*International Journal of Engineering Education* Vol. 32, No. 2(B), pp. 1007–1014, 2016
Printed in Great Britain

0949-149X/91 $3.00+0.00


# IMS-LTI and Web-Services for Integrating Moodle to an Eclipse-Based Distributed Environment for Learning to Program*

FRANCISCO JURADO[1] and MIGUEL A. REDONDO[2]
[1] Dpto. Ingeniería Informática, Escuela Politécnica Superior, Universidad Autónoma de Madrid, Francisco Tomás y Valiente, 11, 28049 Madrid, Spain. E-mail: Francisco.Jurado@uam.es
[2] Dpto. Tecnologías y Sistemas de Información, Escuela Superior de Informática, Universidad de Castilla-La Mancha, Paseo de la Universidad, 4, 13071 Ciudad Real, Spain, Email: Miguel.Redondo@uclm.es

Learning Management Systems (LMS) offer generic services and tools in order to provide eLearning activities that cover a wide scope of teaching/learning methods. However they lack specific tools to support particular learning activities. Taking into account that there is not one unique tool that covers the entire learning/teaching process, an ideal eLearning scenario must use several specific tools for precise learning tasks. If we want to promote autonomous active learning by means of Personal Learning Environments (PLE) or extrapolate this feature to the new tendency of Massive Open Online Course (MOOC), this scenario is not only desirable but also necessary. Nevertheless, as the eLearning environment scales with the integration of more tools, we have to face communication and architectural issues. This paper exposes the solution we implemented by means of a centralized access point constituted by an LMS, an architecture based on Tuple Spaces, and the use of eLearning IMS-LTI specification to allow communication and information exchange among the different services and components of a system whose aim is to help students develop programming skills.

**Keywords:** IMS-LTI; moodle; eclipse; blackboard architecture

## 1. Introduction

eLearning scenarios focused on autonomous active learning like the Personal Learning Environment (PLE) or the current trend in the use of Massive Open Online Course (MOOC) [1] make the eLearning tools integration and information exchange not only desirable but also indispensable. The MOOC platform represents a central access point that manages courses. The students must make use of the necessary services in order to perform their autonomous learning by using both, the tools included in the MOOC platform as well as other applications external to it. However, it would seem that there are no solutions to perform the tools integration and information exchange in a standard way, forcing the implementation of a specific solution for specific courses.

On the one hand, when deploying extensive and institutional eLearning platforms, we can use applications like Learning Management Systems (LMS) to provide generic services and tools in order to offer activities that cover a wide scope of teaching/learning methods. However they lack specific tools to support particular learning activities. On the other hand, we can choose among several eLearning tools, each them providing specific features for specific domains. Nevertheless, a unique tool can not always be used to cover all necessities in an entire learning/teaching process. That is, the ideal scenario passes through the use of specific tools for specific tasks, but exchanging the information between each of them, so that one can benefit from the others. Thus, for example, a programming course in an LMS could take advantage of the use of self-assessment tools like the shown in [2–3] specifically designed for programming skills.

One possible solution is the use of standards. For some years, in order to create this synergetic effect among learning tools, there are working groups, such as IEEE LTSC (http://ltsc.ieee.org), IMS Global Learning Consortium (http://www.imsglobal.org) and ADL (http://www.adlnet.org), which aim to provide standardisation to allow interoperability and reuse in eLearning environments, by taking each eLearning system component as a service.

In this regard, we can spot several reference architectures for developing services-oriented eLearning systems, such as the so-called Services-Oriented Architectures (SOAs) like the IEEE Learning Technology Systems Architecture (LTSA), ADL (Advanced Distributed Learning Network) Shareable Content Object Reference Model (SCORM) [4], the OKI (Open Knowledge Initiative) Framework, the JISC (Joint Information System Committee) eLearning Framework and the IMS Abstract Framework [5]. Their goal is to arrange a set of tools

* Accepted 1 September 2015.

that supports the entire teaching/learning process by means of the integration of different components and services [6–8].

Some attempts that following eLearning standards to integrate videogame in LMS are [9–10], where we can see works that use the IMS Learning Tools Interoperability (IMS-LTI) [8] specification. However, the simplicity of IMS-LTI limits its use to those learning scenarios where, after the invocation of the external tool, both the LMS and the tool run as independently and few or none communication among them is required [10]. One step more is given by gateway4labs [11], an open source initiative to support the integration between remote laboratory management systems and LMS by using IMS-LTI. Nevertheless, the mutual point for all these approaches is they are limited to specific learning activities.

