SOA-based Architecture for a Generic and Flexible E-assessment System

Mohammad AL-Smadi
Institute for Information Systems and computer Media
Graz University of Technology
Graz, Austria
msmadi@iicm.edu

Christian Gütl
Graz University of Technology, Austria,
Curtin University of Technology, Perth, WA.
Graz, Austria
cguetl@iicm.edu

Abstract— **In the last decade, universities and higher education institutes have become more and more interested in using computers to deliver their formative and summative assessments. Therefore, several computer-assisted-assessment systems have been developed. The variance in the application domains of e-assessment has a main influence on having different assessment systems in the same university. Since universities have different colleges and specializations based on their types and in order to deliver their assessment activities online, each college is developing or buying assessment system or tools based on its specializations and courses. This has caused some universities to have more than one computer-assisted-assessment system. In this paper, a service-oriented e-assessment system will be suggested to solve this problem. A service-oriented architecture for a generic and flexible assessment system will be provided with cross-domain use cases to show the flexibility of this architecture.**

Keywords- E-assessment, Modular assessment system, Assessment services, Service-oriented architecture for assessment, Middleware for tools interoperability.

I. INTRODUCTION

The breadth of the assessment field and it's richness of several application domains raise problems when it comes to develop tools to assess those application domains. Numerous commercial and academic assessment systems and tools are used to assess different application domains. The level of coordination between those systems and tools highly depends on the standards and specifications underpin them as well as the domain related activities. In e-assessment design and development, the new developed tools should have the ability to communicate and interact with the existing ones without further modifications. Such tools should not be tightly coupled to a specific assessment system. Rather than, they should be reusable and interoperable so that they can be used to extend other systems and tools.

In the world of Service-oriented Architectures (SOA), the new developed services hold great promise when they are interoperable and fit with existing tools' services. One promising area is to modularize e-assessment systems in a way of composing loosely coupled assessment tools together. Such composition requires: i) clear guidance represented by a well-formed *framework*; ii) *standards and specifications* that represents the whole process of assessment as well as the communication between the services and components; iii) *cross-domain requirements* analysis in order to define the specific requirements for each application domain (such as, educational editor in the mathematic domain); iv) *web services* that provide the cross-domain requirements and interact through well-defined interfaces.

The rest of this paper is organized as follows: Section 2 explores the different aspects for understanding the e-assessment domain. Section 3 suggests a Modular Assessment System for Modern Learning Settings (MASS) towards a generic and flexible assessment system. A SOA-based architecture for MASS is discussed in Section 4. Section 5 discusses how this SOA-based architecture can be mapped into system components as part of more detailed architecture. Problems and challenges for aggregating domain-based tools and services within MASS context are discussed as part of a suggested Middleware in section 6. Section 7 investigates the MASS suggested architecture using a persona-based use case. A conclusion and future work are provided in Section 8.

II. E-ASSESSMENT DOMAIN

Challenges that may face e-assessment development varies according to the level of assessment domain understanding. Bloom's described the assessment process based on the learning outcomes represented by levels of learning objectives [1]. Authors of [2] classified the learning assessment domain into four assessment types *summative, formative, diagnostic, and self-assessment.* Some others believe that there is a need to develop models for exploring and defining this domain [3]. For Instance, the Joint Information Systems Committee (JISC) has initiated a service-oriented based framework for e-learning called "e-Framework" [4]. Based on this framework JISC has presented an e-Framework Reference Model for Assessment (FREMA). FREMA represents an intensive guide for the assessment domain resources *standards, projects, people, organizations, software, services, and use cases.* FREMA structure is based on concept maps describing the ontology that has been used to model the assessment domain. [5].

Other researches described the assessment domain from different perspectives. As in [6] the requirements for what they called the 'Ultimate' assessment engine have been discussed. From this perspective, the assessment domain has

been investigated as the possible requirements for the possible stakeholders and roles of the assessment system.

