A contribution to the e-Framework - a specification of a programming exercise evaluation service

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

Technical Report Series: DCC-2010-0

Departamento de Ciˆencia de Computadores
Faculdade de Ciˆencias da Universidade do Porto
Rua do Campo Alegre, 1021/1055,
4169-007 PORTO,
PORTUGAL
Tel: 220 402 900 Fax: 220 402 950
http://www.dcc.fc.up.pt/Pubs/

A contribution to the e-Framework - a specification of a programming exercise evaluation service

José Paulo Leal
CRACS/INESC-Porto & DCC/FCUP, University of Porto, Portugal
zp@dcc.fc.up.pt

Ricardo Queirós
CRACS/INESC-Porto & DI/ESEIG/IPP, Portugal
ricardo.queiros@eu.ipp.pt

Duarte Ferreira
CRACS/INESC-Porto & DCC/FCUP, University of Porto, Portugal
c0216010@alunos.dcc.fc.up.pt

Abstract

This work is a contribution to the e-Framework, arguably the most prominent e-learning framework today, and consists of the definition of a service for the automatic evaluation of programming exercises. This evaluation domain differs from trivial evaluations modelled by languages such as the IMS Question & Test Interoperability (QTI) specification. Complex evaluation domains justify the development of specialized evaluators that participate in several business processes. These business processes can combine other type of systems such as Programming Contest Management Systems, Learning Management Systems, Integrated Development Environments and Learning Object Repositories where programming exercises are stored as Learning Objects. This contribution describes the implementation approaches used, more precisely, behaviours & requests, use & interactions, applicable standards, interface definition and usage scenarios.

Keywords: SOA, interoperability, e-learning.

I. 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, to cope with the heterogeneity of

the software environments found in most educational institutions. Based on a previous survey [1] we identified the e-Framework as one of the most prominent e-learning framework initiatives. The e-Framework success results from a strong and active community of practice contributing with definitions of service genres, expressions and usage models. Potential submitters are encouraged to use the collaborative tools provided by the e-Framework to share their contributions and obtain feedback from the community.

Our goal with this paper is to detail a contribution for the e-Framework consisting of a definition of a service for automatic evaluation of programming exercises. This evaluation domain differs significantly from trivial evaluations modelled by languages such as the IMS Question & Test Interoperability (QTI) specification. QTI describes a data model for questions and test data and was designed for questions with a set of pre-defined answers. Complex evaluation domains justify the development of specialized evaluators. Exposing this type of evaluation as a service will allow different types of systems to use it in several business processes. Examples of this type of systems are Programming Contest Management Systems, Learning Management Systems (LMS), Integrated Development Environments (IDE) and Learning Object Repository (LOR) where programming exercises are stored as Learning Objects (LO).

The contribution is a specialization of a non-trivial evaluation service genre as a service expression. In the service expression we formalize the implementation approaches, namely, behaviours & requests, use & interactions, applicable standards, interface definition and usage scenarios. An implementation of the proposed service type evaluates an attempt to solve a programming exercise and produces a detailed report. This 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.

The remainder of this paper is organized as follows: section 2 details the evolution of e-learning towards the e-learning frameworks. In the following section we present the e-Framework, more precisely, its technical and contribution model. Then, we detail our contribution to the e-Framework with a service expression for evaluating programming exercises. Finally, we conclude with a summary of the major contributions of this paper and our current work in this project.

II. 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 [2]. 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 many others, standards for e-learning content [3, 4] and interoperability [5]. 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, the automatic adaptation of services to users (both learners and teachers), and the lack of a critical mass of services to supply the demand of e-learning projects. 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 [6, 7, 8] 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 [3, 4] 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.

III. 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 [9]. 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 next subsections details the architecture of the e-Framework, more precisely, its technical and contribution model.

A. Technical Model

The technical model of the e-Framework aims to facilitate system interoperability via a service-oriented approach [10]. 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.

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)

Table 1. *Technical Model.*

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.

