IEEE Global Engineering Education Conference (EDUCON), IEEE, 2013. 1147-1156

DOI: http://dx.doi.org/10.1109/EduCon.2013.6530253

Integrating Open Services for Building Educational Environments

Iván Claros, Ruth Cobos, Esther Guerra, Juan de Lara, Ana Pescador, Jesús Sánchez-Cuadrado
Department of Computer Science

Universidad Autónoma de Madrid (Spain)

***Abstract*— The increasing popularity of Massive Open Online Courses (MOOCs) has raised the need for highly scalable, customizable, open learning environments. At the same time, there is a growing trend to open the services that the companies offer on the web with open APIs and in the form of REST services, facilitating their integration in customized applications. The goal of this work is to show how such open services can be used for the support of on-line educational systems. These services were not created for an education context, so it is necessary to complement it with functionalities for supporting aspects such as evaluations, monitoring or collaboration. This article discusses on the strategies for integrating services for education and presents two cases studies: first, SMLearning, a collaborative learning environment supported by social media platforms *Facebook* and *YouTube*, and second, an application for project-based programming courses, customized through a generative architecture, making heavy use of Google services.**

Keywords- Open Services; Integration; Educational environments; MOOCs, Learning Analytics.

I. INTRODUCTION

Massive Open Online Courses (MOOCs) are educational online web courses designed for large-scale participation and open access [20]. Examples of such initiatives include Coursera (https://www.coursera.org/), Udacity (http://www.udacity.com/) and edX (https://www.edx.org/). Typically, MOOC courses do not belong to official programs of Universities, but learning is normally assessed, and may lead to some kind of certification. While there is an ongoing controversy on the appropriateness of the MOOC approach from a pedagogical point of view (see e.g. [22]), this paper approaches the challenges raised by MOOCs from a purely technical point of view.

The construction and use of MOOCs involves several challenges due to the large number of participants (perhaps in the order of several thousands) and their heterogeneous backgrounds. These challenges include the development and integration of open materials, the design of a highly scalable infrastructure, and the support for high levels of task automation. In this respect, learning analytics techniques are useful to complement traditional knowledge assessment, to track the activity of the users, and to help in improving course organization and content. However, with an increasing number of students able to do complex tasks, the complexity of the analysis of the interaction increases as well. This fact raises the need for the design and support of services simplifying the capture, visualization and processing of students interaction. These services should support processes for monitoring, evaluation and analysis, as a mechanism to measure and characterize a successful learning experience and enabling its replication in other contexts.

Recently, we have witnessed an exponential growth in the availability of services and APIs that different companies make available through the web. Many of them are accessible as REST services, or using languages like JavaScript or Java. Examples of such services include those offered by YouTube, Facebook or Google. For instance, the latter include an API (Google Drive) for file management and sharing, in the cloud, or for analyzing user interaction with a web system (Google Analytics).

While constructing and deploying dedicated learning environments to host MOOCs requires extensive resources, the use of open services can lower the infrastructure required for them. Moreover, the vast amount of available services permits a rich customization of the resulting learning environments. In this paper, we discuss some alternatives to construct learning environments integrating open services, and present two particular applications: one relying on the services offered by Facebook and YouTube, and another one heavily relying on services offered by Google. We also discuss how different learning analysis mechanisms are integrated in them.

The rest of this paper is organized as follows. Section II discusses some strategies to integrate open services for education and reviews some useful ones. Section III presents an approach to build a learning environment embedded in Facebook using open services. Section IV presents another approach based on open services from Google. Section V compares and discusses the benefits of each approach. Secion VI discusses related work and Section VII concludes the paper.

II. OPEN SERVICES FOR EDUCATIONAL ENVIRONMENTS

Educational environments demand support for formal and informal learning on a variety of scenarios. These scenarios involve a complex interaction with the educational environment and rich media resources. Therefore, if such environments are built by combining open services, different strategies are needed to support all interaction modalities.

In our view, an effective service integration process should: 1) encourage the use of existing services, keeping a familiar environment for the users, reducing the cognitive overload that involves a new environment; 2) be based on modular and flexible development paradigm, enabling the easy evolution and extension of its functionality; and 3) abstract away the heterogeneities of the services and technologies used, facilitating their integration.

