Modelling a network of heterogeneous eLearning Systems

José Paulo Leal[1] and Ricardo Queirós[2]

[1]CRACS/INESC-Porto & DCC/FCUP, University of Porto, Portugal
zp@dcc.fc.up.pt

[2]CRACS/INESC-Porto & DI/ESEIG/IPP, Porto, Portugal
ricardo.queiros@eu.ipp.pt

Abstract. In recent years emerged several initiatives promoted by educational organizations to adapt Service Oriented Architectures (SOA) to eLearning. These initiatives commonly named eLearning Frameworks share a common goal: to create flexible learning environments by integrating heterogeneous systems already available in many educational institutions. However, these frameworks were designed for integration of systems participating in business-like processes rather than on complex pedagogical processes as those related to automatic evaluation. Consequently, their knowledge bases lack some fundamental components that are needed to model pedagogical processes. The objective of the research described in this paper is to study the applicability of eLearning frameworks for modelling a network of heterogeneous eLearning systems, using the automatic evaluation of programming exercises as a case study. The paper surveys the existing eLearning frameworks to justify the selection of the e-Framework. This framework is described in detail and identified the necessary components missing from its knowledge base, more precisely, a service genre, expression and usage model for an evaluation service. The extensibility of the framework is tested with the definition of this service. A concrete model for evaluation of programming exercises is presented as a validation of the proposed approach.

Keywords: SOA, interoperability, eLearning.

1 Introduction

The architecture of eLearning platforms is moving away from centralised systems towards decentralised networks of heterogeneous systems. The types of systems participating in these networks range from existing eLearning systems to supporting services for specialized tasks. These systems and services may participate in several learning processes that can be easily reconfigured to meet changing requirements and demands. We are particularly interested in networks of eLearning systems providing services related to the automatic evaluation of programming exercises. Networks of

this kind include systems such as Learning Management Systems (LMS), Evaluation Engines, Learning Objects Repositories and Exercise Resolution Environments. These types of systems have a completely different nature. Some expose their functions as web services, such as Learning Objects Repositories or the Evaluation Engines. Others have their own web interfaces for students and teachers, such as Learning Management Systems. Some, as is the case with Integrated Development Environment (IDE) that we intend to use as exercise resolution environments, were not even designed to interact in the eLearning realm and must be extended for that purpose. Modelling a network with such heterogeneity is challenging.

There are a number of eLearning Frameworks to adapt Service Oriented Architectures (SOA) to eLearning promoted by educational organizations. These frameworks share a common goal: to create flexible learning environments by integrating heterogeneous systems already available in many educational institutions. However, these frameworks were designed for integration of systems participating in business-like processes rather than of complex pedagogical processes as those related to automatic evaluation. Consequently, their knowledge bases lack some fundamental components that are needed to model pedagogical processes.

The objective of the research described in this paper is to study the applicability of eLearning frameworks for modelling a network of heterogeneous eLearning systems, using the automatic evaluation of programming exercises as a case study. The paper surveys the existing eLearning frameworks to justify the selection of the E-Framework. This framework is described in detail and identified the necessary components missing from its knowledge base. The extensibility of the framework is tested with the definition of the missing components. A concrete model for evaluation of programming exercises is presented as a validation of the proposed approach. The proposed model reflects the experience gained by the authors with Mooshak and EduJudge projects. Mooshak [2] is a contest management system for ICPC contests that is being used since 2002 also as an e-Learning tool in computer programming courses. EduJudge [3] is a system developed for enabling the use by Learning Management Systems (LMS) of the collection of programming exercises of the UVA on-line judge[1]. Both systems have automatic evaluation components that if recast as services could provide their functions to different types of e-Learning systems.

The remainder of this paper is organized as follows: section 2 presents a survey of the existing eLearning frameworks. Section 3 introduces the e-Framework and its technical model. In the following section we detail our contribution to the E-Framework's knowledge base with the creation of a service genre, expression and usage model that comprises the definition of an evaluation service. Then, we evaluate the usefulness of this evaluation service by integrating it in a network of systems that aims the automatic evaluation of programming exercises. Finally, the paper summarizes the current trends in eLearning frameworks development and open challenges for research.

1 Official Web Site, http://uva.onlinejudge.org/

2 eLearning frameworks

An eLearning framework can be defined as a specialized software framework. In the eLearning field, this term has been associated with several initiatives to adapt SOA to eLearning. Based on Service Oriented Approaches [8], the process of moving from a framework to a working implementation can be defined by four key concepts: **broad vocabulary** describes all possible 'services' for a domain such as eLearning; **reference model** combines these services for specific learning or teaching requirement; **design** specifies the use of standards and specifications for these combinations; and **artifact** is an implementation (software, process, workflow) of a design.