A different way of domain understanding has been discussed in [7]. In this research the authors investigated the e-assessment domain as a set of possible services. A Service-Oriented Framework for Assessment (SOFA) has been initiated as a set of layers: *User Agents, Assessment services, Common Services, and infrastructure resources*. In presenting this framework they provided people in the domain of e-assessment with a guide to understand the interactions between the domain users and the available resources as a set of services. They also stressed on the importance of designing those services to be standard-conform to avoid having a repository of useless services. Rather than, those services will be interoperable and useful for designing flexible assessment systems.

III. MODULAR ASSESSMENT SYSTEM FOR MODERN LEARNING SETTINGS (MASS)

Based on the highly demand for having a generic and flexible e-assessment system, MASS has been suggested. MASS is supposed to have: (a) *flexible design* to be used as a stand-alone system or to be easily extended by third-party tools. (b) *User-friendly interfaces* for both students and educators where a user interaction and online submission of solution and evaluation can be done. (c) Assessment environment for *various learning and assessment settings* which supports guided as well as self-directed learning. (d) *Management and (semi-)automatic support* over the entire assessment lifecycle (exercises creation, storage and compilation for assessments, as well as assessment performance, grading and feedback provision). (e) *Rubrics design* and implementation interfaces to allow the educators to design their own rubrics based on learning objectives to assess learners' performance against a set of criteria. (f) *Support of various educational objectives and subjects* by using various tools sets which for example enables automatic exercise generation or selection, automatic grading and feedback provision. (g) *Results analysis and feedback provision* (immediately or timely) of the current state of user knowledge and meta-cognitive skills for both educators and learners and also for adapting course activities and learning contents based on users' models. (h) *Standard-conform information and services* to be easily sharable, reusable and interoperable. This may include the tests' questions, answers and students' results, rather than any other required services. And finally, (i) *Security and privacy* where a secure logon of users based on pre-defined levels of access, and also users' authentication based on machine (domain users) or by usernames/passwords.

MASS is planned to be a SOA-based assessment system where different assessment tools can integrate with each other to assess different application domains. MASS highly depends on interoperable web services that can extend MASS's assessment native services (Authoring, Scheduling, Delivering, Scoring, and Reporting) to assess specialized kinds of assessment (e.g. Algebra assessment and Programming assessment) based on several application domains. Fig. 1, shows how cross-domain web services can be used to extend the core services provided by MASS. A Middleware layer has been added between the Application layer and the Application Domains one. This Middleware will be designed to handle the use of domain-based web services to extend MASSs' ones for specialized application domains. Moreover, it will be used to facilitate and tackle the problems and challenges of running external domain-based tools within the context of MASS.

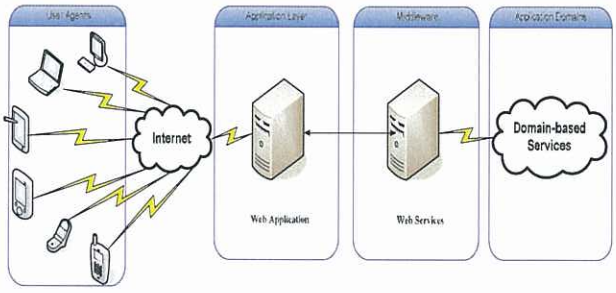

Figure 1. Cross-domain web services to extend MASS native services

IV. SOA-BASED ARCHITECTURE

As Shown in Fig. 2 a SOA-based architecture for MASS has been presented. This architecture consists of four main tiers: Client Application Tier, Business Tier, Service Tier, and Resource Tier.

A. *Client Application Tier*

The client application tier forms the front tier where the user agents interact with the system. Matters such as security and privacy of data and processes are handled in this tier. Furthermore, unauthorized user agents are prevented through this tier from accessing the system resources.

The **controller** of each client application must be capable of controlling the whole actions in this tier. The application processes and requests are passed by this controller to a special **interaction services** layer that communicate with the business tier using communication standards and technologies (such as Simple Object Access Protocol (SOAP), or XML-Remote Procedure Call (RPC)).

B. *Business Tier*

The whole business logic is taken place in this tier. In our case, the main MASS modules (Authoring, scheduling, delivering, and reporting) are part of this tier. For each module a set of core services are available in the service tier. Separating the business logic in a different tier fosters MASS to be more modular and flexible. Furthermore, this tier facilitates the update of the business logic in an easy and flexible manner. As in the client tier, this tier also has a controller and an interaction services to communicate with the service one.

