A Supporting Architecture for Generic Service Integration in IMS Learning Design

Luis de la Fuente Valentin*, Yongwu Miao, Abelardo Pardo, and Carlos Delgado Kloos

Department of Telematics Engineering, Carlos III University of Madrid, Spain
Educational Technology Expertise Center, Open University of The Netherlands
{lfuente,abel,cdk}@it.uc3m.es
yongwo.miao@ou.nl
http://gradient.it.uc3m.es

Abstract. Learning Design offers the possibility of capturing the process, activities, user organization and resources used in a learning experience. But a wider set of scenarios appear when generic services are considered. Integrating such services in a Unit of Learning is difficult due to the lack of a defined bi-directional protocol for information exchange. In this paper the Generic Service Integration paradigm is presented. It extends the Learning Design specification to use generic services, first at the design stage of a Unit of Learning, and then at the deployment and run times. The framework allows for bi-directional exchange of information between a Unit of Learning and a service. The consequences of the approach are that services can be configured to suit the needs of activities in a learning environment, and a Unit of Learning may adapt its behavior based on the events that took place in any of the used services.

Key words: service, integration, learning design, IMS LD, learning, course

1 Introduction

In the evolution of e-learning technical standards, the release of the IMS Learning Design specification [1] (henceforth IMS LD, or simply LD) supposed a shift of focus from supporting content-centric learning to supporting activity-centric learning. Using IMS LD, multiple approaches to learning such as empirical, rationalist, pragmatic, cultural, historic, etc. can be formally modeled as a unit of learning (UoL) [2]. Once defined, a UoL can be instantiated and automatically executed at a run-time environment for scaffolding students to conduct online learning with the help of staff and other learners in a virtual learning context. A UoL prescribes how participants with various roles should individually or collaboratively perform activities in sequence or/and in parallel towards learning objectives within associated learning environments, where necessary learning objects and learning services are available [1].

In certain learning activities, especially those present in rationalist, pragmatic and cultural-historic approaches, learners interact with each other and

* Corresponding author

with learning objects through the use of a variety of learning services. Without these services, activities cannot be properly supported by the environment and the number of possible pedagogical strategies is severely reduced.

The use of learning services is available through the use of the LD specification. Only four types of services are included: send mail, monitor, index search, and conference. To support a wider range of activities, the specification should allow the inclusion of more services. A generic approach that fits with the current specification is required to allow any service to be integrated in a UoL. This is the goal of the architecture presented in this document.

The architecture described in this document aims at minimally extending the current Learning Design specification such that UoLs may instantiate generic services by describing the required functionality. Furthermore, a communication protocol is presented to allow a bi-directional communication between a LD run-time environment and a remote service. Thus, services can be tailored to the specific needs of a learning environment, and the environment can adapt itself depending on the information reported by the service.

The paper is organized as follows. Section 2 presents the main initiatives that have considered the problem of service integration in a learning context. Section 3 includes a formal definition of the problem of generic service integration. A software prototype for testing purposes is outlined in Section 4. Finally, Section 5 is devoted to conclusions and future work.

2 Background

Interaction among services is a research topic that applies to numerous scenarios. In the context of a learning experience, the IMS Tools Interoperability Specification [3] focuses on facilitating integration of third party tools with learning management platforms. The concept of Personal Learning Environment (PLE) [4] also considers the idea of service orchestration in a learning environment.

None of these initiatives are implicitly related to IMS LD. Interaction with generic services will allow additional pedagogical models to be expressed with LD, increasing the current scope of the specification. This section analyzes how other initiatives explore the problem.

CopperCore is a learning design engine that allows its output to be formatted and presented to the user. The CopperCore Service Integration Layer (or simply CCSI) is an additional functionality conceived to be used in conjunction with the LD engine. This layer allows new services to be added and extend the Learning Design Framework. Services are added through Interoperable Segments (APIS) [5], whose adapter allow synchronization between services.

Using this approach, QTI assessments have been integrated within an UoL. Synchronization between QTI outcomes and LD properties is specified at IMS Interoperability Guidelines [3]. In a similar way, SCORM functionality [6] and Adaptive Game Services [7] have been integrated in a LD defined course.

Although CCSI provides the necessary functionality, integrating a new service requires deep knowledge of the specific API. The framework also offers the possibility of writing special purpose functions to interact between the LD Engine and services. As it has been shown with concrete services, the effort required

to perform integration using CCSI approach suggests that it is not a solution suitable for agile integration of a large number of services into IMS LD.

An alternative approach to service integration in LD has been explored within the framework of the TENCompetence project using Widgets as shared services. Widgets provide a very attractive and interactive user interface that could improve engagement with Learning Design-based systems and they offer an interesting new approach on adding interactive features to learning designs [8].

Widgets conception is focused on supporting architecture at runtime. Information that allows the inclusion of a widget in a LD course does not deal with concepts such as roles, permissions, multiplicity, life-span, etc. which are required to express all service behavior details and integrate them in the UoL.