A Framework provides a vocabulary of Services (e.g. digital repositories services), from which a Reference Model (e.g. describing content management) is derived. A particular Design (e.g. repository management application) is modelled based on the Reference Model which is then implemented as an Artifact.

Based on these key concepts, we group them in abstract and concrete frameworks. While **abstract frameworks** provide a broad vocabulary and a reference model for the development of eLearning systems, **concrete frameworks** provide also designs and/or artifacts. In the remainder of this section, we categorize eLearning frameworks based on these groups.

Abstract frameworks aim only at the creation of specifications, recommendations and best practices for the development of eLearning systems. We highlight three initiatives belonging to this category, more precisely, the IMS Abstract Framework [12,13], the Open Knowledge Initiative [10,11] and the IEEE Learning Technology Systems Architecture [9, 10], in the chronological order of their first definition.

Concrete frameworks extend the goals of abstract frameworks by providing a complete service designs and/or components that can be integrated in actual implementations of artifacts. We highlight four initiatives: E-Framework [17,18], Schools Interoperability Framework [15,16] and Open University Support System [14]. Based on the previous grouping we made a survey to categorize and to compare the features of these frameworks. The following figure traces their evolution.

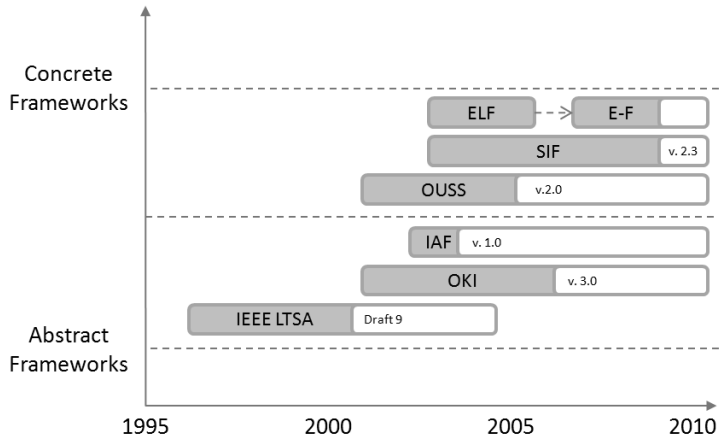

Figure 1 – Evolution of eLearning Frameworks.

The previous figure suggests that, in the last decade, the trend is the appearance of the concrete frameworks rather than abstract frameworks. It is worth noting that none of the concrete frameworks we mentioned actually implement artifacts. At most, these projects include user contributed components, which can be integrated in artifacts for systems using the framework, but are not part of the framework itself.

We also compare five of these frameworks regarding: impact and maturity, architectural models, adopted standards and user groups.

Table 1. eLearning Frameworks survey.

Facets	Features	LTSA	OKI	IAF	SIF	E-F
Impact and Maturity	Creation date	1996	2001	2003	2003	2007
	1st version date	1996	2003	2003	2003	-
	Last vers. Date	2001	2006	2003	2009	-
	Cited projects	-	3	-	37	4
	Contributions	inactive	yes	yes	Yes	yes
Architectural Models	Main model	layered	layered	layered	flat	layered
	SOA	yes	yes	yes	Yes	yes
Adopted Standards	Content format	-	-	-	SCORM	SCORM IMS CP
	Metadata	LOM	LOM	LOM	LOM	LOM,DC
	Web Service	SOAP	SOAP	SOAP, REST	SOAP, REST	SOAP, REST
	Lang. bindings	-	JAVA, PHP, C#	JAVA	JAVA	JAVA
User Groups	Framework users	ESV	ESV	IMS	ESV	ESV
	End users	HE	HE	HE	K-12	HE

From the previous table we can conclude that some frameworks have a very low update frequency (IAF) for several initiatives and one of them is already inactive (LTSA). The frameworks with the most recent updates are the E-Framework and SIF. In the case of the E-Framework, it has been receiving great amount of input from the eLearning community. On the other hand, SIF is the most widely used framework with 37 cited projects in the project web site.

We also conclude that all frameworks adhere to a service-oriented approach. Most of them use the layered architectural model. In this model components communicate only with components in the neighbouring layers. In particular, the LTSA has five layers in its architecture, but only one layer (system components) is normative. In the flat model there is no restriction to the communication among components. The SIF framework is a special case in applying this model since it uses a central component (ZIS) that orchestrate all the communication between applications. These frameworks use different main concepts to present their inner structure. OKI and IAF are an exception since they share their main concepts, which is probably due to the fact that these projects are cooperating [13].