C. *Service Tier*

In the world of SOA there are three main roles of interaction: *service provider, service registry, and service requestor.* Service providers are software agents that provide the service. They should **publish** a service description on a services registry. Service clients are software agents that **request** the execution of services. They should be able to **find** services descriptions on the services registry. The words in bold with the operation **bind** form the main operations in the SOA. During the bind operation the service requestor invokes a web service at run-time using the binding information in the requested service description to locate this service. This invocation has two main possible scenarios: the first one is direct invocation by the service requestor using the technical information in the service description located on the services registry. The second one is via a service discovery agency where the communication between the service requestor and the service provider goes through the services registry of the discovery agency. [8]. In the case of MASS the first scenario will take place where the invocation of the domain web services will be done directly through the services registry in the service tier.

In the server side there are two main operations:

- Service Registration: each service provider from the resources tier registers its web services through the **service provider interface** to the **service registry**. The service providers are normally domain related systems (such as algebra assessment system, programming assignment assessment tool…) that provide their business logic services (such as algebra marking, equation editor) as web services.
- Service Invocation: this operation is mainly handled by the **service request layer** where the requested service from the client application through the business tier modules is searched in the **core services** repository. If the service is not available in the core services repository then it will be searched in the new registered services in the **services registry.**

By using these two operations the requests from the client applications are answered. Once the business tier module (e.g. Authoring) uses a new registered web service from the **service registry** in the service tier, the service becomes available through that module **core services**. So the next time such web service is needed the module can directly use it from its **core services**.

D. *Resource Tier*

The resource tier contains the MASS infrastructure resources (such as MASS databases) as well as the domain related systems and tools. The domain related systems can be assessment related such as algebra assessment system or can be business related such as the Student Information System (SIS). For each of these systems and resources there are a set of web services they must provide. The more web services they provide the more flexible and generic MASS will be.

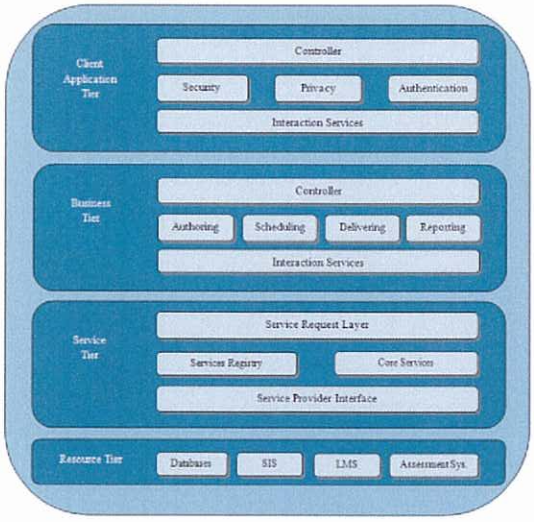

Figure 2. SOA-based Architecture for MASS

V. DETAILED ARCHETICTURE

The architecture presented in Fig. 2 is more conceptual than showing the real components of MASS. This section discusses a more detailed architecture where it explains how the SOA-based architecture layers are presented as system components and modules as well as it suggests architecture for the middleware layer. Moreover, it shows the assessment process represented by a set of core services for assessment and guided by the SOFA layers [7]. Some other common services such as authorization and authentication are also shown in this architecture. Fig. 3 demonstrates the more detailed architecture where it consists of four main layers: User Agents Layer, Application layer, Middleware Layer, and Application Domains Layer.

A. *User Agents Layer*

This layer represents the possible users of MASS. User agents can be native users (such as students, teachers, and administrators) or external users (such as related tools or LMSs). External users can be services requestor or provider. In case of a tool or a LMS is requesting a service from MASS, MASS should be flexible and interoperable to act like a service provider without rewriting code or system rebuilding. But if they providing services they will be part of the application domain layer and their services will be registered to the services registry in the middleware layer.

