Journal of Universal Computer Science, vol. 19, no. 14 (2013), 2127-2149
submitted: 3/2/13, accepted: 29/7/13, appeared: 1/8/13 © J.UCS

Ensemble - an E-Learning Framework

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

José Paulo Leal
(CRACS & INESC-Porto LA, Faculty of Sciences, University of Porto
Porto, Portugal
zp@dcc.fc.up.pt)

Abstract: E-Learning frameworks are conceptual tools to organize networks of e-learning services. Most frameworks cover areas that go beyond the scope of e-learning, from course to financial management, and neglects the typical activities in everyday life of teachers and students at schools such as the creation, delivery, resolution and evaluation of assignments. This paper presents the Ensemble framework - an e-learning framework exclusively focused on the teaching-learning process through the coordination of pedagogical services. The framework presents an abstract data, integration and evaluation model based on content and communications specifications. These specifications must base the implementation of networks in specialized domains with complex evaluations. In this paper we specialize the framework for two domains with complex evaluation: computer programming and computer-aided design (CAD). For each domain we highlight two Ensemble hotspots: data and evaluations procedures. In the former we formally describe the exercise and present possible extensions. In the latter, we describe the automatic evaluation procedures.
Key Words: e-learning, interoperability, standards, automatic evaluation.
Category: D.2.12, D.2.11, D.3.0

1 Introduction

In recent years the concept of E-Learning Framework emerged associated with several initiatives of educational organizations [Leal & Queirós, 2010a].

Some of these frameworks provide guidance in planning and designing e-learning materials and designing distributed learning systems for public and private universities such as Badrul Khan's eight dimensional e-learning framework [Khan, 1997]. Others frameworks address the heterogeneity of the hardware and software environments found in most educational institutions, many of which are not replaceable and are extremely important to the institution. These initiatives allow institutions to develop their own architectures, using a Service Oriented Architecture (SOA) [Girardi, n.d.]. These frameworks provide several layers of services to support the development and management of e-learning systems. Existing frameworks cover areas that go beyond the scope of e-learning, from course to financial management [Leal & Queirós, 2010a].

In this paper we present a proposal for an e-learning framework called the Ensemble E-Learning Framework (EeF). The EeF is a conceptual tool to organize a network of e-learning systems and services based on content and communications specifications. The name "Ensemble"[1] suggests the collaborative work of all the parties in a network to achieve a common goal. The EeF differs from the existing frameworks since it is exclusively focused on the teaching-learning process through the coordination of pedagogical services that are typical in everyday life of teachers and students at schools such as the creation, delivery, resolution and evaluation of assignments.

This framework also emphasizes the use of assessment services to automatically evaluate the attempts of students to solve exercises and to produce relevant feedback on their quality. The need for automatic assessment exists in different domains such as, management, health sciences, electronics. Playing business games in management courses, or simulating a human patient in life sciences courses, or simulating an electronic circuit in electronics courses are examples of learning processes that require the use of special authoring, rendering and assessment tools. These tools should be integrated in instructional environments in order to provide a better learning experience. However, these tools would be too specific to incorporate in an e-learning platform. Even if they could be provided as pluggable components, the burden of maintaining them would be prohibitive to institutions with few courses in those domains.

The proposal of the framework is organized in two parts: formal description and validation. In the description of the framework we present its data, integration and evaluation models that must base the implementation of networks in specialized domains with complex evaluations. In the validation of the framework we specialize the framework for two domains with complex evaluation: computer programming and computer-aided design (CAD) domains. For each domain we describe two frameworks hotspots related with the data and the evaluation model.

2 E-Learning Frameworks

Over the years the word framework has been used to define a work environment specially designed to solve common and complex problems in different domains. Due to its broad definition it is often used as a buzzword, especially when applied to software. A software framework may include support programs, runtime environments, code libraries and other tools, in order to assist the developer in a software project. Usually the functions of a framework are exposed through an API. The code provided by the framework is usually divided in frozen-spots (services already developed in the framework) and hotspots (set of common code that

[1] In music an ensemble is a group of people who perform instrumental or vocal music.

must be overridden or specialized by user code) [Markiewicz & de Lucena, 2001]. Hence, this twofold code feature amongst with the inversion of control is one of distinguishing keys that separate the current frameworks from generic code libraries. An e-learning framework can be defined as a specialized software framework. In the e-learning field, this term has been associated with several initiatives to adapt SOA to e-learning. Based on service oriented approaches the process (Figure 1) of moving from a framework to a working implementation can be defined by four key concepts [Wilson *et al.* , 2004]: **Vocabulary** - describes all possible "services" for a domain such as e-learning; **Reference Model** - combines these services for specific learning-teaching requirement; **Design** - specifies the use of standards and specifications for these combinations; and **Artifact** - implements (software, process, workflow) a design. A Framework provides a vo-

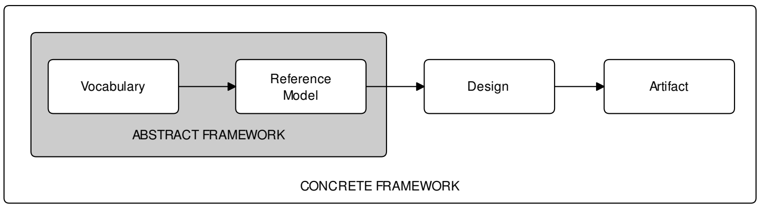

Figure 1: Simple Framework model [Wilson *et al.* , 2004].

cabulary 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 this model two types of frameworks were identified:

Abstract frameworks aim only at the creation of specifications, recommendations and best practises for the development of e-learning systems. Examples of this category are the IEEE Learning Technology Systems Architecture[2], the Open Knowledge Initiative[3] and the IMS Abstract Framework[4].