In terms of standards we conclude that certain standards are common to almost all frameworks. For instance, LOM for metadata content, WSDL for service description, SOAP for web service and Java for language binding are common to all frameworks.
Finally, we notice that Educational Software Vendors (ESV) are the most common framework users, with the exception of IAF. IMS uses the framework to develop

internal specifications (e.g. IMS Enterprise Services Specification). Regarding eLearning systems end users, the Higher Education (HE) sector is the most targeted.

Based on this survey, we conclude that e-Framework and Schools Interoperability Framework (SIF) to be the most promising e-learning frameworks since they are the most active projects, both with a large number of implementations worldwide. In the e-Framework we can contribute by proposing new service genres, service expressions and service usage models. On SIF we cannot make this type of contribution to the abstract framework. However, we can contribute with new agents, such as learning objects repositories.

3 The e-Framework

The e-Framework is arguably the most prominent e-learning framework currently in use. For this reason it was selected as basis for modelling a programming exercises evaluation service.

The e-Framework is an e-learning framework aiming to facilitate technical interoperability within and across higher education and research through improved strategic planning and implementation processes. The e-Framework is an initiative that was initially established by the UK's Joint Information Systems Committee (JISC) and Australia's Department of Education, Employment and Workplace Relations (DEEWR). In 2007, the two founding partners were joined by the New Zealand Ministry of Education (NZ MoE) and The Netherlands SURF Foundation (SURF).

The e-Framework has a knowledge base to support its technical model. The technical model of the e-Framework aims to facilitate system interoperability via a service-oriented approach [11]. The model provides the following set of technical components enumerated in Table 2.

Table 2. Technical Model.

Components	Description	User role
Service Genre	A collection of related behaviours that describe an abstract capability.	No technical expert (e.g. IT Manager)
Service Expression	A specific way to realise a service genre with particular interfaces and standards.	Technical expert (e.g. Developer)
Service Usage Model	The relationships among technical components (services) used for software applications.	Domain expert (e.g. Business Analyst)

A **service genre** describes a generic or abstract service expressed in terms of behaviours (e.g. authenticate, harvest, search). A service genre specifies what a service should do without specifying how it should work. This type of component is usually described by IT Managers without any technical knowledge.

A **service expression** is a realisation of a single service genre by specification of exact interfaces and standards used. Since this component covers various technical aspects is more suitable for programmers.

A **service usage model (SUM)** describes a model of the needs, requirements, workflows, management policies and processes within a domain. Hence, the expected candidates to formally describe SUMs are those with the domains' knowledge. A SUM is composed of either service genres or service expressions, but not a mixture.

Other components such as specifications and standards (e.g. IMS Metadata, LOM) are used by service expressions but are not also defined by the e-Framework.

4 Evaluation service description

In this section we contribute to the E-Framework with the definition of an evaluation service. The purpose of this type of service is to be used by an Evaluator to mark and grade exercises in computer programming courses and in programming contests. By exposing its functions as services a programming exercise evaluator is able to participate in business processes integrating different system types, such as Programming Contest Management Systems, Learning Management Systems, Integrated Development Environments and Learning Object Repositories.

The contribution is composed by a service genre, a service expression and a service usage model.

4.1 Text File Evaluation Service Genre

In the e-Framework a *service genre* describes generic capabilities of a specific service expressed in terms of their behaviours, without prescribing how to make them operational.

In this section a text file evaluation service genre is proposed to the E-Framework. A service of this genre is responsible for the assessment of a text file with an attempt to solve an exercise described by a LO. It supports three functions:

- `ListCapabilities:` provides the requester with a list of all the capabilities supported by a specific evaluator;
- `EvaluateSubmission:` performs the evaluation of a submission to a given exercise, using some of the available capabilities;
- `GetReport:` accesses a detailed report of a previous evaluation.

In the following sub-subsection the three service internal functions are detailed.

The **ListCapabilities function** informs the client systems of the capabilities of a particular evaluator. Capabilities depend strongly on the evaluation domain. For instance, in a computer programming evaluator the capabilities are related with the programming language compiler or interpreter. Each capability has a number of features to describe it and for a programming language they may be the language name (e.g. Java) its version (e.g. 1.5) and vendor (e.g. JDK). On an electronic circuit simulator a capability may be a collection of gates that are allowed on a circuit and features may be the names of individual gates.