From these conditions, several integration strategies can be followed:

1) Through **embedded objects** that display information or services over existing platforms, in formal environments such as a Learning Management Systems (e.g. Moodle [10]) or informal environments such as social network platforms (e.g. Facebook). This strategy involves the creation of extension artefacts that the environment uses to include the new functionality. Some examples of these artefacts formats are the Google OpenSocial Gadgets, W3C Widgets or SCORM. Usually, this approach just creates a visual relationship among services, with a limited parameterization of behaviours.

2) **Extending the functionality** of an existing platform. This requires an understanding of the inputs and outputs of the services involved, and handling communication protocols among entities. Hence, the integrated services have to interact with the hosting platform, while in the embedding approach services are isolated objects with no interaction with the hosting platform. A detailed description of these kinds of relationships among services is presented Section III.

3) Creating new learning environments based on a **mashup approach**, where different services are combined to solve the specific needs of learners. This approach is not based on a hosting platform, but the environment is entirely built by interconnecting services. This approach benefits from tools for rapid development, enabling the expression of the learning requirements and orchestrating different technologies for supporting it. Section IV presents these concepts in more depth.

Next, some of the open services that could be used to develop these strategies are briefly presented. Afterwards a classification of integration processes that we have found in our experiences and a typical interaction among applications and Open Services are described.

A. Some open services useful for educational applications

Creating, extending or customizing a learning environment requires a large amount of resources, so that their development and deployment is expensive. The use of open services can help to reduce the cost, by providing mechanisms for the storage and management of resources, as well as mechanisms for composing such services. This ultimately would allow us to build customized learning environments, adapted to the needs of the target learners, easily and without spending much resources.

Collaboration is a key aspect if we want to emulate classroom dynamics in virtual learning environments. Collaborative learning environments allow users to work together, both synchronously and asynchronously, building work spaces where communication (teacher-student, student-teacher and student-student) is fluent. In this context, there are many open services for collaborative work and resource management available. Next, we list some of them. This list is not intended to be exhaustive, but it gathers only the open services that we have used to build our low-cost, tailored, learning environments (see Sections III and IV):

OAuth 2.0 is an authorization protocol that allows a third-party website or application to access a user's data without the user needing to share login credentials. This is the basic building block for integrating different services.

Services from **Google** that we have used include: **Google UserInfo,** which provides profile information about the authenticated user (name, email, profile picture, etc.). The **Google Calendar API** allows, among other functionalities, creating and sharing calendars, as well as their management (i.e. creation of new events in the calendars, and editing and deletion of existing events). The **Google Drive API** provides storage and distributed access for files. It also promotes collaborative work as it supports files to be shared and edited synchronously by several users. The **Google Picker API** provides File Open dialogs in a Web context using modal windows, which in some cases allow showing previews or thumbnails. The **Google Mail API** allows sending emails from a Gmail account. The **Google Analytics API** allows measuring user interactions with services across various devices and environments.

Skype, allows users to communicate with peers by voice using a microphone, video by using a webcam, and instant messaging over the Internet. While the previous functionalities require to have installed a desktop application, Skype also provides services useful for awareness, which allow querying the state of connected users.

Facebook, makes available a set of services that allows: to access user's data through OAuth 2.0; to get information about profile user (such as email, name, location, gender, and other); to manage groups; to send messages; to publish in the bulletin board; to receive notifications; to manage user resources, such as photos and videos, and others. This API is enabled on several platforms and program languages, both based on the web and in mobile environments.

Youtube provides a **Data API,** allowing searching for videos, retrieve standard feeds, and get the content's metadata. A program can also authenticate as a user to upload videos, modify user playlists, and more. This API works using XML (or JSON) and HTTP, but also there are libraries for easier abstraction. Google Data Protocol and the Atom Publishing Protocol are the standards upon which the responses are built. Youtube also provides a **Player API**, to control the *YouTube* player using *JavaScript* or *ActionScript*, i.e. to access play, seek, stop, and pause and other methods, which allows creating personalized controls. That could be useful in the implementation of interactivity mechanisms over video objects referenced from Youtube.

B. Integrating and Interacting with Open Services

In our experiences, we have found four types of relationships among the different participants in an interaction: Server, Client and (Open Service) Provider. Client refers to the user browser, and the Server hosts the learning environment.