Concrete frameworks extend the goals of abstract frameworks by providing also complete service designs and/or components that can be integrated in actual implementations of artifacts. Examples are the Open University Support

[2] Architecture & Reference Model - Working Group 1 Web Site - http://ltsc.ieee.org/wg1/

[3] Open Knowledge Initiative (OKI) Web Site - http://sourceforge.net/projects/okiproject/

[4] IMS Abstract Framework Web Site - http://www.imsglobal.org/af/

System [5] , the Schools Interoperability Framework [6] and the E-Framework [7].

A comparative study was made using these e-learning frameworks regarding: impact and maturity, architectural models, adopted standards and user groups.

Table 1: E-Learning frameworks survey.

Facets	Features	LTSA	OKI	IAF	SIF	E-F
Impact and Maturity	Creation date	1996	2001	2003	2003	2007
	1st vs. date	1996	2003	2003	2003	-
	Last vs. date	2001	2006	2003	2009	-
	Cited proj.	-	3	-	37	4
	Contributions	inactive	yes	yes	yes	yes
Architec. Models	Main Model	layered	layered	layered	flat	layered
	SOA	yes	yes	yes	yes	yes
Adopted Standards	Content	-	-	-	SCORM	SCORM,CP
	Metadata	LOM	LOM	LOM	LOM	LOM,DC
	Web Service	SOAP	SOAP	S/REST	S/REST	S/REST
	Bindings	-	PHP	JAVA	JAVA	JAVA
User Groups	Framework	ESV	ESV	IMS	ESV	ESV
	End Users	HE	HE	HE	K-12	HE

From Table 1 one 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-F and SIF. In the case of the E-F, it has been receiving great amount of input from the e-learning community. On the other hand, SIF is the most widely used framework with 37 cited projects in the project web site.

All the 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 that orchestrates 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

[5] Official website of OpenUSS - http://openuss.sourceforge.net
[6] Schools Interoperabiliy Framework (SIF) Web Site - http://www.sifinfo.org/
[7] Official website of e-Framework for Education and Research - http://www.e-framework.org

due to the fact that these projects are cooperating [Curbera *et al.* , 2002].

In regard to standards some of them 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, the Educational Software Vendors 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 e-learning systems end users, the Higher Education sector is the most targeted.

Based on this survey, one can conclude that E-F and 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-F, the contribution can be done by proposing new service genres, service expressions and service usage models. On SIF this type of contribution cannot be done to the concrete framework. However, new agents can be developed, such as those related with learning objects repositories.

3 The Ensemble framework

This section presents a proposal for an e-learning framework called the Ensemble E-Learning Framework (EeF). The EeF is a conceptual tool to organize a network of e-learning systems and services based on content and communications specifications. This framework differs significantly from the frameworks presented in the last section in two facets: focus and architectural model. The EeF is exclusively focused on the teaching-learning process through the coordination of pedagogical services that are typical in everyday life of teachers and students at schools such as the creation, delivery, resolution and evaluation of assignments. Another distinctive feature of EeF is its architectural model. The EeF uses a model that can be described as a "decentralized orchestration". An implementation of the EeF uses a pivot component that orchestrates the communication with other services but is replicated and deployed for each end user. This novel approach avoids any single-point-of-failure issues that occur in central orchestrations.

In the following subsections the framework is described based on four models. Firstly, the architectural model and its components are presented. Then, the interoperable facet of the framework is addressed by presenting its data, integration and evaluation model.

3.1 Architectural Model

The EeF is the basis for the design and implementation of **Ensemble instances** as realizations of the framework for specific domains (e.g. computer programming learning). Each instance can be deployed in several locations denominated as **Ensemble networks**. Several users can interoperate within a network using Web Services. The definition of how these Web services cooperate is typically based on coordination models (e.g. orchestration, choreography). In the EeF architectural model the services coordination is based on a "decentralized orchestration" where central components are replicated for each end user. This is a distinctive feature of this framework. Most e-learning frameworks fall in one of two architectural models: either based on layers of services, or on central communication nodes. Both architectural models present communication issues: layered models present unintended noise between the communication of non-contiguous layers and central communication nodes include a single-point-of-failure since all the communication relies in a central node. With this novel approach it may appear that the replication of components in the execution path would adversely affect performance, however decentralized execution brings performance benefits such as, there is no centralized coordinator which can be a potential bottleneck and distributing the data reduces network traffic and improves transfer time [Chafle *et al.* , 2004] or [Chafle *et al.* , 2004].

Figure 2 shows the architectural model of the EeF. On the central axis that is perpendicular to the plan holding the network services resides the central components called axial systems. These central components communicate with services organized at two levels: core services that are crucial for the learning process or secondary services that are used to complement core services in a specialised task.

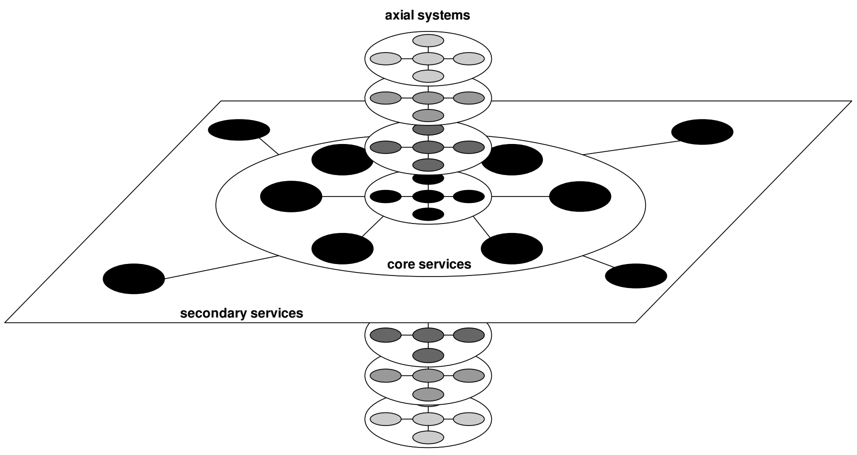

Figure 2: EeF architectural model.