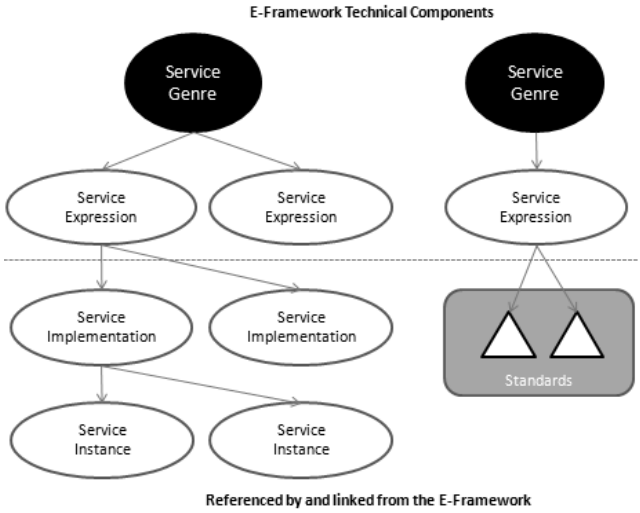

Fig. 1. Relationship of e-Framework concepts.

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. The relationship between these e-Framework concepts is represented diagrammatically in Fig. 1.

B. Contribution Model

To fulfil its vision of service oriented e-learning systems, integrating reusable, interoperable services, the e-Framework seeks to establish a knowledge base, shared by international education and research communities. The collaborative development of that this knowledge base relies on the contributions of a community of practice.

To participate in the e-Framework the contributor must have a precise understanding of how the contribution process works. This process is formally described by a contribution model.

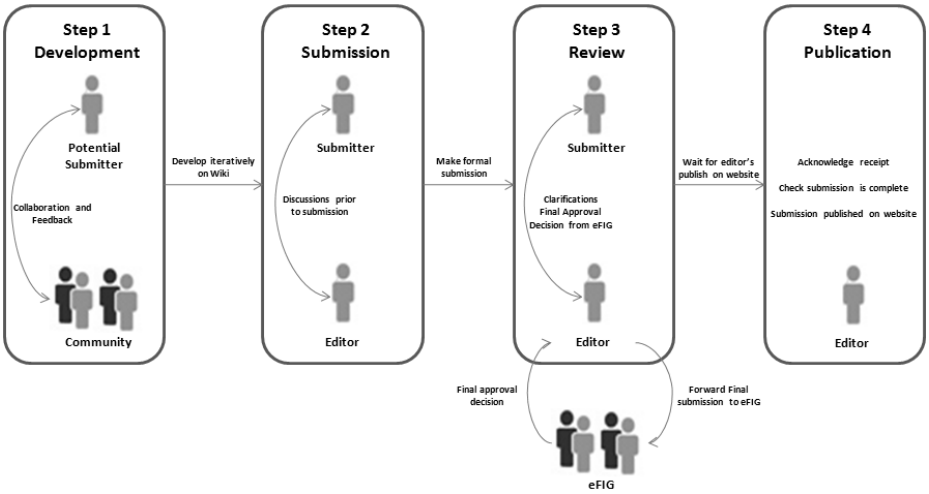

Fig. 2. Contribution model in the e-Framework.

The diagram in Fig. 2 shows the submission process for the contribution of documentation about technical components. The steps involve a series of interactions between a potential submitter, the community, the e-Framework technical editor and the e-Framework Integrity Group (eFIG). The eFIG is the panel that handles the final decisions regarding the contributions to the e-Framework.

Currently, the e-Framework includes a knowledge base with 46 service genres, 6 service expressions and 21 SUMs. However this number will increase as more components finished the above steps of the contribution model.

IV. THE EVALUATE – PROGRAMMING EXERCISE SERVICE EXPRESSION

A service expression is a specialization of a service genre specifying the particular implementation approaches used. In this section we define a new service expression specializing the Evaluate service genre[1], 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. The Table 2 presents the required elements names and the values assigned for the Evaluate – Programming Exercise service expression.