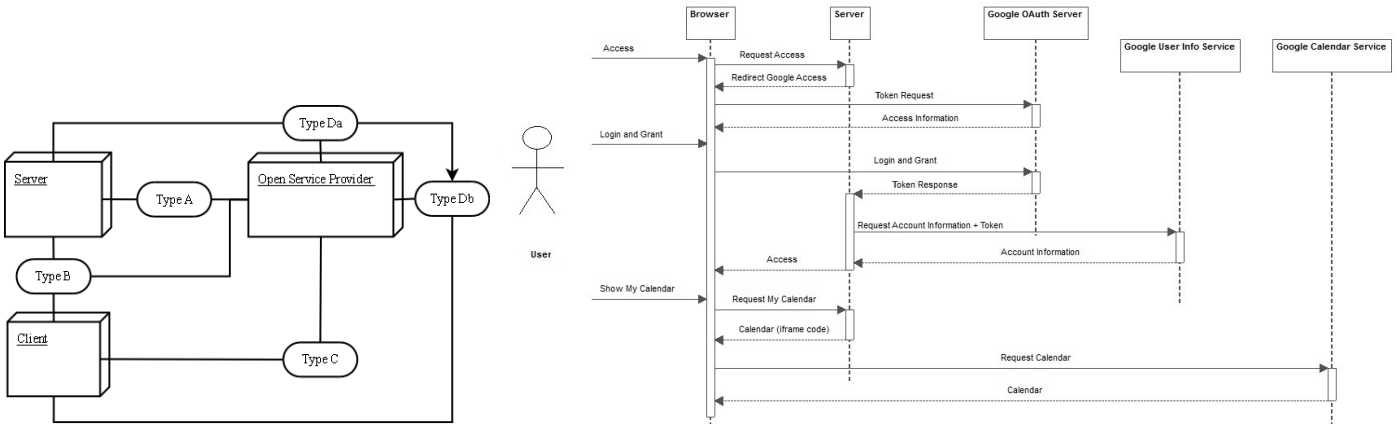

Figure 1. Interactions found when integrating Open Services (left). A typical interaction in an application integrating open services (right)

The left of Figure 1 presents a diagram of these relationships, which are: Server-Provider (Type A), Server-Provider-Client (Type B), Client-Provider (Type C), and Server-Provider-Provider-Client (Type D). More complex integrating processes could be described as a combination of the ones we propose.

- *Type A: Server-Provider*

In this relationship, the Server sends requests to the Provider transparently to the Client entity, i.e. without influencing the user interface. Those requests could be thrown by synchronized user actions or asynchronous server threads. An example is a search service in which the Server creates a remote session with the Provider and transforms both the request and response from a particular provider protocol to its own format.

- *Type B: Server-Provider-Client*

In this relationship, both Server and Provider are in contact with the Client, i.e. a communication process among all entities is required for supporting the service. For example, in an access control process supported by Facebook API, there are validation methods both in the Server as in the Client, so the login request could be initiated by either one of them. In the case of a personalized multimedia reproduction service supported by the API of the Youtube Player, the management of streaming with Youtube is done by the API, but the log of events and handling of sources is responsibility of the Server.

- *Type C: Client-Provider*

This kind of relationship arises when the service that is displayed in the Client is supported just by the Provider without any control by the Server. It is usually presented as an embedded object that deploys an information view supported by the Provider, helping the user with some task. Facebook or Gmail Chat services could be deployed following this pattern.

- *Type D: Server-Provider-Provider-Client*

This interaction arises when a service has no graphical interface (Type Da) but affects other services that have (Type Db). In our experience, this is the case of the Publish Service implemented with Facebook API, which is deployed as a Server-to-Provider request, but produces an update event over the Bulletin Board Service (the users' Wall).

Additionally, there is the usual client-server interaction, where no open service is involved.

The right of Figure 1 shows a concrete example of interaction of an application that integrates several open services. It is illustrated using Google services, but the working scheme is similar in other cases. When the user accesses the application for the first time, the application redirects to the Google page for authorization request (Google OAuth in the figure). Next, the user must login and explicitly grant access to his data. Then, the application gets an authorized request token from the authorization server, which can be exchanged by an access token. In this way, when the user needs to access services which require his authorization (e.g. Google Calendar in the figure), the application can do it by using the obtained access token. Hence, while the authentication process is a Type B interaction, viewing a calendar is of type C, as the browser directly shows the Calendar interface in the web page.

Once we have seen some useful open services, the different strategies for their integration and the typical interactions arising, we next present our case studies.