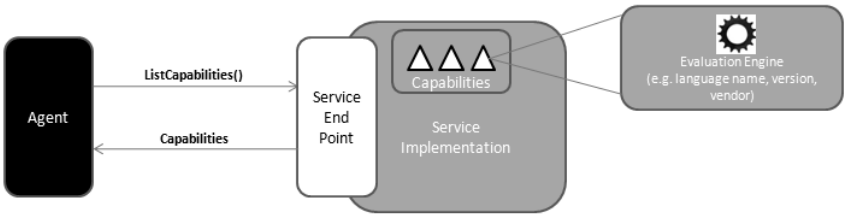

Figure 2 – The ListCapabilities function.

In this function, represented in Figure 2, the request doesn't accept any parameter and the response returns a list of all capabilities of the evaluator. Each capability is described by a list of features, with a name and a value.

The **EvaluateSubmission** function allows the request of an evaluation for a specific exercise. The request includes an exercise or a reference to an exercise represented as a learning object held in a repository and a single attempt to solve a particular exercise. The request may include a specific evaluator capability necessary for a proper evaluation of the attempt. The response returns a ticket for a later report request and may return also a circumstantial report about the respective evaluation of the requester attempt.

A schematic of this function is shown in Figure 3. The service endpoint provides the interfaces for the requests and responses for the evaluation functionality. Internally the service implementation may include several features (indexing, queuing, transforming, flow control, etc.) needed to provide the defined functionality and a connection with a remote data source holding the objects such as a LOR.

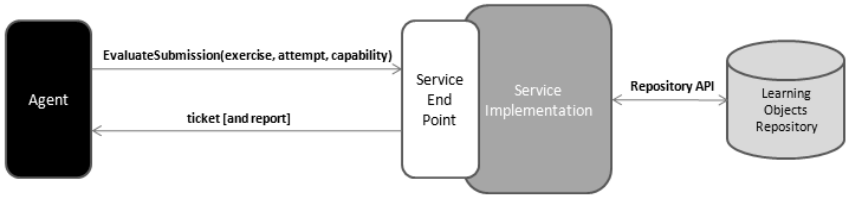

Figure 3 – The EvaluateSubmission function.

The evaluator returns a report on the evaluation, if it is completed within a predefined time frame. The report must contain information about the assessment of the attempt but should not reach to any conclusion. The raw data sent to the client can be used as input for other systems (e.g. classification systems, feedback systems).

In any case the response will include a ticket to recover the report on a later date. Requesting a report using a ticket is supported through another function called GetReport detailed in the next sub-subsection.

The **GetReport function** (Figure 4) allows a requester to get a report for a specific evaluation. The report included in this response may be transformed in the client side based on a XML stylesheet. This way the client will be able to filter out parts of the report and to calculate a classification based on its data. The request of this function includes a ticket sent previously by the service in response to an evaluation. The response returns a report about an evaluation.

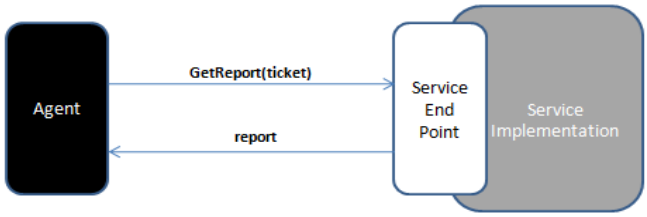

Figure 4 – The GetReport function.

4.2 The Evaluate - Programming Exercise service expression

In this section we define a new service expression, called *Evaluate - Programming Exercise*, that specializes the Evaluate service genre[2], modelling the evaluation of an attempt to solve an exercise defined as a learning object. Examples of this kind of exercise can be drawn from different domains; in this service expression we focus on the automatic evaluation of programming exercises.

The e-Framework model contains 20 distinct elements to describe a service expression, 9 of which are required elements, and the remaining either recommended or optional. For the sake of terseness the remainder of this section concentrates on the most significant of those elements.

4.2.1 Use & Interactions

The Use & Interactions element illustrates how the functions defined in the Requests & Behaviours section are combined to produce a workflow. An interaction involving the evaluator and two other service types, using the three main functions of the evaluator, is depicted schematically in Fig. 4 as an UML sequence diagram. The diagram includes three objects representing:

- Learning Management System - to manage the exercises suitable to specific learner's profiles;
- Evaluation Engine - to automatically evaluate and grade the students' attempts to solve the exercises;
- Learning Objects Repository - to store programming exercises and to retrieve those suited to a particular learner profile.