From this point of view, authors like Dagger et al. [12] have drawn attention to the use of service-oriented frameworks in order to support LMS composed by interoperable services for the next generation of eLearning platforms. This service-oriented perspective has led to approaches like the exposed by Alier et al. [13], who detail the case study for the Campus and Suma projects with an infrastructure based on Open Source Software and eLearning standards to allow the integration of external learning tools into Sakai and Moodle. The approach uses IMS-LTI and the Open Service Interface Definitions (OSIDs) developed by OKI, and conceives the LMS as the main and centralized piece for the teaching/learning process orchestration.

Likewise, Moodbile [14] integrates mobile devices with the Moodle LMS by using Web Services as an extension of the architecture detailed in [15], where the IMS-LTI and the OKI definitions are used to allow the distribution of LMS services to mobile scenarios. However, in spite of the fact that Moodbile provides an extension of Moodle Web Services for mobile integration, the LMS constitutes the central piece. Services, learning objects repositories, collaborative tools, etc. are located in the LMS, and the PLE that can be built is restricted to those services provided directly from the LMS.

Searching for a non-centralized approach, we can integrate not only components and services, but also intelligent agents that support the teaching/learning process during its different stages as we tested in our previous works [16]. However, despite its reuse and interoperability capabilities, as new services and agents are added in order to support more stages in the teaching/learning process, the necessity of creating a centralized single access point where students and teachers work and which shall be in charge of leading them to the appropriate learning tool, service and content becomes clear. This situation has led us to look for a hybrid alternative, where a central access point is necessary, but services can be dynamically added on demand by the teaching/learning process.

The work we present in this paper comes from the issues we have faced while dealing with the application of eLearning standards and reference architectures in order to integrate systems that support the whole teaching/learning process by using the most suitable tool at every stage, but also integrating the various components, so that they can make use of a synergetic effect.

Thus, the remainder of the paper will be structured as follows: first an introduction about the motivation for this research and a starting point will be given in order to introduce the problem we face; then, the implementation we have performed to allow the interoperability among the different learning tools will be discussed in detail; finally, some concluding remarks and future works will be highlighted.

## 2. Motivation and starting point

As introduced previously, our starting point was the distributed architecture we presented in [16], whose main purpose is to integrate and to communicate services and agents in a standard way. Briefly, it consists of a blackboard-based architecture, where heterogeneous distributed components are integrated and communicate among each other by using a Component-Based Software Engineering (CBSE) [17] approach. Moreover, even the user environments have been implemented by using component integration. From this perspective, we are able to build and to integrate different kinds of components such as services, agents, clients, etc. Then, as the teaching/learning process requires specific components, they will take part. In this way, we are capable of implementing environments that give full support to the teaching/learning process, taking advantage of the synergy effect created by the integration of the different components.

This architecture is the basis of COALA (http://chico.esi.uclm.es/coala) [18–19], a distributed Eclipse-based environment to learn to program, which makes use of Adaptive Systems techniques to guide the learning process, and code analysis to provide feedback and advice [20]. Starting from COALA, which has been used in several programming courses since 2008, the architecture has proved to be scalable and extensible allowing the integration of new agents, services and tools like those provided by Cole-Programming (http://chico.esi.uclm.es/coala/index.php/COLE-programming) [21] and other related projects like Edunet (http://

chico.esi.uclm.es/coala/index.php/Edunet) and TupleLD (http://chico.esi.uclm.es/coala/index.php/Tuple-LD).

A tuple space server [22] constitutes the central piece that allows the communication and coordination. It is a centralized distributed memory where the relevant information is written and read by the different components that take part in order to assist in the teaching/learning process. Among the information the tuple space server stores, we can find user session data, communication messages used in different tools (chat, forum, etc.), data related to the learning activities, information associated with the learning activities sequencing, students' scores on assignments, etc.

In addition, our system identifies a set of services and agents classified in layers. We do it in a similar way to the one that JISC puts forward, namely: a *''User Agents Layer''*, a *''Learning Domain Layer''* and a *''Common Service Layer''*, and additionally we have added the *''Communication Middleware''*. By means of the *User Agents Layer*, users can interact with the system and work with the services through the use of both, an Eclipse-based environment or a Web-based environment. The Eclipse-based environment offers advanced programming capabilities while the Web-based environment will provide an alternative and easy-to-access user interface for the same services. The *Learning Domain Services* and the *Common Services Layer* provide the features necessary to provide learning activities sequencing, authentication facilities, auto-assessment feedback and support for Computer Supported Collaborative Learning (CSCL). Finally, the *Communication Middleware* is the central piece we mentioned above, which is constituted by the tuple space server.