III. INTEGRATING LEARNING MATERIALS WITHIN OPEN SERVICES: EXTENSION APPROACH

Social media platforms are characterized by allowing a high social interaction among users and for supporting constructive and evaluative mechanisms from content [8]. These conditions allow the development of educational proposals where the students can be actively involved, while fostering high order intellectual skills such as: critical thinking, analysis, conclusion, social skills, and information management [4; 12; 23]. Also, Social interaction among students promotes each other's understanding through support, help and participation in the learning activities [13]. These conditions have promoted the development of a collaborative environment called Social Media Learning (or SMLearning) [7]. Figure 2 presents an architectural view of this System.

Each user group of this environment is called *Community*, and it is responsible of generating interactive material, while they learn and work in a collaborative way. The proposed roles for this process are the following ones: *Author*, *Evaluator*, *Scriptwriters*, *Supervisor* and *Viewer/Learner*. This

architecture has been implemented with Web technologies with a thick client.

SMLearning has been designed to support the construction of Multimedia-Interactive material from a collaborative process perspective. The three fundamental design premises for this System were: reuse of multimedia material, social interaction mechanism for a successful collaboration and interactivity with the content. Figure 3 shows the implementation of SMLearning System like a Facebook application. Hence, this is an example of the extension approach discussed in Section II, as the open services used need to interact richly within the Facebook platform.

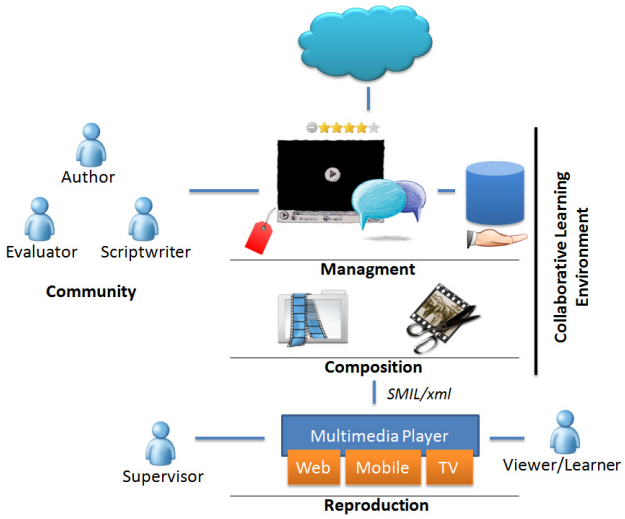

Figure 2. Conceptual model for SMLearning System [7].

Figure 3. A view of the SMLearning System.

For its development several services were supported, such as: managing resources and community; communications services (e.g. comments, forums, chat); evaluation mechanisms (e.g. like/unlike, set a quality level); a tagging system; managing hierarchical lists; a multimedia authoring tool and a multi-platform multimedia player. This implementation includes some services developed using Open Services of Facebook and YouTube that they are described below.

A. Integrating Open Services

The figure 4 presents a deployment view of the System related with integrating Open Services, which includes services and entities, i.e. Server (SMLearning Server), Client (Browser / SMLearning Client), and Providers (Facebook and Youtube Servers). The users require a Facebook account; while the access to Youtube Services is without credentials.

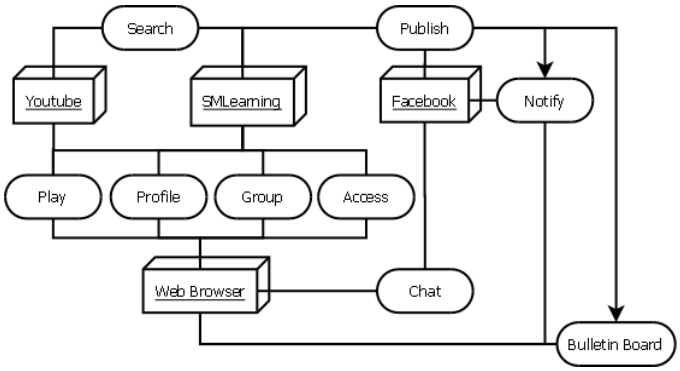

Figure 4. Open service model Integrated on SMLearning System.