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

Required elements	Values
Name	Evaluate – Programming Exercise
Classification	• Maturity: mature • Development status: production • State behaviour: stateless • Transactional behaviour: non-transactional • Service End Point: Transcoder (both requests and provides) • Authorized: yes • Protocol Bindings: REST, SOAP • Status: unapproved
Service Genre	Non-trivial Evaluate service genre
Version	1.0
Description	The service evaluates a student's attempt to solve a programming exercise and produces a detailed report on the evaluation of an attempt.
Functionality	• listing of the programming languages supported by the evaluator; • evaluating a computer program that attempts to solve an exercise in a given programming language; • reporting on an evaluation.
Behaviours & Requests	
Use & Interactions	Detailed in the following subsections.
Applicable Standards	

Table 2. *Required elements of for the Evaluate – Programming Exercise service expression*

For the sake of terseness we describe just a subset of the service expression content based on the templates provided by the e-Framework, more precisely:

- Required elements: Behaviours & Requests, Use & Interactions, Applicable Standards;
- Recommended elements: Interface Definition;
- Optional elements: Usage Scenarios.

A. Behaviours & Requests

The **Behaviours & Requests element** details technical information about the functions and operations of the service expression. The use case diagram on Fig. 3 shows the three types of request handled by this service expression.

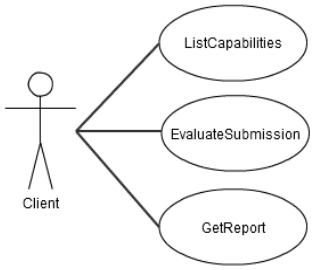

Fig. 3. Use Cases diagram.

- **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, in the sense that does not declare the attempt as acceptable, does not 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.

B. 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.

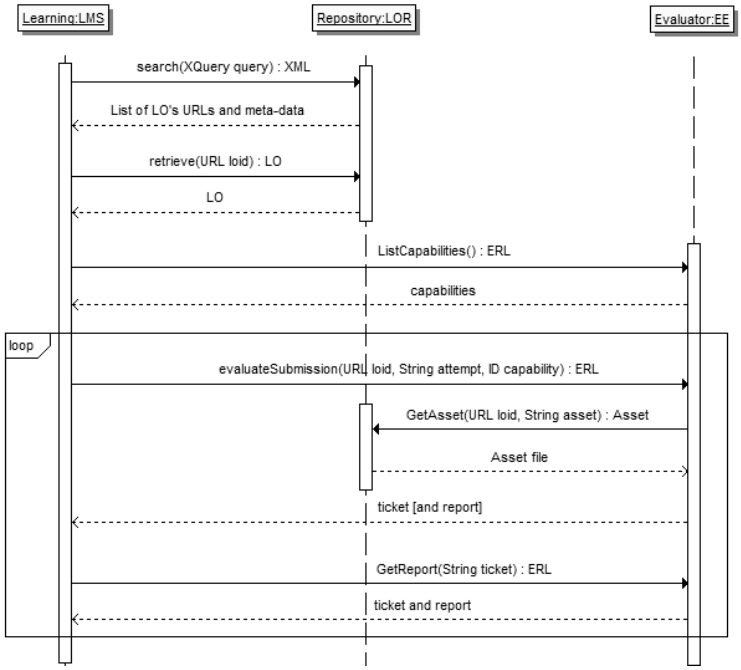

Fig. 4. Interacting with the evaluator.

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.

The workflow presented in Fig. 4 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.

C. 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. We organize them in content and communication, the former including e-learning standards and specifications, and the later including e-learning interoperability and web services standards.

Content standards and specifications

The pertinent e-learning content standards for this service expression are the IMS Content Packaging (IMS CP) [11] v1.1.4 final specification and the IEEE Learning Object Metadata (LOM). We introduce also a specification from a previous work [12] 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.