Currently, in order to allow the integration of new tools that give support to more specific programming related topics, we are attempting to add Greedex (http://www.lite.etsii.urjc.es/greedex/) [23–24] and GreedexTab (http://chico.esi.uclm.es/greedextab/wiki) within the context of our research work. The learning environment will be able to interactively assist in the active learning of greedy algorithms thanks to the integration of these two tools. Additionally, Greedex, a standalone application, and GreedexTab, an iPad application that makes use of cloud services, will both be able to benefit from the services provided by COALA.

However, in spite of the reuse and interoperability capabilities of our architecture, as the number of services and agents scales while giving support to more stages in the teaching/learning, we realized there was a necessity to create a centralised single access point where students and teachers work and which shall be in charge of leading them to the appropriate learning tools. This leads us to the architecture designed by Brusilovsky and known as KnowledgeTree, where four kind of components are identified [25]: *''the learning portal''*, which provides a centralised single-login point where students and teachers work using all the learning tools; *''the activity services''*, in charge of hosting interactive and adaptive learning content, as well as learning services such as discussion forums, shared annotations, etc.; *''the value-adding services''*, that consider functionalities such as adaptive sequencing, annotations, visualisation, etc.; and *''the student model service''*, a component that represents the students' needs and the prospects in the teaching/learning process in order to personalise the learning material for each individual student. That is, we needed a centralized single access point that acts as *''the learning portal''* in the Brusilovsky's KnowledgeTree.

To solve this situation, we decided to add the services provided by a generic LMS [26], like users and course management, learning object distribution, etc. To do so, essentially, we have added an LMS as a service [27] in our architecture. However, this was not an easy issue to overcome. In the following sections we will show why and how we have worked out.

## 3. Adding an LMS as a service

In this section, we will present how the necessity of introducing a centralized single access point leads us to seek mechanisms of bidirectional communication and information exchange among the LMS and the other services.

### *3.1 Choosing the LMS*

There are several LMS that we can choose, both open source and commercial [28–30]. Among all of them, we have chosen Moodle, the LMS created by Dougiamas (https://moodle.org), not just because it is open source, but also because it is the one available in our institutions, which enabled us to better test our approach.

### *3.2 The LMS programming interface*

In order to implement the communication with Moodle as well as to add new features, we have to take into account its architecture and Application Programming Interface (API), analysing how easily it can be integrated with other systems. In brief, the Moodle architecture is divided into large blocks, namely, the *Core*, the *Activity Modules* and the *Plugins*, all of them allowing their corresponding APIs and capabilities.

Firstly, the *Core* constitutes the basic components accessible by an API. Secondly, the *Activity*

*Modules* implement the necessary services in order to perform the corresponding learning activities. These services are tools such as chats, forums, wikis, etc. In order to access the APIs of these two groups of components, we have to use the corresponding PHP interface.

Finally, the *Plugins* provide specific extensions to the LMS. One of the greatest advantages in the use of these extensions is that, in order to communicate with them, we can use different kinds of interfaces. One of these interfaces is the Web Service Interface, which enables external software to add Web Services components into the Moodle architecture, making the communication with external tools easier.

### *3.3 Launching external tools with IMS-LTI*

Since the LMS will constitute *the learning portal* (following Blusilovsky's nomenclature [25]), and it will provide the centralised single-login point which users will access in order to start working and if necessary to switch to other learning tools and services in order to perform the learning process, we need some kind of mechanism that allows the LMS to launch external components from an opened session while working with Moodle. That is, we require a connection with external services, applications and contents by using a web-based connection.

We can find a solution in the IMS-LTI [8] standard specification, which is part of the IMS standard Common Cartridge [31]. The aim of LTI is to integrate rich learning applications supplied by external *Tools Providers* and used by *Tools Consumers* (see Fig. 1). It follows a provider-consumer approach. Typically, the *Tools Consumers* are LMSs and *Tools Providers* will be those applications we want to integrate within the LMS.

From this perspective, the goal of this specification is to set, in a standard way, the mechanisms to allow integration of external assessment applications, virtual labs or any other web applications hosted out of the LMS. The only limitation is that the external applications must be accessible by using the HTTP protocol, so that the LTI-Consumer sends the LTI-Provider the necessary parameters to launch via POST or GET requests. In addition, due to the fact that the external application could need user authentication, the LMS, as the central login-point it represents, must grant the corresponding permissions to the external application. To do so, IMS-LTI suggests using the OAuth protocol, which allows standard secure authorization (http://oauth.net/).

The IMS-LTI specification has been successfully integrated into several LMS such us Moodle or Sakai (https://sakaiproject.org/). Particularly, Moodle has two IMS-LTI modules: one to make it work as an LTI-provider and another to act as an LTI-consumer. In addition, as an alternative, the BasicLTI4Moodle module implements a basic LTI-consumer (https://code.google.com/p/basiclti4-moodle/).