From the types of integrating proposed above, the services implemented with Open Services on the SMLearning System were:

- **Type A**: Search is supported by YouTube Data API. This service allows looking for videos and then saves some its metadata.
- **Type B**: Play, Group, Profile and Access are examples of this type. Play is a personalized multimedia player supported by Youtube Player API. For the Group service, Facebook provides both a user interface as methods that allow managing privacy aspects and membership. Additionally, it enables deploying auxiliary modules for sharing files and creating events, and others. A view of User Profile requires the combinations of data extracted both Facebook (e.g. profile picture) as SMLearning (e.g. the user role as teacher or learner). Similarly, the access control is validated on Facebook and SMLearning.
- **Type C**: SMLearning delegates the Chat service to Facebook. Students can use this, and other services as creating events, without intervention of Server.
- **Type D**: Publish is a service that affects two delegated services: Notify and Bulletin Board (the "Wall", in Facebook context). After an invocation of the Publish method, Facebook updates the notification view, and creates a log over the group' wall, allowing the dissemination of information. By default, notification service additionally sends an email to users.

B. Supporting Learning Analysis

SMLearning implements three types of approaches related with supporting Learning Analysis: Summary, Exportation and

Analysis. Each of these approaches presents different levels of abstraction and could be useful at different stages of the learning process. For instance, a Summary View enables monitoring the learners' activities; while an Analysis View, could simplify performance evaluation; however, the detailed analysis of a learning activity requires advanced tools and therefore facilities to extract and format the users' interaction, i.e. Exportation Views. A description of these Views is presented at following:

1) **Summary Views** present compendiums of actions performed by users during their learning tasks. Indicators of progress and effectiveness of a task could be inferred from the amount of user actions, for instance, the number of resources contributed, comments, votes, and so on. This information is presented as data tables or graphics. Some indicators are presented in a temporal way, while others are compared among users.

2) The **Exportation Views** are functions that allow generating detailed reports about user interaction over standard formats. In particular, SMLearning creates a detailed report in CSV format with user actions. Also, it enables the event log dump to ARFF format (Attribute-Relation File Format), which it is used by tools like WEKA, for information processing based on Data mining techniques. Additionally, some indicators related with social interaction, such as relationships among students based on comments, are exported in VNA format which is used by Social Network Analysis tools like NetDraw or Gephi [3].

3) **Analysis Views**. The views presented above address the problems of capturing and displaying the user interaction, however do not define an approach for the analysis of learning. In the particular context of SMLearning, we have created some rules that create relationships among indicators and performance evaluation of learning tasks. For instance, the relevance of the resources contributed is separated in quantity, quality (measured as average ranking proposed by the community in its votes) and social acceptance (measured as amount of interactivity generated around the resource, i.e. comments, votes, tags, and so on). These rules and indicators have been modelled as mathematical expressions that allow automatic generation of assessment reports that the teacher can query.

   Figure 5 presents an Analysis View of social interaction elaborated from annotations made among students. The size of each node is proportional to the amount of comments made by the student and its saturation by the amount of comments received. This approach has been validated with two experiments in which the teachers' opinion about the performance of each student in the activity was compared with the results of automatic assessment. This process has allowed improving the definition of indicators and relationships.

Each of the presented views complements each other in a real context. The teacher can combine these views according to his own approach to the activity and define his own assessment, for example, if social interaction is more relevant that resources contributed.

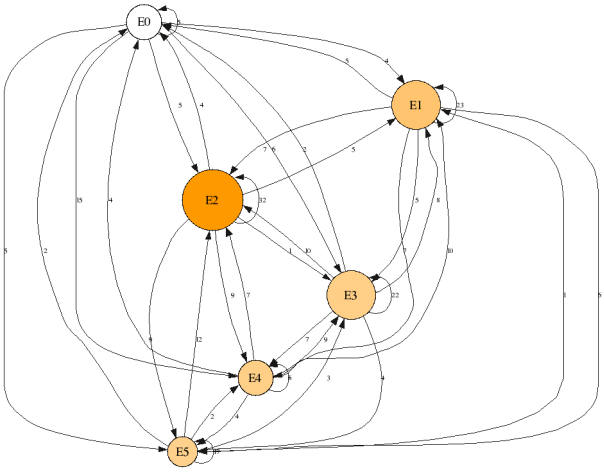

Figure 5. Analysis view of social interaction among students

IV. Building Learning Environments with Open Services: Mashup Approach

A. Architecture

In this case, we follow the mashup approach explained in Section II. In order to cope with the heterogeneity of services and technologies needed to build a collaborative web application, we propose the Model-Driven approach shown in Figure 6.