3 Generic Service Integration

Generic Service Integration is proposed as a specification that complements IMS LD by providing a framework to design and deploy generic services and their inclusion on LD defined courses. UoLs are therefore created by teaching staff with experience not in technology but in pedagogy. In this context, course authors should use GSI only to specify the services that need to be included in a learning experience, and leave the details on how the service is instantiated and deployed to the run-time environment. The proposal is divided in **design** and **deploy**. This section details both elements of the proposal.

Design Time

There is no restriction in the type of services that can be used in the context of a UoL. Depending on the area, tools for simulation, benchmarking, communication, search and many other features are used to improve the learning process. Each of these services needs different settings to be configured. A data model including settings parameters from every type of service is too large to be managed, and too complex to be used in practice. Therefore, the GSI approach proposes the use of common attributes from all services and defines a model valid for any type of service. The required attributes are provided by the instructional designers to define the type of services that will be used in a learning activity. At deploy time, the LD player will use this information to search and instantiate a tool that complies with the given requirements.

The proposed data model, depicted in Figure 1, can be expressed in XML. This information binding is placed inside the *service* element on the LD manifest. Data model is structured as follows: **Group** element references to LD *roles* allowing to set different user rights; **Tool** section specifies service expected behavior and a set of defining keywords; **Constraints** element sets extra requirements such as time limit or service multiplicity; finally, **Alternatives** are used to specify a secondary service used when main one cannot be properly deployed.

In a typical use case, the *group* element consist of references to the student and teacher LD roles. Tool description contains the functions to be called, usually *deploy* (at the beginning) and *close* (i.e. when time expressed by a constraint expires). Permissions will allow students to write contributions, and teachers to administrate the contributions of all participants. By setting multiplicity to *one-per-role*, each group of students will have their own service instance.

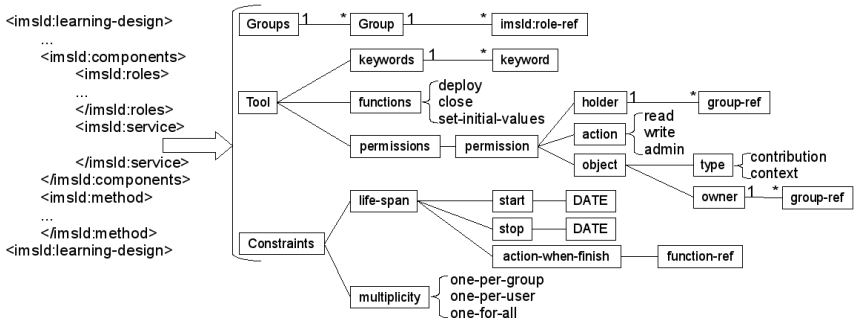

Fig. 1. The service element in the IMS manifest include the GSI information.

Deployment Time

UoLs are imported to a LD runtime environment, where the course can be instantiated several times (that is, several *runs*) with the participation of different users. The UoL defines how the course must behave and react to user's interactions. The course author is unaware of who will take part in the course and what runtime environment will be used.

Services require to be instantiated once per run. At design time, the author define that a learning activity will be supported by a tool (a service), but cannot ensure this tool to be available on the deployment platform, he can only introduces limits on service behavior. These limits are the information compiled in GSI; the deploy manager instantiates and configures a service based on service description. Shown in Figure 2, deployment steps can be summarized as follows.

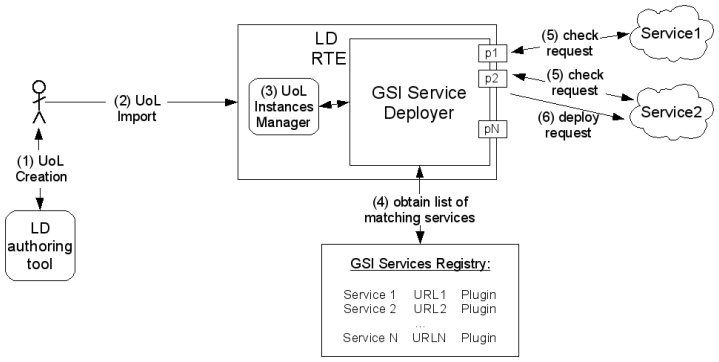

Fig. 2. GSI Supporting Architecture Diagram. The course life-cycle includes course creation and instantiation of services during UoL instantiation.

First, the RTE must find a service that matches the requisites. A **keyword based lookup** (step 4) is performed on a registry, where GSI compliant services have been previously recorded. The retrieved data must include where to find the

service (URL) and how to exchange information with it (plugin to use). Searching criteria is not enough expressive, so next step is negotiation with external services. Using the proper plugin to establish communication, a *check-request* is sent to all found services (step 5). The response contains the requirements that are supported by the service. Then, the LD runtime environment chooses the service that meets more appropriately the requirements, even if not all of them are available. Finally, a *deploy-request* is sent to the chosen service. The answer to such request must include - depending on multiplicity - a list of one or more URL where the requested instances are available (step 6). With these URLs, the LD player inserts a link to the service in the environment of the UoL.