In our deployment, we have used the IMS-LTI module that allows Moodle to act as LTI-consumer. Adding a new LTI-provider is as easy as register and configure the new service's access parameter by using the corresponding Moodle form. Fig. 2 shows the entry for COALA as LTI-provider.

### *3.4 Returning information back into Moodle*

So far, we have presented how to be able to launch external services on the condition that they are accessible via URI and using HTTP protocol. However, occasionally the external service might need to send back information to the LMS, such as a bookmark, a record indicating that the learning activity has finished, a score the student has obtained, etc. This is what we have called the *loop-back communication*.

In order to implement it, there is some Moodle APIs such as the *Activity Completion API* to indicate the user has finished the tasks, or the *Gradebook API* to access and store students' scores. These functionalities are available via the corresponding Moodle WS API. To allow COALA to access these functionalities, we need to activate the corresponding protocols and configure the right service's grants from Moodle. As we can see in Fig. 3, the first step we have to do is to activate the corresponding web services protocols, in our case the REST and SOAP protocols in order to access the corresponding WS APIs (step 1). Next, we need to grant access to the specific external services (step 2) for explicit func-

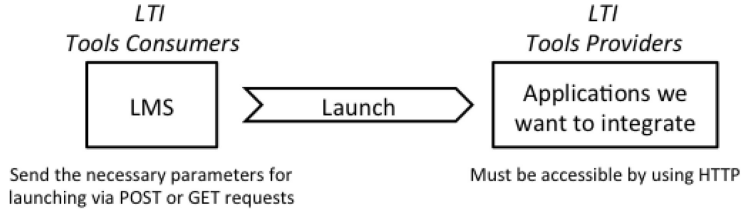


**Fig. 1.** IMS-LTI working scheme.

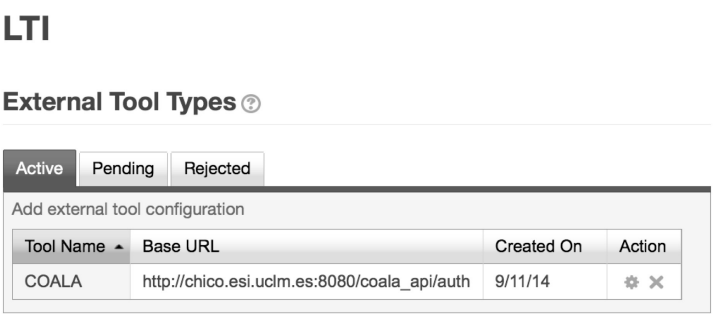


**Fig. 2.** Registering the COALA LTI Tool provider into Moodle.

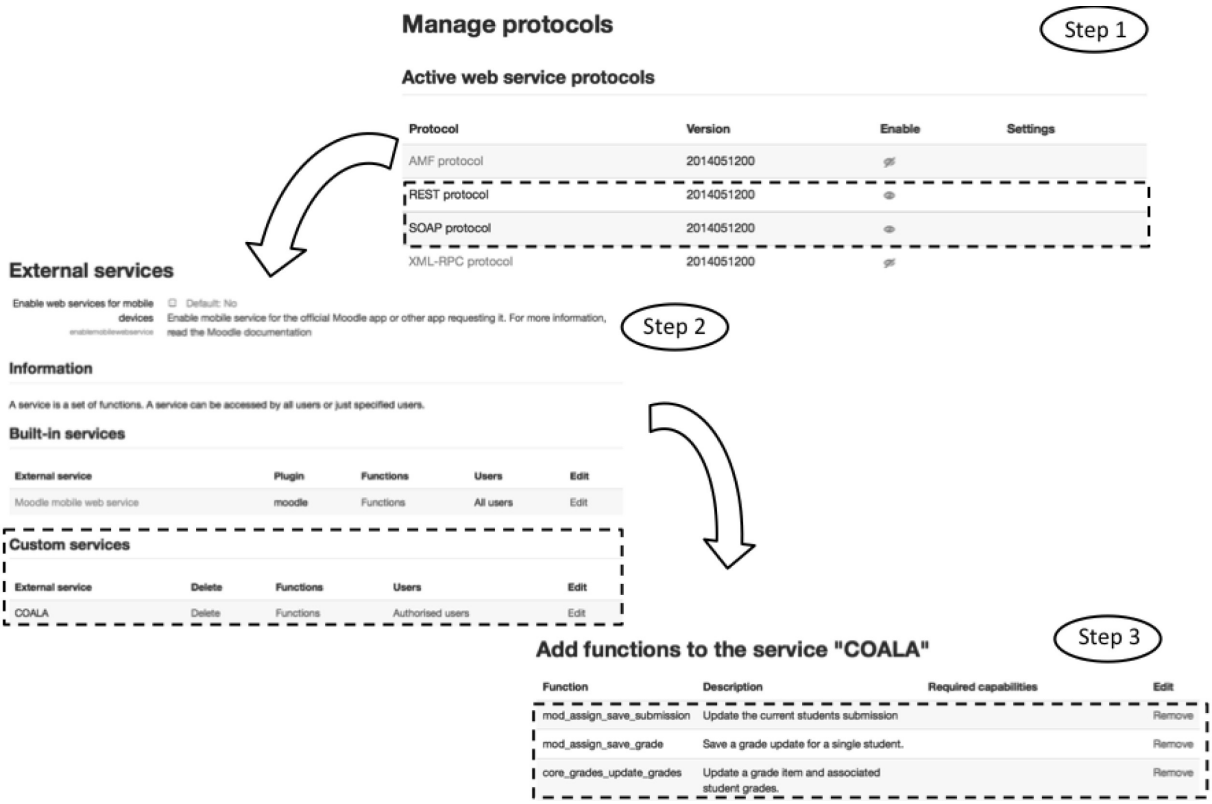


