Specifying a programming exercises evaluation service on the e-Framework

José Paulo Leal[1], Ricardo Queirós[2] and Duarte Ferreira[3]

[1,3]CRACS/INESC-Porto & DCC/FCUP, University of Porto, Portugal
zp@dcc.fc.up.pt, c0216010@alunos.dcc.fc.up.pt

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

Abstract. 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 purpose of this type of evaluator is 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. This paper formalizes the approaches to be used in the implementation of a programming exercise evaluator as a service on the e-Framework.

Keywords: SOA, interoperability, e-learning.

1 Introduction

In recent years several initiatives brought service orientation to e-learning. These initiatives, usually called e-learning frameworks, support the creation of flexible e-learning systems using service oriented approaches. Based on a previous survey [1] we identified the e-Framework as one of the most prominent e-learning framework initiatives. The e-Framework[1] success results from a strong and active community of practice contributing with service definitions. Potential submitters are encouraged to use the collaborative tools provided by the e-Framework to share their contributions and obtain feedback from the community.

In the research presented in this paper the e-Framework was used as basis for the definition of service for marking and grading computer programs. The computer programs processed by this service are submitted either by students in computer

1 Official website http://www.e-framework.org

programming courses, or by teams and contestants in programming contests. 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[2]. Both systems have automatic evaluation components that if recast as services could provide their functions to different types of e-Learning systems.

An implementation of the proposed service type evaluates an attempt to solve a programming exercise and produces a detailed report. This evaluation report includes information to support exercise assessment, grading and/or ranking by client systems. The report itself is not an assessment, does not include a grade and does not compare students. This kind of evaluation differs significantly from evaluations supported by most LMS, encoded in the IMS Question & Test Interoperability (QTI) specification. The data model of QTI was designed for questions with a set of pre-defined answers and cannot handle evaluation domains with specialized requirements, such as programming exercise evaluation. For instance, programming exercises evaluations requires tests cases, program solutions, compilation lines and other specific type of metadata that cannot be encoded in QTI. To cope with this problem the authors have already extended IMS Content Packaging (CP) definition of learning objects [4].

The remainder of this paper is organized as follows: section 2 details the evolution of e-learning towards the e-learning frameworks. The following section introduces the e-Framework and its technical model. Section 3 formalizes the approaches to be used in the implementation of the programming exercise evaluator, as requires by the e-Framework. As a contribution to the e-Framework, this work is a model of an evaluation service rather than report on its implementation. Nevertheless, we are planning the implementation of an evaluation service following this model using virtualization, as explained in the final section.

2 Current trends in e-learning

The evolution of e-learning systems in the last two decades was impressive. In their first generation, e-learning systems were developed for a specific learning domain and had a monolithic architecture [5]. Gradually, these systems evolved and became domain-independent, featuring reusable tools that can be effectively used virtually in any e-learning course. The systems that reach this level of maturity usually follow a component-oriented architecture in order to facilitate tool integration. An example of this type of system is the LMS that integrates several types of tools for delivering content and for recreating a learning context (e.g. Moodle, Sakai).

The present generation values the interchange of learning objects and learners' information through the adoption of new standards that brought content sharing and interoperability to e-learning. In this context, several organizations have developed specifications and standards in the last years. These specifications define, among

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

many others, standards for e-learning content [6, 7] and interoperability [8]. In spite of its adoption they have also been target of criticism. These systems based around pluggable and interchangeable components, led to oversized systems that are difficult to reconvert to changing roles and new demands such as the integration of heterogeneous services based on semantic information and the automatic adaptation of services to users (both learners and teachers). These issues triggered a new generation of e-learning platforms based on services that can be integrated in different scenarios. This new approach provides the basis for Service-oriented architecture (SOA). In the last few years there have been initiatives [9, 10] to adapt SOA to e-learning. These initiatives, commonly named e-learning frameworks, had the same goal: to provide flexible learning environments for learners worldwide. Usually they are characterized by providing a set of open interfaces to numerous reusable services organized in genres or layers and combined in service usage models. These initiatives use intensively the standards [6, 7] for e-learning content sharing and interoperability developed in the last years by several organizations (e.g. ADL, IMS GLC, IEEE).

Based on a previous survey [1], 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 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. A proposal for a new component must use the internal components of the technical model. This proposal might emerge from a technical project where many people with different skills are connected such as vendors, developers, technical people, IT Managers, institutions, hardware and software specialists. Hence, it’s crucial to the community have a basic understanding about the e-Framework Technical Model before contributing

The technical model of the e-Framework aims to facilitate system interoperability via a service-oriented approach [11]. The model provides a set of technical components enumerated in Table 1.

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.

Table 1. 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)

Service genres are technology-neutral descriptions of the behaviours of services. They can be bound to specific technologies by one or more service expressions. Service genres can also be abstracted from service expressions. Service expressions can be implemented in more than one way as service implementations, and these implementations can be deployed in more than one place as service instances. Standards provide the interoperability of the data and messages used in the services. Service implementations and instances may be referenced by the e-Framework through the technical model but are not part of the e-Framework Technical Model.

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 The Evaluate - Programming Exercise service expression

In the e-Framework a service expression is a specialization of a service genre specifying the particular implementation approaches to be used. In this section we define a new service expression, called *Evaluate - Programming Exercise*, that specializes the Evaluate service genre[3], 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.

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

4.1 Behaviours & Requests

The **Behaviours & Requests element** details technical information about the functions and operations of the service expression. The three types of request handled by this service expression are:

- **ListCapabilities:** provides the client systems with the capabilities of a particular evaluator;
- **EvaluateSubmission:** allows the request of an evaluation for a specific programming exercise;
- **GetReport:** allows a requester to get a report for a specific evaluation using a ticket.