2 We completed the definition of this service genre and we expect to publish it shortly.

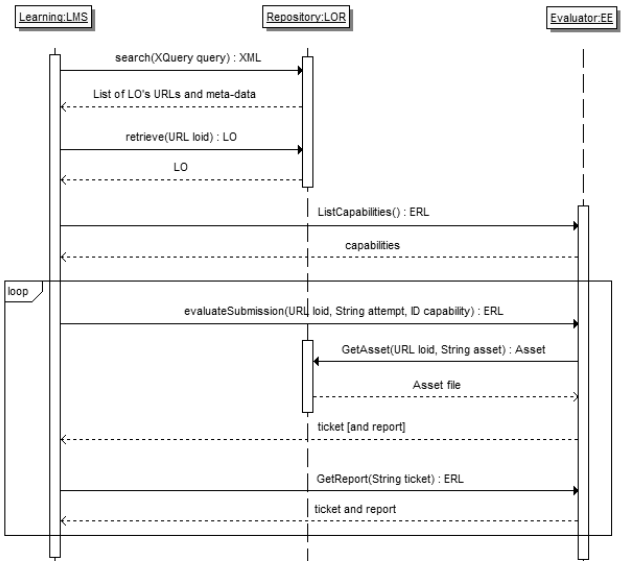

Fig. 5. Interacting with the evaluator.

The workflow presented in Fig. 5 starts with the configuration of an evaluation activity in an LMS (e.g. Moodle with an evaluation plugin). The configuration involves the selection of programming exercises and programming languages and will be carried out by a teacher. To select relevant programming exercises the LMS forwards the searches to a repository. To select programming language the LMS uses the ListCapabilities function of the evaluator.

During the evaluation activity itself the LMS iterates on the evaluation of all submissions. In general each student is able to make several submissions for the same exercise and an activity may include several exercises. Each evaluation starts with an EvaluateSubmission request from the LMS to the evaluator, sending a program and referring an exercise and a programming language. The evaluator retrieves the LO from the repository to have access to test cases, special correctors and other metadata. The response to of this function returns a ticket and an evaluation report, if the evaluation is completed within a certain time frame. The LMS may retrieve the evaluation report using the GetReport function with the ticket as argument.

4.2.2 Applicable Standards

The Applicable Standards element enumerates the names and versions of all the domain and technical standards, specifications and application profiles needed to provide the functionality of the service expression.

The pertinent e-learning content standards for this service expression are the IMS Content Packaging (IMS CP) [12] v1.1.4 final specification and the IEEE Learning Object Metadata (LOM). We introduce also a specification from a previous work [4] where we defined programming exercises as learning objects based on the IMS CP.

An IMS CP learning object assembles resources and meta-data into a distribution medium, typically a file archive in zip format, with its content described in a file named imsmanifest.xml at the root level. The manifest contains four sections: meta-data, organizations, resources and sub-manifests. The main sections are meta-data, which includes a description of the package, and resources, containing a list of references to other files in the archive (resources) and dependency between them.

This standard was defined for LO in general, not specifically for programming problems. In particular, the IMS CP schemata (including the IEEE LOM) lack features for describing all the resources required to perform the automatic evaluation of programming problems. For instance, there is no way to assert the role of specific resources, such as test cases or solutions. Fortunately, IMS CP was designed to be straightforward to extend it and thus we were able to use this standard for our purpose of defining programming problems as learning objects.

Meta-data information in the manifest file usually follows the IEEE LOM schema, although other schemata can be used. Since the meta-data related to the automatic evaluation cannot be conveniently represented using the IEEE LOM, it is encoded in elements of a new schema - the EduJudge Meta-data Specification (EJ MD).

The only e-learning interoperability standard relevant to this service expression is the IMS DRI specification [8]. It was created by the IMS Global Learning Consortium (IMS GLC) and provides a functional architecture and reference model for repository interoperability. The IMS DRI provides recommendations for common repository functions, namely the submission, search and download of LO. The IMS-DRI must be used by the evaluator with the LO repository.

4.2.3 Interface Definition

The Interface Definition element formalizes the interfaces of the service expression, namely the syntax of requests and responses of its functions. This particular service expression exposes its functions as SOAP and REST web services. The syntax of function requests in both flavours is summarized in Table 3.

Table 3. Service Expression function requests in SOAP and REST.

