. Examples of business processes are: Initiate New Employee, Sell Products or Services, and Fulfill Order.

In SOA terms, a business process consists of a series of operations which are executed in an ordered sequence according to a set of business rules. The sequencing, selection, and execution of operations are termed service or process choreography. Typically, choreographed services are invoked in order to respond to business events.

Since we have only a few generic interfaces available, we must express application-specific semantics in messages. We can send any kind of message over our interfaces, but there are a few rules to follow before we can say that an architecture is service oriented.

First, the messages must be descriptive, rather than instructive, because the service provider is responsible for solving the problem. This is like going to a restaurant: you tell your waiter what you would like to order and your preferences but you don't tell their cook how to cook your dish step by step.

Second, service providers will be unable to understand your request if your messages are not written in a format, structure, and vocabulary that is understood by all parties. Limiting the vocabulary and structure of messages is a necessity for any efficient communication. The more restricted a message is, the easier it is to understand the message, although it comes at the expense of reduced extensibility.

Third, extensibility is vitally important. It is not difficult to understand why. The world is an ever-changing place and so is any environment in which a software system lives. Those changes demand corresponding changes in the software system, service consumers, providers, and the messages they exchange. If messages are not extensible, consumers and providers will be locked into one particular version of a service. Despite the importance of extensibility, it has been traditionally overlooked. At best, it was regarded simply as a good practice rather than something fundamental. Restriction and extensibility are deeply entwined. You need both, and increasing one comes at the expense of reducing the other. The trick is to have a right balance.

Fourth, an SOA must have a mechanism that enables a consumer to discover a service provider under the context of a service sought by the consumer. The mechanism can be really flexible, and it does not have to be a centralized registry.

Additional constraints

There are a number of additional constraints one can apply on SOA in order to improve its scalability, performance and, reliability.

2.1 Stateless Service

Each message that a consumer sends to a provider must contain all necessary information for the provider to process it. This constraint makes a service provider more scalable because the provider does not have to store state information between requests. This is effectively "service in mass production" since each request can be treated as generic. It is also claimed that this constraint improves visibility because any monitoring software can inspect one single request and figure out its intention. There are no intermediate states to worry about, so recovery from partial failure is also relatively easy. This makes a service more reliable.

2.2 Stateful Service

Stateful service is difficult to avoid in a number of situations. One situation is to establish a session between a consumer and a provider. A session is typically established for efficiency reasons. For example, sending a security certificate with each request is a serious burden for both any consumer and provider. It is much quicker to replace the certificate with a token shared just between the consumer and provider. Another situation is to provide customized service.

Stateful services require both the consumer and the provider to share the same consumer-specific context, which is either included in or referenced by messages exchanged between the provider and the consumer. The drawback of this constraint is that it may reduce the overall scalability of the service provider because it may need to remember the shared context for each consumer. It also increases the coupling between a service provider and a consumer and makes switching service providers more difficult.

2.3 Idempotent Request

Duplicate requests received by a software agent have the same effects as a unique request. This constraint allows providers and consumers to improve the overall service reliability by simply repeating the request if faults are encountered.

REFERENCES

Clark R.C. (2003), *E-Learning and the Science of Instruction : Proven Guidelines for Consumers and Designers of Multimedia Learning,* John Wiley & sons, ISBN: 0-7879-6051-9

Hashimi S (2003), *Service-Oriented Architecture Explained*, O'Reilly Network

OASIS, Reference Model for Service Oriented Architectures, Working Draft 08, September 14, 2005, http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=soa-rm

Rogers R. , *Reuse engineering for SOA*, Senior Technical Staff Member, IBM

Rosenberg M.J. (2001), *E-Learning: Strategies for Delivering Knowledge in the Digital Age*, McGraw-Hill, ISBN: 0-07-136268-1

W3C Working Group Note, Web Services Architecture, http://www.w3.org/TR/ws-arch/ , 11 February 2004