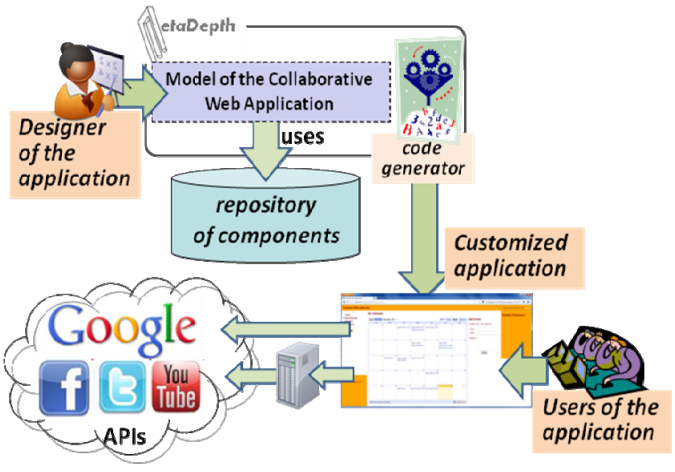

Figure 6. Proposed architecture

In this architecture, the designer of an application does not deal directly with web programming languages like JavaScript, JSP, Java or HTML. Instead, we provide a modeling language so that he can describe the application using concepts of the

domain and not of the technological space. Our goal is to make available a repository of components that implement functionalities given by open services while hiding their complexity, and that the designer can use to build applications easily.

The approach is currently under construction, but a prototype already exists based on the METADEPTH modeling tool [14, 15]. In particular, we have designed a simple family of languages enabling the description of the navigation, content, users, roles, and functionalities of the application as separate concerns. The functionality is described by selecting and instantiating the components of the repository. A code generator produces the final application, integrating and orchestrating the services encapsulated by the chosen components, and using technologies such as JavaScript, Java, JSP, HTML and CSS. Nonetheless, this complexity is hidden to the designer of the application by the use of a modeling language.

The generated application needs from a thin server to perform some coordination, e.g. concerning the awareness of which user is connected, and the different user states. However, most of the functionality is implemented by using open services. Next section provides an example application of the architecture in the e-learning domain.

B. Example Application

We have used the previous architecture for the construction of an environment for collaborative web learning for a project course on object oriented design. In this course, the students work in groups to build an application in Java, covering all phases of the development, from requirements gathering, to implementation and testing. The environment for this course will be accessible by all stakeholders in the subject: the teacher, an administrator and the students, each with different access roles.

For the collaborative work, the students are organized in groups from 2 to 4 people. The application provides the groups with a work place that supports communication between the members of the group, offering the following services:

- Visualization of the state of the other members of the group within the application, including the date of their last connection.
- Methods for synchronous (voice call, chat) and asynchronous (e-mails, calendar events) communication.
- A file management system, which allows accessing and sharing working documents, and permits the teacher a personalized monitoring of the students activity (updates, comments, etc).
- Management of the grades for each group on each deliverable (analysis, design, coding, testing). This includes grading, as well as sending and receiving notifications if desired.
- Shared calendars for organizing group events. By default, a calendar is created and shared between the members of the group (to agree on project meetings), and between the teachers and all the groups (to set dates for delivering the different artifacts, tutorial sessions or exams).

The project course is divided in four phases: Analysis, Design, Coding and Testing, so that the application follows this structure for the organization of documents and grades.

On the other hand, the application allows the teacher to monitor the work done by students, either by groups or individually (visualizing their accesses, shared documents, comments and planned events). Additionally, the teacher can maintain contact with the students in a passive (publishing the different educational materials of the course), or active (making use of the chat, adding comments to documents, sending or receiving e-mails and notifications with the grades) way.

Instead of using a closed solution, e.g. based on Moodle [10], we have implemented the above mentioned environment using open services, mainly from Google. The advantage is that we obtain a highly scalable, customized environment, requiring very few resources, as all materials and services are hosted by Google.

The functionality and components used are summarized in Table 1. Each function is classified according to one of the following general aspects: awareness, communication, coordination, document sharing and evaluation. Sometimes, the same component provides functionality crosscutting several aspects. For instance, we use Skype for both user awareness and communication between the members of a group. We also provide a classification according to the interaction styles for services presented in Section II.B (where "-" denotes a server-client interaction with no use of open services). Finally, we have developed a few components for which we did not found an open service providing the required functionality. One example is the visualization of online users.