Function	Web Service	Syntax
ListCapabilities	SOAP	`ERL ListCapabilities()`
	REST	`GET /evaluate/ > ERL`
EvaluateSubmission	SOAP	`ERL Evaluate (Problem, Attempt ,Capability)`
	REST	`POST /evaluate/$CID?id=LOID < PROGRAM > ERL`
GetReport	SOAP	`ERL GetReport(Ticket)`
	REST	`GET $Ticket > ERL`

The remainder of this sub-section describes these functions in detail. All these functions respond with an XML document complying with the Evaluation Response Language (ERL). The ERL is formalised in XML Schema and covers the definition of the response messages for the three evaluator functions. The diagram depicted in the Figure 6 includes two main elements: `request` and `reply`. The former echoes the

request function and its parameters as received by the evaluation service and the later contains the output to that request.

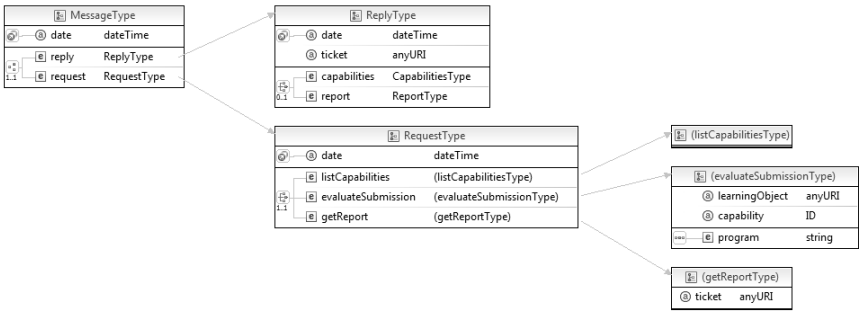

Fig. 6. The ERL schema.

The `request` element contains a different sub-element according to the function type. The `reply` element includes two sub-elements representing the possible responses of the service, more precisely, the `capabilities` and `report` elements. The `capabilities` element is used in a ListCapabilities response. This element has several capability sub-elements each with several `feature` elements to describe it. The `ticket` attribute holds a ticket to recover a report on a later date.

4.2.4 Usage Scenarios

The Usage Scenarios element characterizes the types of workflows in which the service expression is used. In our case these workflow types can be classified as curricular and competitive learning. In this sub-section we detail the requirements of these different scenarios.

Curricular learning in computer programming requires the evaluation of exercises in several moments such as practical classes, assignments and examinations. A programming evaluation service can be used in all three cases. Its usefulness in practical classes results from the instant feedback it provides to students, identifying the failed test cases and providing hints to resolve them. In programming assignments combining automatic and human evaluation both feedback and grading are relevant. In this scenario the student may submit multiple times, until a number of tests is passed, and receive automated feedback in the process. In examinations grading is the most relevant part and different grading policies can be implemented by the client based on the tests cases that were successfully completed.

Competitive learning relies on the competitiveness of students to increase their programming skills. This is the common goal of several programming contests where students at different levels compete such as: the International Olympiad in Informatics (IOI)[3], for secondary school students; the ACM International Collegiate Programming

3 IOI Official Web Site, www.ioinformatics.org

Contests (ICPC)[4], for university students; and the IEEExtreme[5], for IEEE student members. Each programming contest type has its own set of rules. In some cases students participate individually (as in IOI and IEEExtreme) in other cases they participate as a team (as in ICPC). Moreover, each contest has its own policy for grading and ranking submissions. For instance, IO assigns points to tests and ICPC just accepts a submission if it passes all tests, and gives a penalty for failed submissions when an exercise is accepted.

An implementation of the proposed service expression meets the evaluation requirements of this wide range of scenarios, from curricular and competitive learning. The evaluation report does not compute a grade, points or classification, nor produces a feedback for any particular scenario. However, all these can be easily computed by clients using a XSL transformation on the XML formatted report.

4.3 Text File Evaluation Service Usage Model

In the E-Framework, a Service Usage Model (SUM) describes the needs, requirements, workflows, management policies and processes within a domain. A SUM is composed of either Service Genres or Service Expressions, but not a mixture. In this section the SUM for the text file evaluation of learning objects is detailed. The E-Framework has 22 distinct elements to describe a SUM, 12 are required elements and the rest is either recommended or optional. For the sake of terseness just a subset of the SUM content based on the templates provided by the E-Framework is detailed. In concrete is described the SUM diagram, the technical functionality, the structure and arrangement of the functions and the data sources and services used.

The **SUM Diagram element**, depicted in Figure 7, defines a visual representation of the SUM for presentation purposes. This type of diagram is suggested by the E-F templates [12]. It organizes business processes in columns. For each business process the summary and name are highlighted in square rectangles in the top and the services genres it includes as ovals. Data sources are represented in the footer of the diagram.