**Fig. 3.** Giving access to specific external services for explicit functionalities.

tionalities (step 3). In particular we have given access for updating students' submission, saving grades updates, and updating grade items and the associated student grades.

### *3.5 Putting everything together*

In order to follow the explanation, Fig. 4 summarizes our approach. In the figure, we can see the mechanisms that allow launching external applications from within the Moodle platform, but maintaining Moodle as the centralized single-login point, as well as the way the information is sent back when the external learning activity finishes.

At the beginning, the user will access the programming course available on Moodle and will follow the course as usual. If it is necessary to perform a specific programming assignment then Moodle will act as an LTI-Consumer and will start the mechanisms to launch the external application. Thus, it will invoke the external service and send an OAuth authentication message to a component we called the *TupleSpace Connector* (step 1). Basically, this component acts as an LTI-Provider for Moodle and as a bridge with the blackboard architecture.

To do so, it is hosted in a well-known host and port and provides a Web Service API. Specifically in our system, it implements a Representational State Transfer (REST) API [32] by using the Spark microframework (http://www.sparkjava.com/). These features make this component easily accessible for the LMS and much faster in the execution due the nature of a RESTful system. In order to allow the

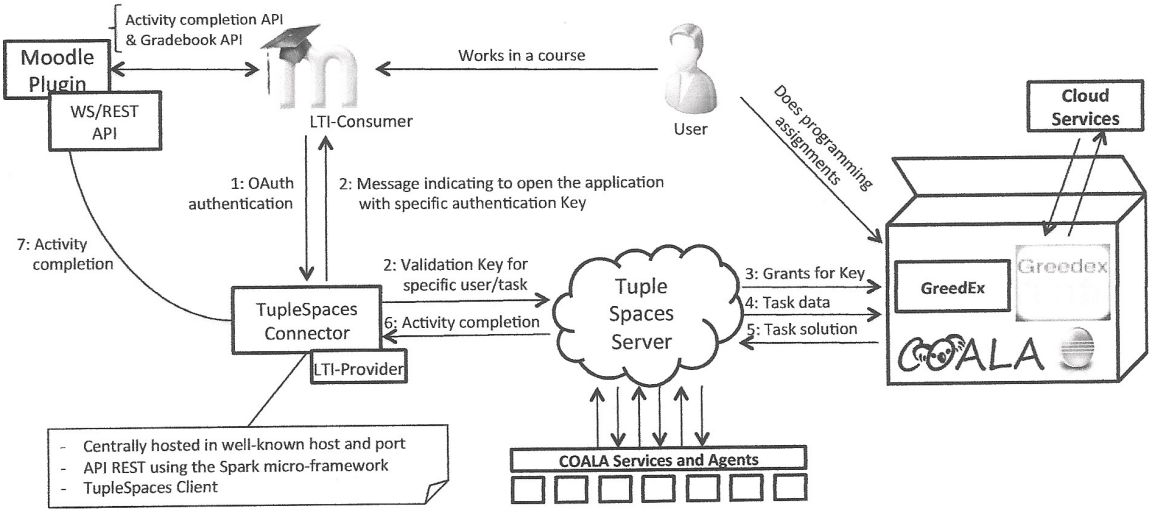


**Fig. 4.** Information exchange among the COALA Eclipse components and Moodle.

communication with the blackboard architecture, the TupleSpace Connector works as a TupleSpace client more.