B. *Application Layer*

This layer represents the native components of MASS. All the assessment native services (Authoring, scheduling, delivering, analyzing or scoring, and reporting) and the common services (user management, authentication, authorization, security, and privacy) are parts of this layer. This layer can also be referred as the layer of MASS native

services. Based on SOFA [7], the assessment services layer and the common services layer are mapped into modules. For instance, the assessment core services in this layer are represented by a set of main modules (Authoring, Scheduling, Delivering, Analyzing, and Reporting). The assessment process is explained using these modules by defining the main tasks of each module. As shown in the architecture of this Layer the assessment cycle starts with authoring assessments and ends with reporting. Adapting of the assessment process to achieve the learning goals is part of the analysis module. The adaptation process can suggest changes in the assessment process which may go back to the authoring services. All the communications between the assessment services are done through the services buss.

C. *Middleware Layer*

For the sake of generality and flexibility this layer has been added to the architecture. One of the main goals of MASS is to be flexible in order to work as a stand-alone system or to integrate with other systems and tools. This middleware forms as a run-time platform where the domain-based services (e.g. Domain-based Editor) will be registered in order to be used by MASS modules.

The domain-based services will be used to extend the services provided by MASS. As well as they will be used to integrate between MASS and other application domain systems and tools. For each new domain-based service the service provider should publish the service description on the middleware services registry. This description will be used by the MASS modules to bind the service to any possible service requestor. Moreover, a flexible design will be suggested for this layer fostering MASS to run domain-based tools as part of its context.

D. *Application Domains Layer*

This layer represents the systems and tools that are used in specialized application domains such as Algebra assessment and Programming assessment. For those systems and tools it is supposed that they provide their services as web services. These web services will be used to extend the MASS core services which gives MASS the ability to assess different application domains. MASS should be flexible to run those tools within its context without the need to build them from scratch. Next section discusses this challenge and suggests possible solution for that.

VI. MASS MIDDLEWARE

In order to have a flexible and generic e-assessment system the following aspects should be considered:

- Standards-conformation
- Learning Objects (LO) Interoperability
- Tools Interoperability
- User Interface Interoperability
- Web services management

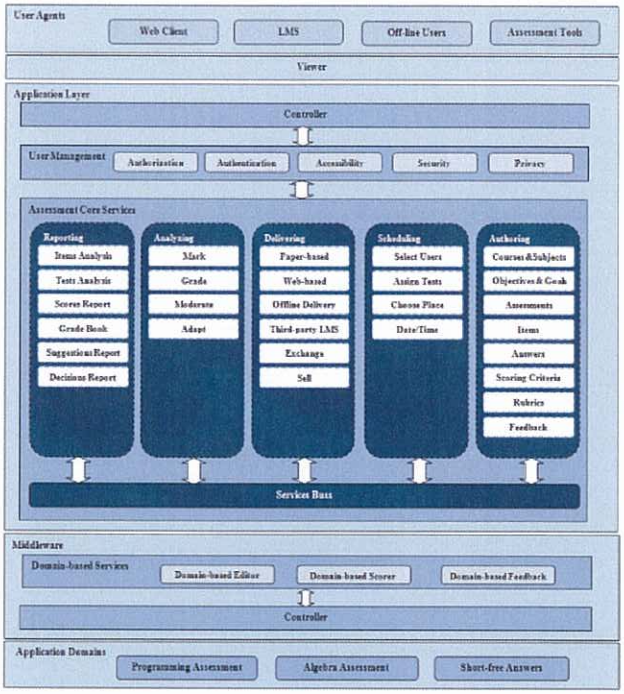

Figure 3. Detailed Architecture for MASS

In the context of e-assessment the term interoperability mainly refers to the ability of sharing Learning Objects (LO) between different assessment systems. For the sake of LOs interoperability, LOs should be standard-conform. Once they are standard-conform they can be easily shared between different standardized e-assessment systems [7]. Furthermore, these LOs can be re-used to construct new assessment activities without a need to build them from scratch. Some of the possible e-learning and e-assessment standards and specifications consortia are: The Instructional Management System Global Learning Consortium (IMS GLC) [9], Advanced Distributed Learning (ADL) [10], ADL Sharable Courseware Object Reference Model (SCORM) [11], The Aviation Industry CBT (Computer Based Training) Committee (AICC) [12], and IEEE Learning Technology Standardization Committee (IEEE LTSC) [13]. According to [14] almost all of these standards require the following aspects regarding LOs communication:

- *Launch:* the requirements for launching an LO in a web-based environment.
- *Application Programming Interface (API):* the interface of methods to be invoked by an LO to communicate with LMSs.
- *Data Model:* the dataset for the communication process.