The remainder of this section defines in detail the concepts of **axial systems**, **core services** and **secondary services**.

3.1.1 Axial systems

Axial systems are central components in the EeF architectural model. The main features of these systems are:

Centrality - they are able to communicate with all core services;

Locality - they are replicated on the computer of each end user of a specific network;

Interactivity - they mediate the interaction between the users (teachers and students) and the network by means of a user interface.

One of such systems assume a **pivot role**. The role of the pivot component is twofold: orchestration and interface.

The pivot component orchestrates the communication among services and is replicated for each end user. Since it is distributed over each network user this approach prevents the occurrence of any single-point-of-failure issues that might occur.

The pivot component also acts as the graphical interface between users and the network. In the EeF jargon a pivot component is called a Teacher Assistant (TA). A Human TA is a person who assists a teacher in practical classes. The task of a automated TA is to act as an interface to users (both with teachers and students) and, unlike its human counterpart, to delegate most of its work to others, as it is fundamentally a coordinator of e-learning systems. For instance, in a programming course, an automated TA is used to help students with programming tools (integrated programming environments, compilers, and debuggers), check if they have solved the exercises and provide feedback to help them to overcome their difficulties. This type of tool can also be described as a scaffolding tool since it complements existing tools and was designed to be easily removed once it is no longer needed.

Apart from the TA, other systems reside in this axial area. Using again the programming domain as example, an experimentation system (e.g. IDE) can be used by students for solving programming exercises and may be extended to communicate with other services in the network. In the medical training domain, teachers use simulation models to improve students skills in several medical processes (e.g. birth) and to use medical tools properly, as they would in real world hospitals and clinics.

All these systems need to communicate with other services. The TA may need to interact with an assessment system to submit a student attempt to solve an exercise. A business simulation game may require a repository containing specialized LO describing simulations. Due to size, complexity and security issues, these services should be accessed remotely rather than being installed locally.

3.1.2 Core and Secondary services

An Ensemble instance handles multiple pedagogical learning process. A learning process is a collection of related and structured activities implemented by e-learning services. A typical example of a pedagogical learning process is a classroom assignment. The teacher starts by setting a number of activities in the LMS, including the resolution of a number of relevant exercises in a specific domain obtained from a repository. The learner tries to solve the exercises set by the teacher using an experimentation system that recovers exercise descriptions from the repository. After solving the exercise the learner sends an attempt to an assessment system. The learner may submit repeatedly, integrating the feedback received from the assessment system. In the end the assessment system sends a grade to the LMS gradebook. Most of these services are commonly provided by e-learning systems such as: Learning Management Systems - to manage and retrieve the exercises to the learners; Learning Objects Repositories - to persist LO and related meta-information; Assessment Systems - to evaluate and produce feedback on attempts to solve exercises; and E-Portfolio systems - to organize students achievements.

These types of services are very different in nature. Repositories and Assessment Systems provide truly specialized services. An LMS is not in strict sense a service. It is a system designed to be a complete and generic e-learning environment rather than a service. Nevertheless, since a typical LMS is a component based system, it may be extended to incorporate the features it lacks to communicate with other services.

Secondary services are complementary services that complement the core services in a specific task, although its absence does not alter the execution flow of a learning process. Usually these services do not have graphical interfaces and are more specialized than the core services. An example of this kind of services is an adaptation service. Taking the previous example, an adaptation service could adjust the presentation order in accordance with the effective difficulty of the exercises (not the difficulty stated on the LO) and the needs of a particular student. Other example is a service for handling the conversion between different exercises formats.

Another example of a secondary service is a social media service that resides on the cloud and can be used to integrate social features from a Social Media Platform (SMP) such as Facebook or Twitter in the Ensemble framework. In this context, a social service could set/get information to/from social networks such as profile information (user data), friends information (social graph) and activities.

3.2 Data Model

The concept of Learning Object (LO) is the cornerstone for producing, sharing and reusing content in e-learning. The most widely used standard for LO is the IMS CP that uses the LOM standard to describe the learning resources included in the package. The QTI specification endows this data model with the capacity for describing questions and test data. Despite its widespread use, this specification is not adequate to specific domains [Leal & Queirós, 2009, Queirs & Leal, 2011]. Recently, IMS GLC proposed the IMS CC that bundles the previous specifications and its main goal is to organize and distribute digital learning content.

The Ensemble data model specification is based on the Common Cartridge specification. This choice is sustained by some experiments [Kurilovas, 2012] in the deployment of CC packages in an educational context and is also justified by the following features:

Packaging - flexible packaging via URL references to web content;

Communication - Web 2.0 mash-ups (provisioning of IMS LTI);

Security - content authorization via protected resources;

Content sharing - migration from other data sources (e.g. SCORM 2004);

Extension - data augment by using new Learning Application Objects (LAO);

Integration - access to learner data by privileges and outcomes (e.g. IMS LIS).