Then, when the TupleSpace Connector receives the OAuth message, it will send two messages containing a validation key for that user in that task (step 2). Both Moodle and the TupleSpace server will receive this message. The one received by Moodle will be handled to prompt the user (see top of Fig. 5). The received by the TupleSpace server will be used to validate the user for that assignment. At that moment, the TupleSpace stores all the grants the user needs in order to perform the programming assignment.

Thus, the user can open the correct application and once she introduces the validation key Moodle prompted (see bottom of Fig. 5), it will receive permission (step 3) and download all the data to perform the assignment (step 4). A User can benefit from the features of each application, either, automatic assessment and collaborative tools like those provided by Cole-Programming in COALA, algo-

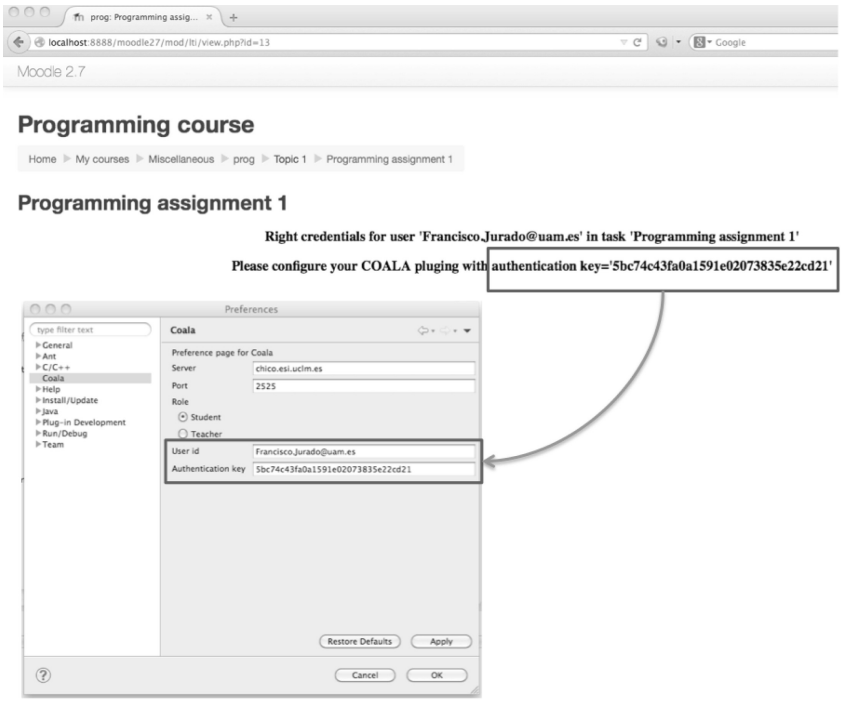


**Fig. 5.** Key configuration to allow COALA to update the user's grades in Moodle.

rithm analysis and execution visualization like those provided by GreedEx, or a cloud storage service like those provided by GreedexTab.

Once the user has finished their programming assignment they can create the solution (step 5). This action will start the corresponding activity completion message to the TupleSpace Connector (step 6) and the loop-back communication acts by using the Moodle Activity completion API to indicate to Moodle that the user has finished the tasks, and access the Moodle Gradebook API if a score must be stored (step 7). After the communication loop is closed, the user can continue working on the Moodle course until a new learning activity that requires an external eLearning tool is necessary.

As we can see, this approach can be extrapolated to any other eLearning tool. Thus, the user has a whole set of learning tools at their disposal, each to work on a specific feature, but the login-point is centralized in the LMS and the orchestration is performed through the Tuple Space server.

## 4. Conclusions

Throughout this paper, we have presented how, starting from an architecture that allows the integration of heterogeneous eLearning components, the necessity of introducing a central login-point comes up as the number of components increases due to their use during several academic years. Then, we have presented how we have solved this issue by introducing the Moodle LMS as a service in our architecture using IMS-LTI and the Moodle Web Services API. Although the COALA architecture has been widely validated, our current research centres on using this new approach to contrast results and experiences.

The solution we provided is not limited to the learning domain we have applied to, and it can be extrapolated to other courses where the use of external services outside of the LMS is necessary. Particularly, the current tendency in the use of MOOCs poses new challenges to manage online open courses without limiting the amount of students. So, to explore the use of our approach in these kinds of courses seems to be the natural evolution of our research. The MOOC platform could be the central access point that manages the course and the students can use the necessary services in order to develop the assignments. In addition, to give support to autonomous active learning processes, the corresponding automatic feedback about what goes well and what does not can be given to them without waiting for the teacher, which is one of the strengths of our COALA system.