According to [15] *interoperability* refers to "the ability of two or more software components to cooperate despite differences in languages, interface, and execution platform". Tools interoperability concerns the ability of aggregating

third-party tools to cooperate with a LMS platform. Third-party tools can be used to extend the services provided by the core system with services for specialized application domains. One of the possible specifications for tools interoperability is the work provided by the IMS GLC. IMS GLC has provided an architecture for tools interoperability as web services. IMS Tools Interoperability Guidelines v1.0 has described this architecture as well as its main components [16]. The suggested architecture has introduced two main concepts:

- *Proxy Tool:* from its name this tool will be used by the LMS to communicate with the external tools. A standard mechanism for packaging this tool to be deployed to an LMS has been defined by the architecture. The proxy tool is meant to be environment-independent where it does not require specialized code. The proxy tool is entirely a *descriptor-based package* that describes the deployment, configuration, and runtime context.
- *Tools Interoperability Runtime (TIR):* is a set of services that have to be implemented to the hosting environment (MASS in our case and the application domains tools). The TIR facilitates the deployment, configuration, and launching of the proxy tool. This deployment, configuration, launch, and outcome services are in somehow similar to the launch, API, and data model discussed earlier.

The communication between the TIR/Proxy Tools is handled by a *core protocol* defined for this purpose. The core protocol is based upon the IMS General Web Services (GWS) specifications v1.0 [17] which utilize XML with WSDL for defining services and SOAP for the base transport protocol.

Since the communication between the third-party tools and the core system is performed as web services some kind of web services management is required. Matters such as services provision, services registration, services invocation, and security and accessibility should be taken into consideration. Special standards and specifications can be used to represent these web services descriptions. For instance the IMS General Web Services (GWS) specifications can be used where the WSDL/XSD created files are designed to comply with Web Service Interoperability (WS-I) Consortium Base Profile v1.1 [18].

A. *Middleware Functionalities*

Having that MASS is standard-conform and its LOs are authored with regarding to e-learning and e-assessment standards (e.g. IMS Questions and Test Interoperability (IMS-QTI) specifications [19]), the challenge of aggregating tools within MASS services is still there. In fact, designing MASS core services with respect to standards and specifications will not fix the problem of integrating third-party tools within MASS context. For instance, some of the services provided by the application domains tools such as cross-word puzzles in the domain of Natural Languages teaching and assessment are not covered by the available standards and specifications (e.g. IMS QTI). Another example could be the requirement for domain-based assessment tools or services such as services for scoring special types of assessment items (e.g. algebraic questions). Moreover, the need for third-party tools and services is not limited to the assessment types, rather than it exceeds that to the integration services between two systems such as e-assessment system and Student Information System (SIS).

As shown in [20] SCORM has a lack in its interoperability definition. Addressing the interoperability of content packaging and the interoperability of the Run-Time Environment (RTE) is not enough to have flexible tools interoperability. The author stressed on two other important concepts for improving content interoperability: *Standardized user interface controls, and open-source of a standardized SCORM API implementation.* Launching the content in the UI with a lack of standards causes different behaviors based on the properties of the web browser such as *width, height of the browser, resizing, area of launching within the browser.* LMS developers do not follow the same programming strategies SCORM supposing they are.

In order to tackle the fore mentioned problems and challenges the middleware has been added to MASS architecture. The middleware layer will form as a platform on which aggregating runtime interoperable tools will take place. Furthermore, the registration of the domain-based web services will be handled by this middleware through well-formed interface to a service registry. The later on binding of these services to the service requestor (MASS User Agent) is also done by this middleware. Designing this middleware in a flexible way will foster MASS to be more and more flexible.