Of all these reasons, the extension facet is the most relevant. The extension can be achieved by adding new LAO resources in the CC package. Each LAO must have a corresponding resource element in the manifest. Physically, a LAO is a directory in the content package containing a descriptor file, its schema and additional files and sub-directories used exclusively by that LAO. These files held within a LAO directory structure are described as associated content resources. LAO resources differ from web content resources (e.g. HTML and image files) since they require additional processing and interpretation before they can be imported and subsequently used within the target system. Examples of LAO resources predefined in the CC specification are the QTI assessments and discussion forums.

Figure 3 shows the structure of a CC package with the inclusion of a LAO resource. An Ensemble instance must follow this data model. However, it is not mandatory to use the LAO extension mechanism. This should only be used in

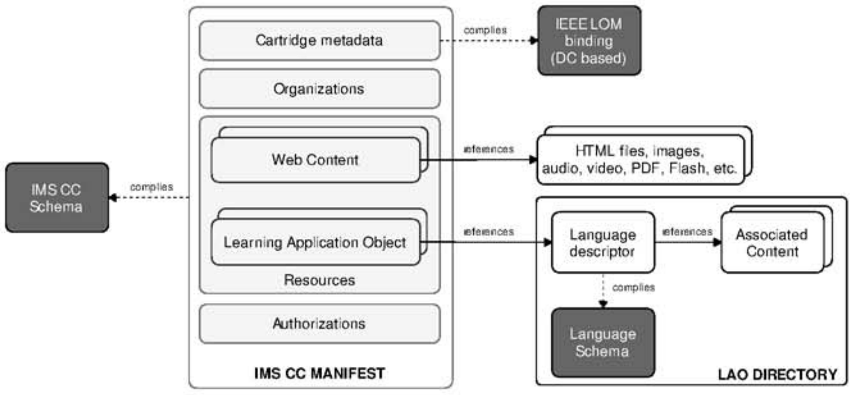

Figure 3: Structure of an IMS CC package.

cases where the specification is not sufficient for a specialized domain. This is the case of the computer programming domain where programming exercises should be described. Current specifications (e.g. LOM, QTI) are insufficient and cannot describe how an exercise should be compiled and executed, which test data should be used or how it should be graded.

3.3 Integration Model

The Ensemble specification also comprises an integration model. This model recommends specifications for the communication between the axial systems and the core/secondary services.

To illustrate the integration model of the framework we present a specialization of the framework for a specialized domain: the computer programming domain. The overall architecture of a network of e-learning systems and services participating in the automatic evaluation of programming exercises is depicted in Figure 4.

The network is composed by the following components: 1) Teaching Assistant (TA) to interface the users with the network and to mediate the communication among all components; 2) Integrated Development Environment (IDE) to code the exercises; 3) Assessment System (AS) to evaluate exercises of students; 4) Learning Objects Repository (LOR) to store and retrieve exercises; 5) Learning Management System (LMS) to present the exercises to students; 6) Conversion System (CS) to convert exercises formats; 7) Social Network Platform (SNP) to share comments about the exercises.

This section analyses the communication specifications for the interaction of axial systems with three core services typically found in e-learning environments:

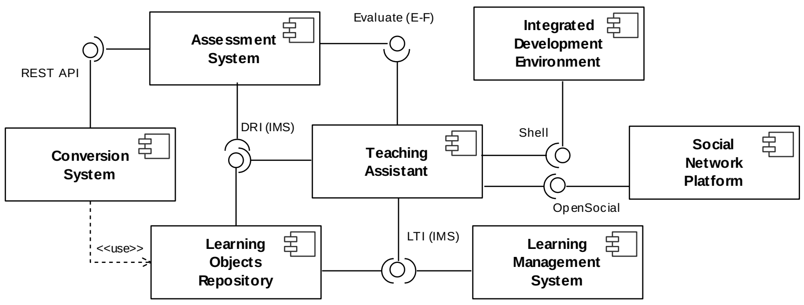

Figure 4: Network component diagram.

repositories, assessment systems, learning management systems and social network platforms. The recommendation of the EeF for the assessment is the **Text file evaluation specification** [Leal & Queirós, 2010b]. The selected specification for the communication with repositories is the **IMS DRI specification** and the LTI specification for the communication with the LMS. Finally, for interacting with social platforms the Ensemble specification recommends the **OpenSocial specification**. Other specifications may exist in an Ensemble instance even if not addressed by this model.

This integration model relies on web services for communication among systems. Web services can be used mostly in two flavours: SOAP and REST. SOAP web services are usually action oriented, mainly when used in Remote Procedure Call (RPC) mode and implemented by an off-the-shelf SOAP engine such as Axis. Web services based on the REST style are object (resource) oriented and implemented directly over the HTTP protocol mostly to put and get resources.

Both specifications have matured in distinct periods and they coexist nowadays. This explains why older specifications such as DRI recommends SOAP, while newest such as LTI are based on REST: SOAP started earlier and now the trend is REST as stated from a directory of 3200 web APIs listed at ProgrammableWeb[8] (May 2011).

Regardless of these trends, the EeF specification does not encourage the use of any flavour in the communication specifications detailed in the following subsections. As far as possible the EeF tries to keep an equidistant position from both flavours.

[8] Official web site: http://www.programmableweb.com/

3.3.1 Text File Evaluation Service Genre

Text file evaluation responds to the shortcomings of the assessment based on questions with predefined answers. Questions with predefined answers are formalized in languages such as IMS QTI and supported by many e-learning systems. Complex evaluation domains justify the development of specialized evaluators that participate in several business processes as autonomous services. The definition of this specification was inspired by the service definition model of the E-Framework[9]. The specification consists of an abstract service type (service genre) that describes a text file evaluation service. 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. The service modelled by the proposed definition receives a text file with an attempt to solve an exercise and produces an evaluation report. The exercise is referenced as a LO available on an interoperable repository. The abstract service supports three functions:

- The `ListCapabilities` function lists all the capabilities supported by a specific evaluator;
- The `EvaluateSubmission` function evaluates a submission to a given exercise, using an available capability;
- The `GetReport` function provides a detailed report of a previous evaluation.