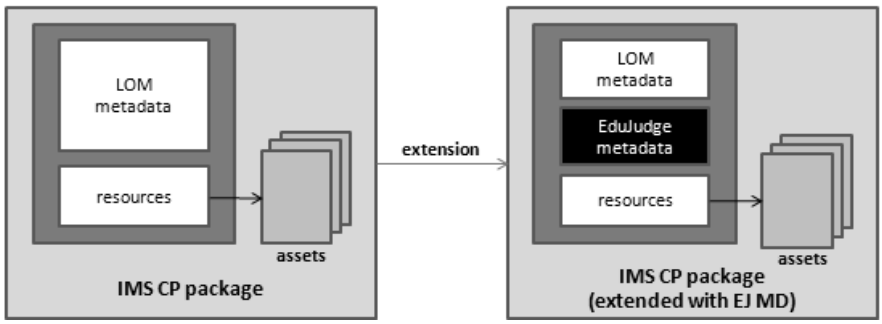

Fig. 5. The extension of the IMS CP specification to describe programming exercises.

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) as shown in Fig.5.

All these standards as well the message binding of this service expression are described by schema languages, most often using the XML Schema Definition language (XSD). This language overcame Document Type Definition (DTD) limitations and provided several advanced features, such as, the ability to build new types derived from basic ones, manage relationships between elements (similar to relational databases) and combine elements from several schemata.

Communication standards

The only e-learning interoperability standard relevant to this service expression is the IMS DRI specification [5]. 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) [16], 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.

D. 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.

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

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

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) that we describe in the first sub-subsection. The following sub-subsections describe the use of each function with examples of requests and the respective responses in ERL.

Evaluation Response Language

The Evaluation Response Language (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. 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.

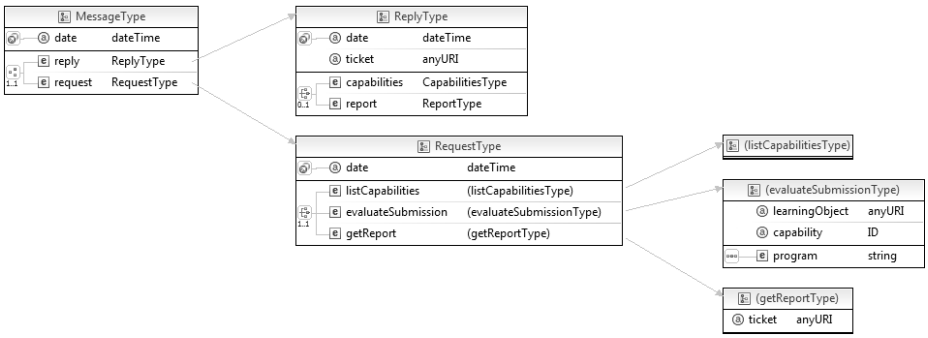

Fig. 6. The ERL schema.

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 as shown in Fig. 7, 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.

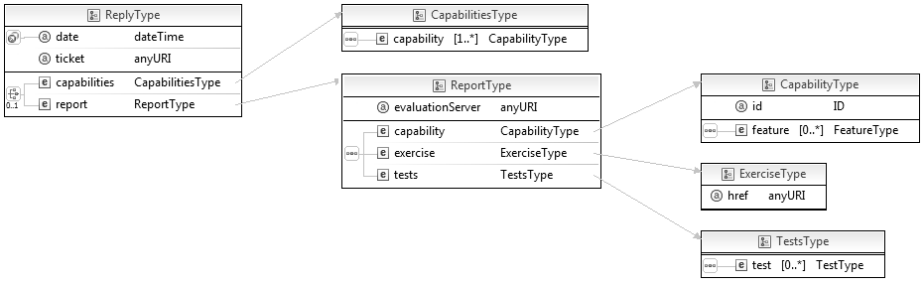

Fig. 7. The reply element.