B. *Middleware Architecture*

Fig. 4 shows the main components of MASS middleware as well as the required services and communication patterns with the Application Domains Tools. This architecture is highly designed based on the architecture provided by the IMS Tools Interoperability Guidelines. Based on these guidelines the following steps will take place in the deployment of the Proxy Tool:

1. *Tool developer creates Proxy Tool deployment package. The deployment package is an archive contains a manifest and Proxy Tool's deployment descriptor.*
2. *MASS administrator deploys Proxy Tool deployment package to the MASS container.*
3. *MASS TIRs deployment service loads the Proxy Tool, thus creating a Proxy Tool Definition within MASS (during which validation of the deployment profile occurs, including a validation of profiles like security, outcome, user and delivery context asserted by the Proxy Tool deployment descriptor. MASS administrator can choose to not deploy the Proxy Tool if it does not support the required profiles).*
4. *MASS Administrator configures the Proxy Tool for use within an Institution by updating its definition appropriately with MASS data model.*

5. *An instructor or course designer utilizes the Proxy Tool definition to create a Proxy Tool Instance within a delivery context (course). The instance inherits the full base configuration and is additionally customized by the instructor/designer for launch from within the specific delivery context.*
6. *A learner in the course of a learning session is presented with the Proxy Tool instance (as a Learning Object or as part of one) and subsequently launches the Proxy Tool by selecting the URL provided by the MASS's user interface.*
7. *Tool's TIR validates and accepts launch in collaboration with the MASS's TIR which directs the user to the Tool's user interface, e.g., a redirect to a Tool specific URL.*
8. *Learner uses the Tool and in doing so potentially generates an outcome.*
9. *Tools' TIR sends the outcome to the MASS's TIR at the end of a learner's interaction with the Tool.*

A possible enhancement for this procedure could be the registering of the Tool's service after the end of step 9 in the services registry located in the MASS middleware. The Tool's Service description as well as the service endpoint represented by the service URL can be published to the service registry so that in future requests the service will be directly provided to the learner or to the instructor user interface.

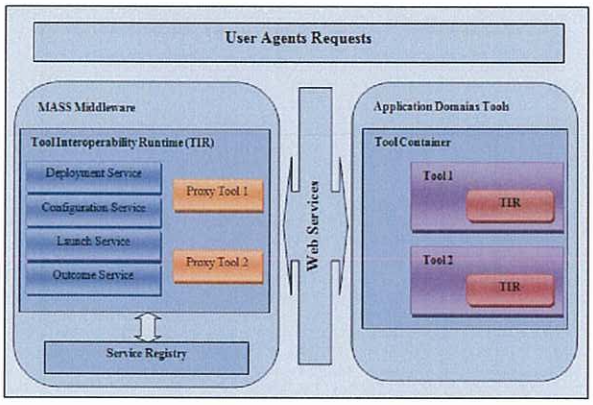

Figure 4. The cooperation between MASS modules and Application Domains' Tools through MASS Middleware

As shown in this architecture the Tool Interoperability Runtime services are:

- *Deployment Service*: The main function of this service is to interpret and load the Proxy Tool definition into the host TIR via its deployment descriptor. Thus this service is also expected to perform validation of the Proxy Tool settings in order to ensure correctness of and compatibility with the MASS's TIR.

- *Configuration Service*: The main function of this service is to manage the runtime settings of the Proxy Tool in order to provide the proper set of the same response for/during any given launch context.

- *Launch Service*: This service provides two main services, depending upon the context:

    - *Proxy Tool Host*: Performs all the functions related to launch of a Proxy Tool, including generating the relevant Proxy Tool launch message, utilizing the appropriate security profile, etc.

    - *Tool*: Exposed as a web service (Proxy Tool 1 and 2) that accepts launch messages from the MASS's TIR, understands the security profile used therein and responds back to the MASS's TIR using the Proxy Tool core protocol as to the status of the launch.

- *Outcome Service*: This service provides two main services, depending upon the context:

    - *Tool*: A web service client that generates outcome messages from the Tool's TIR conforming to a specific outcome profile type, for a given interaction of a user with the Proxy Tool/Tool, including utilizing the appropriate security profile.