The **ListCapabilities function** provides the client systems with the capabilities of a particular evaluator. Capabilities depend strongly on the evaluation domain. In a programming exercise the evaluator capabilities are related to the supported programming language compilers or interpreters. Each capability is described by a set of features; for a programming language they may be the language name (e.g. Java), its version (e.g. 1.5) and vendor (e.g. JDK).

The **EvaluateSubmission** function requests the evaluation of a program. The request of an evaluation is based on three parameters: a reference for a programming exercise described as a learning object, an attempt to solve the exercise and a specific capability to be used in evaluation (e.g. compile and execute as a Java program). The evaluator returns a report on the evaluation, if it is completed within a predefined time frame. In any case the response will include a ticket to recover the report on a later date.

The **GetReport function** returns a report for a specific evaluation using a ticket. The report contains detailed information on the evaluation but should not be view as an assessment, since it neither declares the attempt as acceptable, nor does it include a grade. The report sent to the client can be used as input for other systems (e.g. classification systems, feedback systems). 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.

4.2 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.

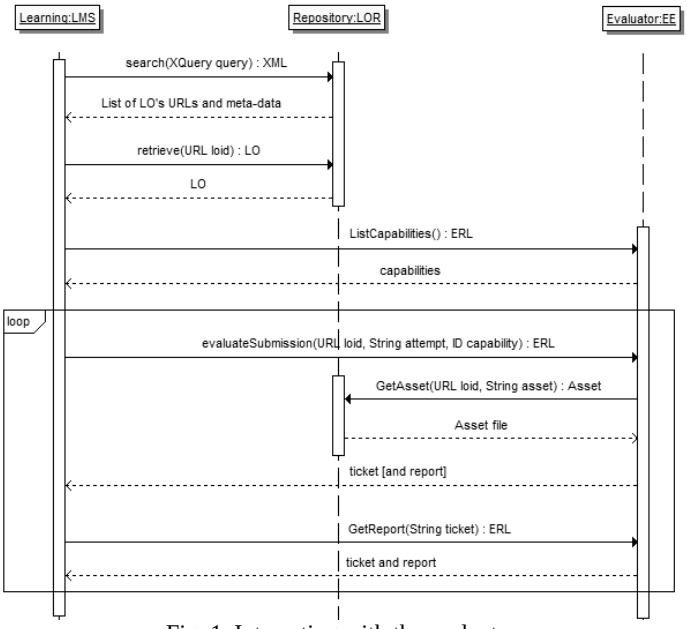

Fig. 1. Interacting with the evaluator.

The workflow presented in Fig. 1 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.

4.3 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.

There are no e-learning standards for interoperability with evaluators thus we focus on general communication standards such as those related with web service communication. There are two main web services flavours: Simple Object Access Protocol (SOAP) [13] and Representational State Transfer (REST) [14]. We propose that the service expression supports both flavours.

SOAP web services are usually action oriented, especially when used in Remote Procedure Call (RPC) mode and implemented by an off-the-shelf SOAP engine such as Axis [15]. REST web services are object (resource) oriented and implemented directly over the HTTP protocol, mostly to put and get resources. The reason to provide two distinct web service flavours is to encourage the use of the evaluator by developers with different interoperability requirements. A system requiring a formal an explicit definition of the API in Web Services Description Language (WSDL) [13], to use automated tools to create stubs, will select the SOAP flavour. A lightweight system seeking a small memory footprint at the expense of a less formal definition of the API will select the REST flavour.

4.4 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 2.

Table 2. 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 Fig. 2 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.

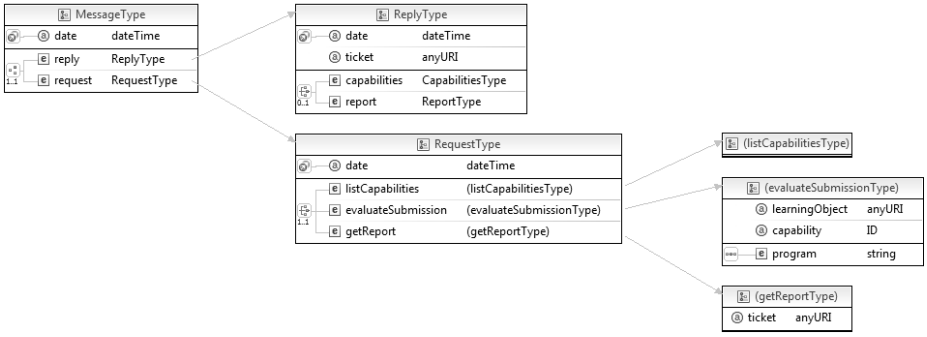

Fig. 2. The ERL schema.

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.

4.5 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)[4], for secondary school students; the ACM International Collegiate Programming Contests (ICPC)[5], for university students; and the IEEExtreme[6], 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.

5 Conclusion and ongoing work

This paper presents a contribution to the e-Framework consisting of an evaluation service for programming exercises. More precisely, we add a new service expression specializing an existing service genre refining its behaviours and requests, and specified implementation approaches such as applicable standards and interface definitions.

4 IOI Official Web Site, www.ioinformatics.org
5 Official Web Site, http://icpc.baylor.edu/
6 IEEExtreme Official Web Site, http://ieeextreme.org/

We are currently developing an evaluation engine based on this service expression. This implementation is based on Virtual Machines (VM) to execute the programs on a safe and controlled environment and is divided into five components, two controlling the evaluation service and other three supporting the execution of the programs on the VM. The five independent components give the evaluation engine a higher scalability. The use of VM allows us to manage a high number of capabilities such as languages and programming environments from different operating systems, including obsolete versions.

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