Continuing this work, currently we are exploring a new perspective where cloud services from companies like Google, Apple or Amazon can be integrated into our architecture in a standard way as an additional service in order to create richer Personal Learning Environments.

*Acknowledgements*—This research has been partially funded by the Spanish Ministry of Economy and Competitivity through the project *"Software environment for learning to program in group and its integration, by mean of standards, with learning management systems"* (in Spanish *"Entorno software para el aprendizaje en grupo de la programación y su integración, mediante estándares, en sistemas de gestión del aprendizaje"*) (REF: TIN2011-29542-C02-02) with evaluation of ANEP, and the project *"Flexible Model-Driven Engineering for Mobile, Open, Dynamic Data Systems"* (REF: TIN2014-52129-R), by the Ministry of Education, Youth and Sports, Community of Madrid through the project *"eMadrid-CM: Research and Development of Educational Technology in Madrid"* (in Spanish *"eMadrid-CM: Investigación y Desarrollo de Tecnologías Educativas en la Comunidad de Madrid"*) (REF: S2013/ICE-2715) and through Thematic Network 513RT0481 (in Spanish: "*Red iberoamericana de apoyo a los procesos de enseñanza-aprendizaje de competencias profesionales a través de entornos ubicuos y colaborativos U-CSCL"*).

## References

1. A. McAuley, B. Stewart, D. Cormier and G. Siemens, In the Open: *The MOOC model for digital practice*. SSHRC Application, Knowledge Synthesis for the Digital Economy, 2010
2. A. Garcia-Beltrán and R. Martinez, Web Assisted Self-assessment in Computer Programming Learning Using AulaWeb, *International Journal of Engineering Education*, **22**(5), 2006, pp. 1063–1069.
3. P. Molins-Ruano, C. González-Sacristán, F. Díez, P. Rodriguez and G. M. Sacha, Adaptive Model for Computer-Assisted Assessment in Programming Skills, *International Journal of Engineering Education*, **31**(3), 2015, pp. 764–770.
4. SCORM 2004, Advanced Distributed Learning, http://www.adlnet.gov/scorm/. Accessed March 2015.
5. IMS Abstract Framework: White Paper, IMS Global Learning Consortium Inc., 2003, http://www.imsglobal.org/af/. Accessed March 2015.
6. IMS Service Oriented Architecture (SOA): Adoption of Service Oriented Architecture for Enterprise Systems in Education: Recommended Practices, IMS Global Learning Consortium, Version 1.0 White Paper. 2009. http://www.imsglobal.org/soa/imsSOAWhitePaper_v1p0pd.html. Accessed March 2015.
7. IMS General Web Services Base Profile, IMS Global Learning Consortium Inc., 2005. http://www.imsglobal.org/gws/. Accessed March 2015.
8. IMS Learning Tools Interoperability Basic LTI Implementation Guide. Version 1.0 Final, IMS Global Learning Consortium. 2010. http://www.imsglobal.org/toolsinteroperability2.cfm. Accessed March 2015.
9. A. Cerezo, O. Pecalba and A. Garcia, Integration of an Educational Videogame in E-Learning Platforms using IMS LTI, *EDULEARN12 Proceedings*, IATED, 2012, pp. 1700–1709.
10. J. Fontenla, R. Pérez and M. Caeiro, Using IMS basic LTI to integrate games in LMSs—Lessons from Game-Tel, *Global Engineering Education Conference (EDUCON)*, 2011 IEEE, 2011, pp. 299–306.
11. P. Orduña, S. Botero Uribe, N. Hock Isaza, E. Sancristobal, M. Emaldi, A. Pesquera Martin, K. DeLong, P Bailey, D. Lopez-de-Ipiña, M. Castro and J. Garcia-Zubia, Generic integration of remote laboratories in learning and content management systems through federation protocols, *Frontiers in Education Conference*, 2013 IEEE, 2013, pp. 1372–1378.
12. D. Dagger, A. O'Connor, S. Lawless, E. Walsh and V. Wade, Service-Oriented E-Learning Platforms: From Monolithic