    - *Proxy Tool Host*: Exposed as a web service that accepts outcome messages from the Tool's TIR, understands the security profile used therein and responds back to the Tool TIR using the Proxy Tool core protocol as to the status of the outcome processing.

VII. USE CASE

In order to investigate the eligibility of MASS suggested architecture, this section will discuss a use case of a programming course for beginners lecturer based on the following persona:

Persona, Jordi
Jordi is a lecturer at a department of computer science. One of his courses is "Programming for Beginners". For each semester a number of at least 500 students are attending this course from different departments and colleges. His department is using MASS to manage their online-assessments.

Scenario
"*I am using a programming assessment tool with simple functionalities of code-peer-review and feedback. This tool allows the annotation of code examples from students by tutors. Then the students have to annotate their pre-annotated examples. The students' annotations are compared with the tutors' ones and a valuable feedback is*

provided. I want this tool to be integrated with MASS in order to allow the only registered students in the course to upload their assignments to MASS and then those assignments will be used by the code-peer-review tool within the environment of MASS. "

Based on this persona and scenario the following procedure will take place to integrate the code-peer-review tool with MASS:

1. Jordi registers to MASS users as a lecturer which will allow him to create the course of "Programming for Beginners".
2. The students register as MASS users with the role students and request participation-confirm from Jordi to participate in the course.
3. After accepting students, Jordi creates a new assessment activity with type assignment using the Authoring module of MASS. As this assignment is using an external tool Jordi has to ask MASS administrator to integrate the tool with MASS resources.
4. Assuming that this tool has a Proxy Tool Deployment Package, the administrator deploys this package to the MASS container. MASS TIRs deployment service loads the Proxy Tool, thus creating a Proxy Tool Definition within MASS. If not the administrator has to ask the Tool developer/ Supplier to provide a Proxy Tool Deployment Package.
5. The administrator configures the Proxy Tool to fit with MASS data model and grants a privilege for Jordi over it. After that this tool will appear in the resources of the Authoring module used by Jordi to create the assignment activity.
6. Jordi utilizes the Proxy Tool definition from the Authoring Module resources to create a Proxy Tool Instance within the assignment (delivery context). The instance inherits the full base configuration and is additionally customized by Jordi using the Authoring module of MASS for launch from within the assignment.
7. Students who are participating in the assignment activity will launch the Proxy Tool by selecting the URL provided by the MASS's assignment user interface.
8. The code-peer-review tool's TIR validates and accepts launch in collaboration with the MASS's TIR which directs the student to the tool's user interface.
9. Students use the tool and in doing so potentially generate an outcome.
10. The code-peer-review tool's TIR sends the outcome to the MASS's TIR at the end of a student's interaction with the tool.
11. The outcome is passed to the MASS modules for further processing for instance to the Reporting module to provide feedback about the student-tool interaction.

VIII. CONCLUSIONS AND FUTURE WORK

According to the breadth and depth of the assessment application domains, several academic and commercial assessment tools and systems have developed. Some of these assessment tools and systems are limited to specific application domains. Moreover, they are not carefully designed to be standard-conform as well as they barely coordinate to other tools and systems in different application domains of assessment. This has caused universities and higher education institutes to have more than assessment tool. Managing several assessment tools are money and time consuming as well as requires extra resources. Tackling such problems and challenges by having a generic and flexible assessment system is a great challenge. Aggregating tools from different assessment application-domains in order to have a single flexible and generic assessment system requires: *Framework* for domain systems and tools development guidance; following *Standards and Specifications* that will help developers in producing their systems and tools; *Cross-domain requirements;* and *Web services* achieving those requirements.

A Modular Assessment system for Modern learning Settings (MASS) has been suggested as a generic and flexible e-assessment system. For the sake of flexibility and generality a SOA-based architecture for MASS has been designed. The design of this architecture is done with the guidance of the SOFA framework for assessment. The SOA-based architecture has been mapped into a more detailed architecture with a clear view of MASS components and modules. The detailed architecture has four main layers of: User Agents Layer, Application layer, Middleware Layer, and Application Domains Layer.