In the first business process called Archive Learning Objects, the teacher searches in a repository for learning objects. Then, it selects the most appropriate and archives it, for instance, in a LMS for future use.

The Evaluate Learning Objects business process details the attempt of the student to solve a particular learning object and the request for its evaluation. In this business process the Evaluation Service Genre, detailed in the previous subsection, was used. This service includes the EvaluateSubmission function that returns ticket for a later report request and may also return an evaluation report. The report could be sent to both student and teacher or be transformed for a personalized notification about the evaluation of the students' attempt.

4 Official Web Site, http://icpc.baylor.edu/

5 IEEExtreme Official Web Site, http://ieeextreme.org/

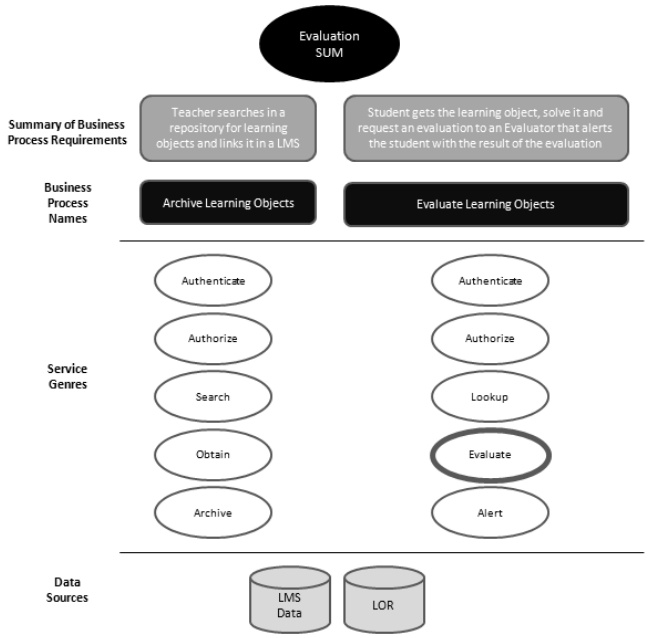

Figure 7 – The SUM diagram.

The **Functionality element** categorizes the functions supported by the SUM from a system viewpoint. The functions used in this SUM are organized as follows: common functions (Authenticate and Authorize), repository functions (Search, Obtain, Archive, Lookup and Alert) and evaluation functions (Evaluate).

The **Structure & Arrangement element** illustrates how a SUM is used in a particular business process by identifying the services used, data sources and their interactions within the SUM. An apt illustration of the use of this SUM is the pedagogical learning process involving the evaluation of programming exercises, presented in the following section.

5. Validation

To evaluate the usefulness of these service definitions we made a concrete definition of a service expression based on the proposed service genre, a programming exercise evaluation service. The evaluation of programming exercises involves the following types of services:

Learning Management System - to manage and retrieve the exercises to the learners. We chose the Moodle LMS since it is a free and open-source LMS with a significant share on the LMS market [3];

Learning Objects Repository - to persist LOs and related meta-information. We developed a specialized repository named crimsonHex [4] which currently stores more than 2000 programming exercises;

Evaluation engine - to evaluate and produce feedback on the learners' attempts to solve the exercise. We will use the Mooshak system as the evaluation engine based on a shared service [5];

Exercises Resolution Environment – to code the attempts of solving an exercise. We will use the Eclipse IDE since it is a free, widely used and open-source solution.

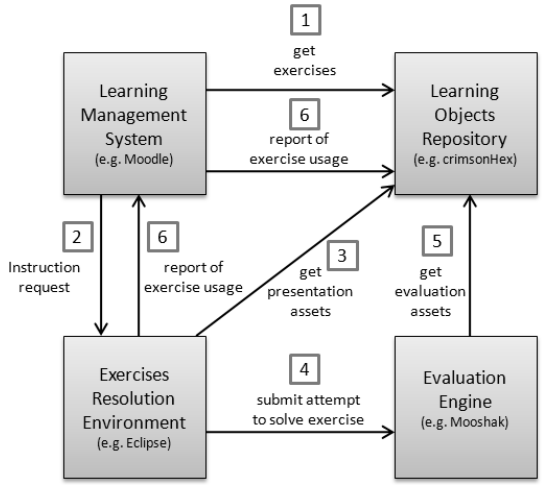

Figure 8 – Service integration in a pedagogical e-Learning process.