The report element contains a detailed evaluation report. It has a single mandatory evaluationServer attribute representing the URL of the evaluator. This element also includes the following sequence of sub-elements:

- capability: a specific evaluator capability used to evaluate this attempt;
- exercise: a reference to the Learning Object and the title of the exercise;
- tests: contains a set of tests for the evaluation of the submitted attempt. Each test element represents a test case describing resources supplied to evaluate the submitted program.

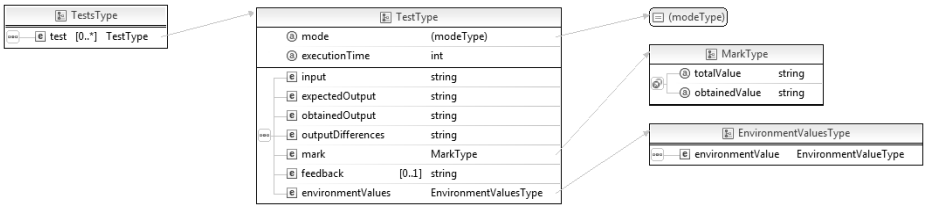

Fig. 8. The test element.

As shown in Fig. 8, each test corresponds to a single test case that can be repeated to create a test set. The submitted program is executed once for each test element, receiving as input the content of the input element. The resulting output, stored in the obtainedOutput element, is compared to the expected output contained in the expectedOutput element. The outputDifferences element describes the differences between the two previous elements using the syntax of the Unix diff command [17].

The test element contains also data for grading and correcting programs. This element includes a mark element to assign a mark for a successful execution. The client may compute a grade for the submission as the sum of the marks of successful executions. The optional feedback element contains detailed feedback for an unsuccessful execution. The environment values are a list of property-value pairs that may be supplied by the execution environment. For instance, if the execution environment is able to report the memory usage of a program execution then this data is recorded in this element.

ListCapabilities function

The ListCapabilities function returns the capabilities of a specific evaluator. Using the REST API this operation is performed by sending a GET HTTP request to the evaluator, as in the following example.

GET http://eval.domain.org/evaluate > ERL

The following document is an example of the HTTP response complying with the ERL specification.

```
<?xml version="1.0" encoding="UTF-8"?>
<message date="2010-12-31T12:00:00">
 <request date="2010-12-31T12:00:00">
  <listCapabilities/>
 </request>
 <reply date="2010-12-31T12:00:00">
  <capabilities>
   <capability id="Java1.5">
    <feature name="Language" value="Java"/>
    <feature name="Language Version" value="1.5"/>
   </capability>
   <capability id="Java1.6">
    <feature name="Language" value="Java" />
    <feature name="Language Version" value="1.6"/>
   </capability>
  </capabilities>
 </reply>
</message>
```

EvaluateSubmission function

The EvaluateSubmission function evaluates a computer program that attempts to solve an exercise in a given programming language. Using the REST API this operation is performed by sending a POST HTTP request to the server, as in the following example.

POST http://eval.domain.org/evaluate/java1.6?
id=http://lor.domain.org/lo/123 < PROGRAM > ERL

The HTTP parameter id is a reference to a LO with the programming exercise. The PROGRAM is an attempt to solve it. The ERL is the content of the HTTP response to the above request. It includes an XML file complying with the ERL specification and containing a ticket.

```
<?xml version="1.0" encoding="UTF-8"?>
<message date="2001-12-31T12:00:03" >
 <request date="2001-12-31T12:00:01">
  <evaluateSubmission
            capability=”Java1.6” learningObject=”http://lor.domain.org/lo/123”>
   <program><![CDATA[  … program code here …  ]]></program>
  </evaluateSubmission>
 </request>
 <reply
            date="2001-12-31T12:00:02" ticket=”https://eval.domain.org/report/123/xpto”/>
  </reply>
</message>
```

GetReport function

The GetReport function returns a report on an evaluation attempt. Using the REST API this operation is performed by sending a GET HTTP request to the evaluator, as in the following example.

```
GET https://eval.domain.org/report/123/xpto > ERL
```