In most cases, user registration in services is simply not possible to be automated because the procedure includes a challenge-response test to ensure that registration is not being done by a computer. Shared identity initiatives such as OpenID[9] could offer the required flexibility to bypass this problem.

4 Software prototype description

GSI has been tested in a software prototype built, in GRAIL [10], developed to act as proof of concept of the specification. This runtime environment is fully integrated within the .LRN LMS [11]. The modularity of the OpenACS architecture [12] - the underlying technology of .LRN - facilitates the inclusion of new functionalities such as the one proposed in this document.

A plugin based architecture, where a simple API for the plugin layer is defined, allows service-independence. New services can be included without changing implementation of existing ones. Two functions must be implemented to build a new plugin: *check-request* to provide the GSI negotiation with services, and *deploy-request* to ask the service for a new instance.

In the implemented prototype, the service chosen to interact with is a wiki editor. The plugin has been built in a *simplified* way: the service only receives calls for functionality that already exists on the service API. Based on the answers obtained from the service, the plugin simulates an information exchange that fits the specified behavior in GSI. Thus, the service is compatible with GSI with almost no required modification. A different development approach can include *active plugins* which exchange information in a two-way communication, requiring the inclusion of GSI functionality on the service software.

The prototype requires the administrator to manually select the service for a set of options returned as a result of the registry lookup. However, this selection can be easily automated thus leaving the process with no human intervention. Future versions of the prototype are expected to support customization of the amount of automated tasks.

5 Conclusions

This paper presents *Generic Service Integration* as the way to integrate the use of services into Learning Design courses. GSI recognizes the relevancy of UoL authoring in the course life-cycle and provides authors with the capability of capturing service behavior in a packaged UoL. Service definition given by authors

is later used to find a proper tool that matches the expressed requirements. The negotiation of available functionality is then carried out with the available tools, which may be placed in a remote system. Instantiation of the service and URL retrieval is the last phase of deployment.

A plugin based implementation allows communication with different services. Using the public API of any tool, a one-way strategy can be performed in case the service cannot be modified. Otherwise, information exchange between service and LD player may result in a more powerful configuration of the service. In any case, the Learning Design runtime environment perceives the process as a request-response communication where the selected plugin hides the complexity.

A first prototype has been implemented and tested. The requested service was an instance of a wiki for each instance of a given role. GSI offered the proper functionality to be able to request such service at design time and perform the initialization of the different instances with almost no human intervention.

Acknowledgment

Work partially funded by *Programa Nacional de Tecnologías de la Información y de las Comunicaciones*, project TSI2005-08225-C07-01/02

References

1. "IMS Learning Design specification," http://www.imsglobal.org/learningdesign/, Feb. 2003, [On line].
2. R. Koper, "Modelling units of study from a pedagogical perspective: The pedagogical meta-model behind eml," 2001, document prepared for the IMS Learning Design Working Group. Heerlen: Open Universiteit Nederland.
3. "IMS Tools Interoperability," http://www.imsglobal.org/ti/, Feb. 2006, [On line].
4. S. Wilson, O. Liber, M. Johnson, P. Beauvoir, P. Sharples, and C. Milligan, "Personal Learning Environments: Challenging the dominant design of educational systems," in *Joint International Workshop on Professional Learning, Competence Development and Knowledge Management (LOKMOL and L3NCD)*, October 2006.
5. "Assessment provision through interoperable segments," http://www.jisc.ac.uk/whatwedo/projects/apis.aspx, May 2007, [On line].
6. P. Sharples, D. Griffiths, and C. Tattersall, "Integrating IMS Learning Design and ADL SCORM using CopperCore Service Integration," in *Proc. of the 2nd TENCompetence Open Workshop*, 2007.
7. P. Moreno-Ger, D. Burgos, J. L. Sierra, and B. Fernández-Manjón, "A Game-Based Adaptive Unit of Learning with IMS Learning Design and <e-adventure>." in *EC-TEL*, ser. Lecture Notes in Computer Science. Springer, 2007, pp. 247–261.
8. S. Wilson, P. Sharples, and D. Griffiths, "Extending IMS Learning Design services using Widgets: Initial findings and proposed architecture," in *Proc. of the 3rd TENCompetence Open Workshop on Current Research on IMS Learning Design and Lifelong Competence Development Infrastructures*, 2007.
9. "OpenID specification," 2006. [Online]. Available: http://openid.net/specs.bml
10. J. E. del Cid, L. de la Fuente Valentín, S. Gutiérrez, A. Pardo, and C. D. Kloos, "Implementation of a Learning Design Run-Time Environment for the .LRN Learning Management System," Journal of Interactive Media in Education, 2007.
11. "The .LRN platform," http://dotlrn.org, Oct. 2007, [On line].
12. "Open Architecture Community System," http://openacs.org, Oct. 2007, [On line].