General and flexible assessment systems are capable to share their components with other systems. Therefore, MASS should be standard-conform in order to easily share its components. Following standards lacks a clear view of how to develop interoperable assessment tools. This lack has caused by having different standards and specifications (such as SCORM RTE, IMS TI) that have been published to support developers for developing interoperable assessment tools with weak definition for the interoperability process. For instance, in the case of SCORM RTE the standard explained how content packages conforming to SCORM standards can be shared between tools during their run-time but, it did not carefully explained launching this content in the user interface [20]. In order to facilitate tools interoperability during MASS runtime, the middleware layer has been suggested. The Middleware layer will form as a platform for tools runtime interoperability as well as for content sharing. Moreover, the middleware architecture is supposed to facilitate the domain-based web services registration to MASS. Those web services will be used to extend the capabilities of MASS native services assessing different specialized application domains.

For future work, the first prototype of MASS will be developed. A set of application-domains tools will be selected to extend MASS services. The first prototype side-by-side with the selected third-party tools will be used to evaluate the suggested architecture of MASS.

REFERENCES

[1] B.S. Bloom, Taxonomy of Educational Objectives. Longman, 1956.

[2] J. Bull, and C. McKenna, Blueprint for Computer Assisted Assessment. Routledge Falmer, 2004.

[3] G. Conole, and B. Warburton, "A review of computer-assisted assessment", ALT-J Research in Learning Technology, 13, 17-31, 2005.

[4] B. Olivier, T. Roberts, and K. Blinco, "The e-framework for education and research: an overview", 2005. retrieved from www.e-framework.org, last retrieved November 1st 2009.

[5] G. Wills, et al. "An e-learning framework for assessment (FREMA)". In Proceedings of 11th CAA Conference, 2007.

[6] N. Sclater, and K. Howie, "User requirements of the ''ultimate'' online assessment engine", Computers & Education, 40, 285–306, 2003.

[7] M. AL-Smadi, C. Guetl, D. Helic, "Towards a standardized e-assessment system: motivations, challenges and first findings", International Journal of Emerging Technologies in Learning (iJET), Volume 4, Special Issue 2: "IMCL2009" , 2009.

[8] M. P. Papazoglou, Web Services: Principles and Technology. Printice Hall, pp. 22-26, 2008.

[9] IMS GLC, "The IMS Global Learning Consortium", http://www.imsglobal.org/background.html, last retrieved November 7th 2009.

[10] ADL, "The Advanced Distributed Learning", http://www.adlnet.gov/about/index.aspx, last retrieved Octuber 7th 2009.

[11] SCORM, "ADL Sharable Courseware Object Reference Model", http://www.adlnet.gov/scorm/index.aspx, last retrieved Octuber 7th 2009.

[12] AICC, "The Aviation Industry CBT (Computer Based Learning) Committee", http://www.aicc.org/index.html, last retrieved Octuber 7th 2009.

[13] IEEE LTSC, "The IEEE Learning Technology Standards Committee", http://ieeeltsc.org/, last retrieved Octuber 7th 2009.

[14] G. Costagliola, F. Ferrucci, and V. Fuccella, "Scorm run-time environment as a service," in ICWE '06: Proceedings of the 6th international conference on Web engineering. New York, NY, USA: ACM, 2006, pp. 103-110.

[15] P. Wegner, Interactive software technology. in The Computer Science and Engineering Handbook, pp. 2440–2463, 1997.

[16] IMS TI, "The IMS Tools Interoperability guidelines", http://www.imsglobal.org/ti/index.html, last retrieved Octuber 7th 2009.

[17] IMS GWS, "The IMS general web services v1.0", http://www.imsglobal.org/gws/index.html, last retrieved Octuber 7th 2009.

[18] WS-I, "Web services interoperability basic profile v1.1", http://www.ws-i.org/, last retrieved Octuber 7th 2009.

[19] IMS QTI, "IMS Question & Test Interoperability Specification", Version 2.0 - Final Specification, http://www.imsglobal.org/question/index.html, last retrieved Octuber 7th 2009.

[20] J. Haag, "User interface (UI) interoperability for SCROM 2.0", LETSI, SCORM 2.o White Papers, 2008. http://wiki.letsi.org/display/nextscorm/SCORM+2.0+White+Papers, last retrieved November 1st 2009.