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 features of compilers or interpreters. Each capability has a number of features to describe it and for a programming language they may be the name of the language (e.g. Java), its version (e.g. 1.7) 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. In this function, the request does not 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 format of this listing is outside of the scope of this specification and must be defined by the concrete service definition.

The `EvaluateSubmission` function requests the evaluation of an attempt to solve 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 text file with a single attempt to solve that particular exercise. The request may include a specific evaluator capability necessary for the proper evaluation of that attempt. The response returns either a ticket for a later report request or a report on the

[9] Official web site of E-Framework for Education and Research - http://www.e-framework.org

evaluation. In any event the response will include a ticket to recover the report on a later date. The service endpoint provides the interfaces for the requests and responses for the evaluation functionality. Internally the service implementation may include several features (indexing, queueing, 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. The evaluator returns a report on the evaluation, if it is completed within a predefined time frame. The report should contain detailed information on the assessment rather than a verdict such as passed or failed. The format of the report sent to the client should be designed for using it as input to other systems (e.g. classification systems, feedback systems) and may be, for instance, transformed in the client side based on a XML stylesheet. The specification of the response format is outside the scope of this specification and must be defined by the concrete service definition. Requesting a report using a ticket is supported through another function called `GetReport` detailed in the next sub-subsection.

The `GetReport` function allows a requester to get a report for a specific evaluation. 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 an evaluation report.

3.3.2 Digital Repositories Interoperability

The IMS Digital Repositories v1.0 specification[10], released in 2003, aims to provide recommendations for the interoperation of the most common repository functions. In order to use the IMS DRI these recommendations should be implementable across systems to enable them to present a common interface for those functions. These core functions are:

- The `Submit` function defines how an object is moved to a repository from a given network-accessible location and how the object will then be represented in that repository for access. The recommended communication protocol is SOAP with attachments[11] with the attachments taking the form of one or more IMS-compliant Content Packages;
- The `Search` function defines the searching of the meta-data associated with the content exposed by repositories. Two protocols are suggested: XQuery over SOAP (for learning object repositories) and Z39.50 (for libraries). Searching is performed using the XQuery protocol over XML meta-data, adhering to the IMS Meta-Data Schema;

[10] http://www.imsglobal.org/digitalrepositories/
[11] http://www.w3.org/TR/SOAP-attachments

– The `Request` function allows a client that has located a meta-data record via the `Search` function to request access to the LO described by that meta-data;

– The `Alert` function defines an intermediary aggregation service and envisions that e-mail/SMTP (Simple Mail Transfer Protocol) could provide this functionality.

The DRI specification also includes a messaging model. This model standardizes general communication between the components defined by the DRI architecture. The basic message has two parts: a message header and a message body. XML bindings are also provided by this specification but further extensions are needed to enhance the communication between the repository and other systems [Rodriguez *et al.* , 2006, Eap *et al.* , 2004].

3.3.3 Learning Tools Interoperability

IMS developed the Learning Tools Interoperability v.1.0 (LTI) specification in 2010. This recent specification provides a standard way of integrating rich learning applications - in LTI called Tool Providers (TP) - with platforms like LMSs, portals, or other systems from which applications can be launched - called Tool Consumers (TC). LTI v1.0 defines a formal deployment process whereby the LMS and the application reach an agreement on the run-time services and security policies. In order to accelerate the conformance to this new specification by Tool Consumers the IMS launched also a subset of the full LTI v1.0 specification called IMS Basic LTI. This subset exposes a single (but limited) link between the LMS and the application. In March 2012 IMS launched the LTI v1.1 (final version) merging both specifications (Basic LTI and LTI). This new version includes the support for an outcomes service based on a subset of the IMS Learning Information Services (LIS)[12] - the LTI Basic Outcomes Service. The LIS specification is the definition of how systems manage the exchange of information that describes people, groups, memberships, courses and outcomes within the context of learning. Figure 5 shows how the bi-directionality of the LTI specification is performed.

TC provides launch data with LIS pointers to the TP. It is not required for the TC to provide these services. The LIS services could even be provided by a third system such as a Student Information System. Then, the TP calls the LTI Basic Outcomes Service if available. This service receives "Plain Old XML" (POX) messages signed using OAuth. The service supports setting, retrieving and deleting LIS results associated with a particular user/resource combination. The following functions are supported:

[12] Official Web site: http://www.imsglobal.org/lis/

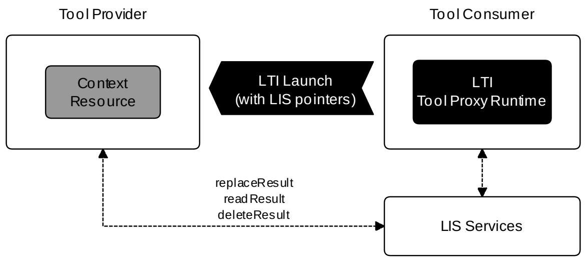

Figure 5: The IMS LTI Specification v.1.1 - integration with LIS services.

- The `replaceResultRequest` function sets a numeric grade (0.0 - 1.0) for a particular result;
- The `readResultRequest` function returns the current grade for a particular result;
- The `deleteResultRequest` function deletes the grade for a particular result.

3.3.4 OpenSocial APIs

OpenSocial is a set of three programming interfaces (APIs) for web-based social network applications created by Google. The OpenSocial API covers a broad range of capabilities such as:

- Profile information (user data);
- Relationships information (social graph);
- Activities (e.g. news feed).