The URL is the ticket obtained in the last request. The following is an example of the HTTP response.

```
<message date="2001-12-31T12:00:00">
 <request date="2001-12-31T12:00:00">
  <getReport ticket=”https://eval.domain.org/report/123/xpto”/>
 </request>
 <reply date="2001-12-31T12:00:00">
  <report evaluationServer=”https://eval.domain.org/”>
   <capability id="Java1.6"/>
   <exercise href="http://lor.domain.org/lo/123”>
    A very simple Problem</exercise>
    <tests>
     <test executionTime="100" mode="program">
      <input>1,2,3,4</input>
      <expectedOutput>4</expectedOutput>
      <obtainedOutput>4</obtainedOutput>
      <outputDifferances></outputDifferences>
      <mark obtainedValue="1" totalValue="1"/>
      <feedback></feedback>
      <environmentValues>
       <environmentValue name="memory" value="12kb" />
      </environmentValues>
     </test>
    </tests>
   </report>
  </reply>
</message>
```

E. 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) [18], for secondary school students; the ACM International Collegiate Programming Contests (ICPC) [19], for university students; and the IEEExtreme [20], 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.

V. CONCLUSION AND ONGOING WORK

This paper presents a contribution to the e-Framework consisting of a non-trivial 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.

The main contribution of this work is the proposal of a new service expression itself. A secondary contribution is the description of the e-Framework technical and contribution models that may prove useful to other persons or organizations considering a similar contribution.

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] Dagger, D., O'Connor, A., Lawless, S., Walsh, E., Wade, V.: Service Oriented eLearning Platforms: From Monolithic Systems to Flexible Services (2007)

[3] IMS CC Specification, Version 1.0 Final Specification, http://www.imsglobal.org/cc/index.html

[4] SCORM specification, http://www.adlnet.gov/Pages/Default.aspx

[5] IMS DRI - IMS Digital Repositories Interoperability, 2003. Core Functions Information Model, http://www.imsglobal.org/digitalrepositories

[6] C. Smythe: “IMS Abstract Framework - A review", IMS Global Learning Consortium, Inc. (2003)

[7] S. Wilson, K. Blinco, D. Rehak: “An e-Learning Framework” - Paper prepared on behalf of DEST (Australia), JISC-CETIS (UK), and Industry Canada, (2004)

[8] Schools Interoperability Framework, http://www.sifassociation.org

[9] Official website of e-Framework for Education and Reseach, http://www.e-framework.org

[10] e-Framework Technical Walk-through, http://www.e-framework.org/Portals/9/docs/e-Framework%20technical%20walk-through%20v1.1.pdf

[11] 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

[12] Leal, J.P., Queirós, R.: Defining Programming Problems as Learning Objects - ICCEIT 2009 - International Conference on Computer Education and Instructional Technology, Venice, Italy, (2009)

[13] SOAP (Simple Object Access Protocol), Version 1.2. Part 0: Primer, 2nd edition, http://www.w3.org/TR/2007/REC-soap12-part0-20070427/ (2007)

[14] Fielding, R.: Architectural Styles and the Design of Network-based Software Architectures, *Phd dissertation*, http://www.ics.uci.edu/~fielding, (2000) /pubs/dissertation/rest_arch_style.htm

[15] Apache Axis, 2006. Project homepage, http://ws.apache.org/axis/

[16] WSDL, 2007. Web Services Description Language (WSDL) Version 2.0 Part 1: Core Language, http://www.w3.org/TR/wsdl20/

[17] Johnson, M.: Diff, Patch, and Friends. Linux Journal, Volume 1996, Issue 28es, (1996)

[18] IOI Official Web Site, www.ioinformatics.org

[19] ACM International Collegiate Programming Contest (ICPC) Official Web Site, http://icpc.baylor.edu/

[20] IEEExtreme Official Web Site, http://ieeextreme.org/