Systems to Flexible Services, *IEEE Internet Computing*, **11**(3), 2007, pp. 28–35.
13. M. Alier, E. Mayol, M. J. Casac, J. Piguillem, J. W. Merriman, M. Á. Conde, F. J. Garcia-Pecalvo, W. Tebben and C. Severance, Clustering projects for eLearning interoperability, *Journal of Universal Computer Science*, **18**(1), 2012, pp. 106–122.
14. M. J. Casany, M. Alier, E. Mayol, J. Piguillem, N. Galanis, F. J. García-Peñalvo and M. Á. Conde, Moodbile: A Framework to Integrate m-Learning Applications with the LMS, *Journal of Research and Practice in Information Technology*, **44**(2), 2012, pp. 129–149.
15. M. J. Casany, M. Alier, M. A. Conde and F. J. García-Peñalvo, SOA Initiatives for eLearning: A Moodle Case, *Advanced Information Networking and Applications Workshops WAINA'09*. International Conference on, 2009, pp. 750–755.
16. F. Jurado, M. A. Redondo and M. Ortega, Blackboard Architecture to Integrate Components and Agents in Heterogeneous Distributed eLearning Systems: An Application to Learning to Program, *Journal of Systems and Software*, **85**(7), 2012, pp. 1621–1636.
17. I. Crnkovic, S. Larsson and M. Chaudron, Component-based development process and component lifecycle. *Proceedings of the ICSEA'06*, 2006, IEEE.
18. F. Jurado, A. I. Molina, M. A. Redondo, M. Ortega, A. Giemza, L. Bollen and H. U. Hoppe, Learning to Program with COALA, a Distributed Computer Assisted Environment, *Journal of Universal Computer Science*, **15**(7), 2009, pp. 1472–1485.
19. F. Jurado, M. A. Redondo and M. Ortega, eLearning Standards and Automatic Assessment in a Distributed Eclipse Based Environment for Computer Programming Learning, *Computer Applications in Engineering Education*, **22**, 2014, pp. 774–787.
20. F. Jurado, M. A. Redondo and M. Ortega,Using fuzzy logic applied to software metrics and test cases to assess programming assignments and give advice, *Journal of Network and Computer Applications*, **35** (2), 2012, pp. 695–712.
21. F. Jurado, A. I. Molina, M. A. Redondo and M. Ortega, Cole-Programming: Shaping Collaborative Learning Support in Eclipse, *IEEE-Revista Iberoamericana de Tecnologías de Aprendizaje*, **8**(4), 2013, pp. 153–162.
22. N. Carriero and D. Gelernter, How to Write Parallel Programs: A Guide to the Perplexed, *ACM Comput. Surv.*, **21**(3), 1989, pp. 323–357.
23. J. A. Velázquez-Iturbide and A. Pérez-Carrasco, Active learning of greedy algorithms by means of interactive experimentation, *14th Annual Conference on Innovation and Technology in Computer Science Education, ITiCSE 2009*, ACM Press, 2009, pp. 119–123.
24. J. A. Velázquez-Iturbide, O. Debdi, N. Esteban-Sánchez and C. Pizarro, GreedEx: A visualization tool for experimentation and discovery learning of greedy algorithms, *IEEE Transactions on Learning Technologies*, **6**(2), 2013, pp. 130–143.
25. P. Brusilovsky, KnowledgeTree: a distributed architecture for adaptive e-learning, *Proceedings of the 13th international World Wide Web conference on Alternate track papers & posters*, ACM Press, New York, NY, USA, 2004, pp. 104–113.
26. M. F. Paulsen, *Online Education Systems: Discussion and Definition of Terms*, NKI Distance Educ., 2002.
27. A. Lassila and P. Pöyry, Online education and learning management systems from service-centered perspective, *Proceedings of the sixth conference on IASTED International Conference Web-Based Education*—Volume 2, ACTA Press, Anaheim, CA, USA, 2007, pp. 322–330.
28. *Open-Source Learning Management Systems: Sakai and Moodle*, Monarch Media, Inc., Business White Paper, 2010.
29. J. Jobst, *Course and Learning Management System Project Report and Recommendations*, Information Technology Services, University of Texas at Austin, 2011.
30. J. Cobb, *Steele, Association Learning Management System*, Tagoras, 2013.
31. IMS *Common Cartridge Specification*, IMS Global Learning Consortium. 2013, http://www.imsglobal.org/cc/. Accessed March 2015.
32. R. T. Fielding, *Architectural Styles and the Design of Network-based Software Architectures*, PhD thesis, University of California, 2000.

**Francisco Jurado** is Lecturer in the Computer Engineering Department at the Universidad Autónoma de Madrid, Spain. He received his Ph.D. degree with honours in Computer Science from the University of Castilla–La Mancha in 2010. His research areas include Intelligent Tutoring Systems, Heterogeneous Distributed eLearning Systems, eLearning Standards and Computer Supported Collaborative Environments.

**Miguel A. Redondo** has a PhD in Computer Science (2002) and is Associate Professor at the Escuela Superior de Informática (College of Computer Science and Engineering) at the University of Castilla–La Mancha. His research interests focus on the fields of new Information Technologies applied to Computers in Education and Computer-Human Interaction.