The main goal of OpenSocial is to provide a common framework to be used by developers to guarantee interoperability across several social networks on the Internet, which act as containers for each OpenSocial-compliant application.

This specification provides a REST and RPC API communication flavours through which OpenSocial-compliant applications and containers interact with each other, transmitting user data, friend lists and activities. These protocols support various data exchange formats such as XML, JSON and ATOM. In order to authorise access to data stored in social networks, these APIs rely on the OAuth specification.

The lack of adoption by major players such as Facebook affects negatively the OpenSocial adoption. In order to get around with this issue other alternatives can be used. A well known approach is to build API wrappers that map the

OpenSocial API to the native APIs provided by the various SNPs. This is the approach taken, for instance, by the JokerD project.

Regarding the Ensemble framework and the specialization on the computer programming domain, the goal is to take benefits from social media platforms to the teaching-learning process. One good example is to gather the exercises comments about programming exercises from students into a public and common place. These reviews can help peers to accelerate the beginning of solving an exercise or even to decide which exercise first start. Also using the OpenSocial API will be possible to adapt the most suitable exercises based on the learner's social profile.

3.4 Evaluation Model

In order to provide meaningful metadata to the evaluation engines, an exercise definition must have an unambiguous evaluation model. Otherwise, authors could create exercises that risked to be evaluated differently from want they intended. After considering several alternatives a single and simple four steps evaluation model was selected based on the comparison of the output and side effects of the students' code with those of a standard solution. This model is depicted in Figure 6 and enumerated bellow. Another approach is to compare a set of programs from different learners and evaluate them competitively.

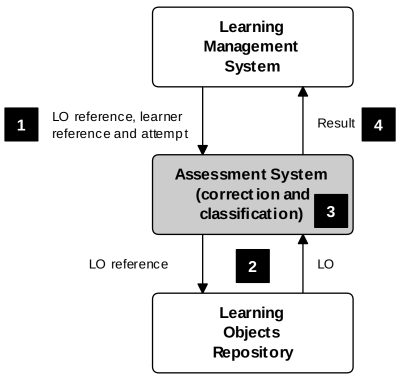
Learning Management System
1
LO reference, learner reference and attempt
Result
4
Assessment System (correction and classification)
3
LO reference
2
LO
Learning Objects Repository

Figure 6: Evaluation model.

1. The evaluator receives three pieces of data: a reference to the LO with an exercise; an attempt to solve it - a single file, a program or an archive containing files of different types (e.g. ZIP, JAR); and a reference to the learner submitting the attempt;

2. The evaluator loads the LO from a repository using the reference and uses the assets available in the LO (static tests, generated tests, unit tests, solution files, etc.) according to their role.

3. The evaluator produces an evaluation report with a classification and possibly also with a correction and feedback. The feedback that may depend on the learner's reference and may be stored for future incremental feedback to the same learner;

4. The evaluator returns the evaluation report immediately or makes it available within a short delay.

4 Validation

The Ensemble framework includes an abstract data and integration model that must base the implementation of networks in specialized domains with complex evaluations. In this section we present the specialization of Ensemble for two domains: computer programming and computer-aided design. For each domain we describe the specialization of two Ensemble hotspots as described in Table 2.

Table 2: Hotspots of the Ensemble framework.

HotSpots	Computer Programming	Computer-Aided Design
Data	CC with LAO	CC
Evaluation	test cases evaluation	primitive elements matching

4.1 An Ensemble instance for the computer programming domain

In this Ensemble instance the **data** model based on the IMS CC specification is extended for describing programming exercises. This extension is achieved by adding a new LAO resource in the CC package as recommended by the Ensemble specification. The LAO resource is formally described as an XML language called PExIL (Programming Exercises Interoperability Language) [Queirs & Leal, 2011]. The aim of PExIL is to consolidate all the data required in the programming exercise life-cycle, from its creation to when its solving and grading.

The automatic **evaluation** of programming exercises follows a test case approach. Student programs are tested against a set of test cases provided by the teacher as outlined in algorithm 1. The revaluation procedure receives as input the student program and a programming exercise, and produces as output a status and a feedback. A status is a small label such as "Accepted", "Wrong answer", "Execution error", "Time limit exceeded", or "Compilation error". The feedback is a longer text that may be presented to the student to help him/her to overcome the detected problem. The evaluation procedure is composed of two phases: static and dynamic evaluation.

Algorithm 1: Test case evaluation procedure

```
Input   : solution,attempt
Output: status,feedback
compilation ← compile(solution)
if compilation.error = ∅ then
    feedback ← ∅
    severity ← 0
    for testCase ∈ testCaseSet do
        execution ← execute(executable, testCase.input)
        if execution.status = "Accepted" then
            if differences(testCase.output, execution.output) = ∅ then
                stat ←"Accepted"
            else
                normal ← normalize(execution.output)
                if differences(testCase.output, normal) == ∅ then
                    stat ←"Presentation Error"
                else
                    stat ←"Wrong Answer"
                feed ← testCase.feedback
        else
            stat ← execution.status
            feed ← execution.error
        if severityOf(stat) > severity then
            status ← stat
            severity ← severityOf(stat)
            feedback ← feed
else
    status ←"Compilation Error"
    feedback ← compilation.error
```

The static evaluation is responsible for checking the syntax of the submitted program. If static evaluation fails them the evaluation procedure is aborted with a "Compile time error", given as possible feedback the compilation error. For compiled languages, such as C/C++ or Java, the static analysis coincides with a compilation that produces and executable required for the next phase. For interpreted languages this phase may be skipped of replaced by a preprocessing of the program to check syntactic validity.