Figure 8 shows the integration of these services in a pedagogical learning process. In this particular scenario the teacher starts by setting a number of activities in the LMS, including the resolution of programming exercises. To select the relevant programming exercises the teacher 1) searches for relevant exercises in the repository. Then, the learner 2) tries to solve the exercises set be the teacher using an Experimentation Environment (e.g. Eclipse IDE). The IDE 3) recovers exercises descriptions from the repository showing them to the student. After coding the program the learner 4) send an attempt to the Evaluation Engine. The Evaluation Engine 5) recovers test cases from the repository. The learner may submit repeatedly, integrating the feedback received from the Evaluation Engine. In the end, the Evaluation Engine 6) sends a grade to the LMS that records it and reports the LO usage data back to the repository.

6 Conclusion and ongoing work

This paper presents a survey on eLearning frameworks and the contribution of an evaluation service for programming exercises to one of the most prominent eLearning frameworks - the E-Framework. The contribution includes three components of the E-Fremework's technical model: a service genre, a service expression and a service usage model.

In the Service Genre the authors made an abstract description of the behaviours expected from a text file evaluation service. Then, we add a new service expression that specializes the previous service genre by refining its behaviours and requests, and

also specify several implementation details such as applicable standards and interface definitions. Finally, in the Service Usage Model, we present the relationships between services through business processes and the usage scenario based on a particular domain - the automatic evaluation of programming exercises.

To evaluate the usefulness of these service definitions we made a concrete definition of an Evaluation service expression, more precisely, a programming exercise evaluation service. For this purpose, we are currently recasting Mooshak [2], a contest management system, in order to implement this service expression.

References

1. Leal, J.P. and Queirós, R.: eLearning Frameworks: a survey. Proceedings of International Technology, Education and Development Conference 2010, Valencia, Spain, (2010)
2. Leal, J.P and Silva, F.: Mooshak: a Web-based multi-site programming contest system Software, Practice & Experience, Volume 33 , Issue 6 (May 2003), Pages: 567 - 581, 2003, ISSN:0038-0644
3. Leal J.P and Queirós, R.: CrimsonHex: a Service Oriented Repository of Specialised Learning Objects, in Joaquim Filipe and José Cordeiro (Eds.) Proceedings of ICEIS'09: 11th International Conference on Enterprise Information Systems, pages 102-113, Milan, Italy, May 2009, ISBN: 978-3-642-01346-1.
4. Leal, J.P., Queirós, R.: Defining Programming Problems as Learning Objects - ICCEIT 2009 - International Conference on Computer Education and Instructional Technology, Venice, Italy, (2009)
5. Dagger, D., O'Connor, A., Lawless, S., Walsh, E., Wade, V.: Service Oriented eLearning Platforms: From Monolithic Systems to Flexible Services (2007)
6. IMS CC Specification, Version 1.0 Final Specification, http://www.imsglobal.org/cc/index.html
7. Bohl, O., Scheuhase, J., Sengler, R. and Winand, U.: The shareable content object reference model (SCORM)-a critical review, Proceedings of the International Conference on Computers in Education, 2002, pages 950-951
8. IMS DRI - IMS Digital Repositories Interoperability, 2003. Core Functions Information Model, http://www.imsglobal.org/digitalrepositories
9. C. Smythe: IMS Abstract Framework - A review, IMS Global Learning Consortium, Inc. (2003)
10. Wilson, S., Blinco, K. , Rehak, D. : An e-Learning Framework - Paper prepared on behalf of DEST (Australia), JISC-CETIS (UK), and Industry Canada, (2004)
11. e-Framework Technical Walk-through, http://www.e-framework.org/Portals/9/docs/e-Framework%20technical%20walk-through%20v1.1.pdf
12. IMS-CP – IMS Content Packaging, Information Model, Best Practice and Implementation Guide, Version 1.1.4 Final Specification IMS Global Learning Consortium Inc., http://www.imsglobal.org/content/packaging/#version1.1.4
13. Curbera, F., Duftler, M., Khalaf, R., Nagy, W., Mukhi, N. and Weerawarana, S., Unraveling the Web services web: an introduction to SOAP, WSDL, and UDDI, IEEE Internet computing, 2002, Volume 6, Issue 2, Pages: 86-93
14. Fielding, R.T. and Taylor, R.N. (2002-05), Principled Design of the Modern Web Architecture, ACM Transactions on Internet Technology (TOIT) (New York: Association for Computing Machinery) 2 (2): pages: 115–150, doi:10.1145/514183.514185, ISSN 1533-5399
15. Clark, D., Next-generation web services, IEEE Internet Computing, 2002, Volume 6, Issue 2, Pages: 12-14