The dynamic evaluation is the heart of the test based approach. Each test case has two data files: an input and an output. The program is executed with each test case and the obtained output compared with the expected output of the test case, if an output is produced. The execution may result in a number of errors, such as time limit exceeded. If none of these errors was produced then both outputs are compared. In this case the result may be "Wrong Answer", or "Presentation Error" if the differences are just in formatting characters, such as white characters, or "Accepted" if no differences are detected.

The result of executions of all test cases must be synthesizing in single result. For this one has to assign a severity with each status. The "Accepted" status has severity 0 and the rest of them in order ("Presentation error" , "Wrong Answer", "Time limit Exceeded", "Execution Error", "Compilation Error", etc) have increasing severity. The final status of the submission is the highest severity status of all tests.

4.2 An Ensemble instance for the CAD domain

This subsection presents the specialization of the Ensemble framework for the computer-aided design domain. Typically, a CAD exercise is composed by a description of the exercise represented with text and/or an image (Figure 7) and a solution (e.g. DWG, DSF, SVG).

These assets can be described and aggregated in a IMS CC package without the need for a LAO extension.

The automatic evaluation of computer-aided design (CAD) follows a graphic primitive matching approach. Student designs are checked against a set of possible solutions provided by the teacher as outlined in algorithm 2. In this case is the actual design submitted by the student that is compared with a solution, rather than its side-effects as happens with the test case based approach described for programming exercises. This evaluation procedure receives as input a CAD document and a CAD exercise, and returns a status – either "Accepted" or "Wrong answer" – and in the last case a feedback. The evaluation procedure is also composed of two phases: normalization and matching.

The purpose of normalization phase is to convert every design into a common and normalized format. The designs may be entered in multiple formats (DXF, SVG, DWG, etc) and are converted to an internal representation. Also, the

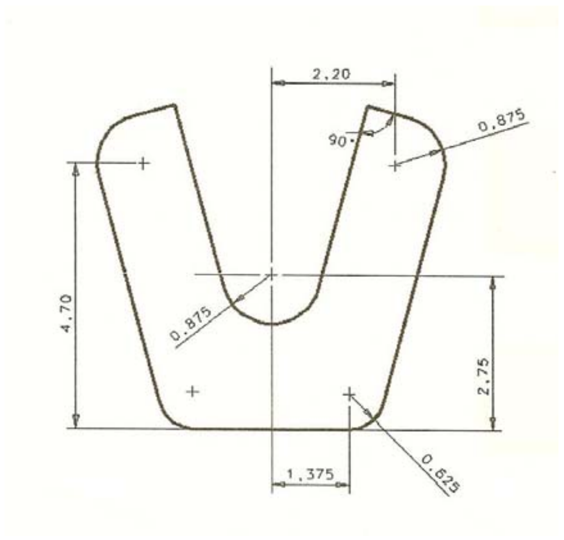

Figure 7: A CAD drawing.

design elements may be different although equivalent. For instance, a set of line segments may be equivalent to a single polygonal line or a rectangle to two pairs of parallel line segments. Drawings are thus converted to a set of basic primitives – line segments and arcs – and this normalized internal representation is the basis for the matching phase.

The matching phase is the core of the primitive matching evaluation. In this evaluation approach the normalized design submitted by a student is matched against one of more designs provided by the teacher. To attempt to match two normalized designs the evaluator iterates over the primitives of one of them (e.g., the student's attempt) and tries to find a corresponding primitive in the other (e.g. a teacher's possible solution). The primitives must have the same type (line or arc segment) and similar properties (segment length or arc angle) within a certain error margin. Also, it must have similar relations with previously matched elements. For instance, if segment s_1 matched with segment s'_1 and segment s_2 has a common end point with s_1 and a $90\,^\circ$ angle between them, then the segment s'_2 matches s_2 if it has also a common end point with s'_1 and a similar angle. The best possible matching may still lack primitives or have them in excess. Even when a match is found for every primitive, the properties of matching primitives must also coincide. For instance, a solid line segment in the submitted design will produce an error if the matching segmented is a dotted line in the teacher solution. If the least number of errors is zero then the attempt is "Accepted". Otherwise it reports "Wrong answer" and returns the list of errors as feedback.

Algorithm 2: Matching primitives evaluation procedure

```
Input  : solutionSet,attempt
Output: status,feedback

for solution $\in$ solutionSet do
    normalizedSet $\ni$ normalize(convert(solution))

feedback $\leftarrow \emptyset$
errorCount $\leftarrow \infty$
for solution $\in$ normalizedSet do
    matchSet $\leftarrow \emptyset$
    errors $\leftarrow \emptyset$
    for primitive$_1$ $\in$ solution do
        for primitive$_2$ $\in$ attempt do
            if $match$(primitive$_1$, primitive$_2$, matchSet) then
                matchSet $\ni$ (primitive$_1$, primitive$_2$)
                errors $\ni$ differences(primitive$_1$, primitive$_2$)

    if $\sharp$matchSet = $\sharp$solution then
        if $\sharp$errors < errorCount then
            feedback $\leftarrow$ errors
            errorCount $\leftarrow$ $\sharp$feedback

    else
        feedback $\leftarrow$ {"No match"}

if feedback = $\emptyset$ then
    status $\leftarrow$"Accepted"
else
    status $\leftarrow$"Wrong Answer"
```

5 Conclusions

This paper presents the Ensemble framework as a conceptual tool to organize a network of e-learning systems and services based on content and communications specifications. The framework includes a data, integration and evaluation models that should base the implementation of networks in specialized domains with complex evaluations. For validation purposes we specialized the framework for two domains: computer programming and computer-aided design. This specialization is presented based on two Ensemble hotspots: data and evaluations models. In the former we formally describe the exercise and present possible extensions. In the latter, we describe the evaluation procedures. It should be noted that only the first specialization is implemented and is being used at ESEIG - a school of the Polytechnic of Porto.

The main challenge resulting from this research is to apply the framework to other domains. Based on the validation process we conclude that most of the framework components can be shared for several domains.

Other interesting domain is serious games applied to management courses where students develop their skills using simulation. Business simulation games improve the strategic thinking and decision making skills of students in several areas (e.g. finances, logistics, and production). Through these simulations students compete among them as they would in a real world companies. A business simulation service fulfils a role similar to that of the assessment systems in programming exercises and it also requires a repository containing specialized LO describing simulations. Thus, this specific domain poses challenges not only in the development of the network TA , but also in the refinement of the framework specifications and services (e.g. repository, assessment system) to meet the new evaluation domains requirements.

As future work we intend to work in the two presented specializations. Regarding the Ensemble instance, in the field of computer programming, the idea is to distribute it as a Massive Open Online Course (MOOC). This will require recasting some services in the Ensemble framework such as the Teaching Assistant and the Assessment System while creating new ones such as a Sequencing tool. The combination of these type of tools will foster the "practice based learning" where learners browse in an instruction two types of digital resources: expositive and evaluative. The former are typically short videos with workout examples on solving programming exercises. The latter are programming exercises that learners should solve on their favourite IDE guided by the pivot component of Ensemble, the Teaching Assistant. Both type of resources are stored in a cloud environment to facilitate access and deployment. Regarding the computer-aided design instance the goal is to implement the instance. This will require the creation from scratch of an assessment system based on a graphic primitive matching approach previously described.

References

[Chafle *et al.* , 2004] Chafle, Girish B., Chandra, Sunil, Mann, Vijay, & Nanda, Mangala Gowri. 2004. Decentralized orchestration of composite web services. *Pages 134–143 of: Proceedings of the 13th international World Wide Web conference on Alternate track papers & posters.* WWW Alt. '04. New York, NY, USA: ACM.

[Curbera *et al.* , 2002] Curbera, F., Duftler, M., Khalaf, R., Nagy, W., Mukhi, N., & Weerawarana, S. 2002. Unraveling the Web services web: an introduction to SOAP, WSDL, and UDDI. *Internet Computing, IEEE*, **6**(2), 86–93. http://dx.doi.org/10.1109/4236.991449.

[Eap *et al.* , 2004] Eap, Ty Mey, Hatala, Marek, & Richards, Griff. 2004. Digital repository interoperability: design, implementation and deployment of the ecl protocol and connecting middleware. *Pages 376–377 of: Proceedings of the 13th international World Wide Web conference on Alternate track papers & posters.* WWW Alt. '04. New York, NY, USA: ACM.

[Girardi, n.d.] Girardi, R. *Framework para coordenao e mediao de Web Services modelados como Learning Objects para ambientes de aprendizado na Web.* M.Phil. thesis, Universidade Catlica do Rio de Janeiro - PUCRio, Rio de Janeiro.

[Khan, 1997] Khan, B. H. 1997. Elearning framework dimensions and subdimensions. *Web-Based Instruction Journal.* `http://asianvu.com/bookstoread/framework/elearning_framework_flyer.pdf`.

[Kurilovas, 2012] Kurilovas, E. 2012. *European Learning Resource Exchange: A Platform for Collaboration of Researchers, Policy Makers, Practitioners, and Publishers to Share Digital Learning Resources and New e-Learning Practices.* IGI-Global.

[Leal & Queirós, 2009] Leal, José Paulo, & Queirós, Ricardo. 2009. Defining Programming problems as Learning Objects. *Pages 1033–1040 of: ICCEIT - World Academy of Science, Engineering and Technology*, vol. 58.

[Leal & Queirós, 2010a] Leal, José Paulo, & Queirós, Ricardo. 2010a (March). eLearning Frameworks: a survey. *In: International Technology, Education and Development Conference, Valencia, Spain.*

[Leal & Queirós, 2010b] Leal, José Paulo, & Queirós, Ricardo. 2010b. Modelling text file evaluation processes. *In: 2010 International Workshop on Cognitive-based Interactive Computing and Web Wisdom (CICW'10).* LNCS. Shanghai, China: Springer-Verlag, for Springer-Verlag.

[Markiewicz & de Lucena, 2001] Markiewicz, Marcus Eduardo, & de Lucena, Carlos J. P. 2001. Object oriented framework development. *Crossroads*, **7**(July), 3–9. `http://doi.acm.org/10.1145/372765.372771`.

[Queirs & Leal, 2011] Queirs, Ricardo, & Leal, Jos Paulo. 2011. PExIL: Programming Exercises Interoperability Language. Conferência - XML: Aplicações e Tecnologias Associadas (XATA).

[Rodriguez *et al.* , 2006] Rodriguez, E, Sicilia, M A, & Arroyo, Sinuhe. 2006. Bridging the semantic gap in standards-based learning object repositories. *Pages 478–483 of: Proceedings of the Workshop on Learning Object Repositories as Digital Libraries Current challenges.*

[Wilson *et al.* , 2004] Wilson, Scott, Blinco, Kerry, & Rehak, Daniel. 2004 (July). *Service-Oriented Frameworks: Modelling the infrastructure for the next generation of e-Learning Systems.* Tech. rept. JISC Report. `http://www.jisc.ac.uk/uploaded_documents/AltilabServiceOrientedFrameworks.pdf`.