Users of the environment can be assigned different roles, by means of which they acquire permissions to perform certain functionalities. We distinguish three roles: *teacher*, *student* and *administrator*. The last three columns in Table 1 summarize the functions each role can perform. In addition, all roles can access the environment after their authentication. *Teachers* can perform most functions, except creating folder structures and calendars, tasks which are performed by the *administrator*. Frequently, both roles *teacher* and *administrator* are played by the same user (the teacher of the course). Finally, note that teachers can perform most functions over all groups (e.g. view any user who is online), whereas functionality for students is restricted to the members of the groups he belongs to, as well as the teacher in some cases.

Figure 7 shows the login page (label 1). This page performs login to the Google OAuth server (see the right of Figure 1 for a working scheme). In this way, the Google server asks for user and password if the user is not already logged in with Google (label 2), and then request for permissions to access the different services required by the application (label 3), like managing calendars, view information about the account, view and manage documents with Google Drive, view the e-mail

Kind	Interac. Type	Function	Component	Role Teacher	Role Student	Role Admin.
	B	Authentication	Google OAuth	Yes	Yes	Yes
Awareness	B	Obtain Profile	Google User Info	Yes	Yes	Yes
	-	Skype name	Skype	Yes	Yes	Yes
	-	Change Status	Self-made	Yes	Yes	Yes
	B	View Online users	Self-made, Skype	Yes (groups)	Yes (teacher and group)	Yes (teacher and groups)
	B	View Users Info	Self-made	Yes (groups)	Yes (teacher and group)	Yes (teacher and groups)
Comm unicat.	C	Voice Call	Skype	Yes (groups)	Yes (teacher and group)	Yes (teacher and groups)
	C	Chat	Skype	Yes (groups)	Yes (teacher and group)	Yes (teacher and groups)
	B	e-mail	Gmail	Yes (groups)	Yes (teacher and group)	Yes (teacher and groups)
Coordinat ion	B	Create/Share Calendars	Google Calendar	No	No	Yes
	C	View Calendars	Google Calendar	Yes (own and groups)	Yes (teacher and group)	No
	B	Create Events	Google Calendar	Yes (own)	Yes (group)	No
	B	View Events	Google Calendar	Yes (own)	Yes (teacher and group)	No
Document Sharing and Management	B	Create Folder Structure	Google Drive	No	No	Yes
	B	View Folder Structure	Google Drive	Yes (own and groups)	Yes (teacher and group)	No
	C	View documents	Google Drive	Yes (own and groups)	Yes (teacher and group)	No
	C	Edit documents	Google Drive/ Picker	Yes (own)	Yes (group)	No
	C	Comment documents	Google Drive	Yes (own and groups)	Yes (group)	No
	C	Upload documents	Google Drive/ Picker	Yes (own)	Yes (group)	No
Eval.	-	Provide Mark	Self-made	Yes (groups)	No	No
	B	Send Mark	Self-made / Gmail	Yes (groups)	No	No
	-	View Mark	Self-made	Yes (groups)	Yes (group)	No

Table 1: Application functionality, components used, and permissions assigned to roles.

address and manage the e-mail. The requested authorizations depend on the functionality of the particular application and user role. Once the login is performed, the user enters in the start page (label 4), where he can see the active online users (lower left panel) and their state. Two kinds of states are provided: the application state (which can be configured in the METADEPTH model, and associated to each page of the application), and the Skype state, which is taken from a Skype service. This page also shows the navigation structure and the events (right panel). The events are classified according to whether they concern the teacher or the students group. In the case of the figure, it only shows the teacher events, because the logged user is a teacher. The center of the page shows some account information retrieved from Google (using the UserInfo API), including a photo, and allows changing the application state (which is also updated automatically depending on the page the user is located).

Figure 8 shows the management of documents by the teacher. In this view, he can add course materials in his different folders (upper part), which are automatically shared among all groups. This is done using the Drive API. In the lower side, he can see the folders of the different student groups.

Figure 9 shows how the teacher can grade the work of the students. The items that can be graded are initially defined in the METADEPTH model (Analysis, Design, Implementation and Testing in our case). There is also a customized notification service that enables selecting the grades to be sent to the users, and adding a general message. The notification is sent through Gmail, as the system can access the e-mail address of every student using the Google UserInfo service.

Figure 10 shows a moment in the interaction of a student with the application. It is part of a process where the student is looking at the comments added by the teacher on one of the deliverables. This functionality is taken from Google Drive.

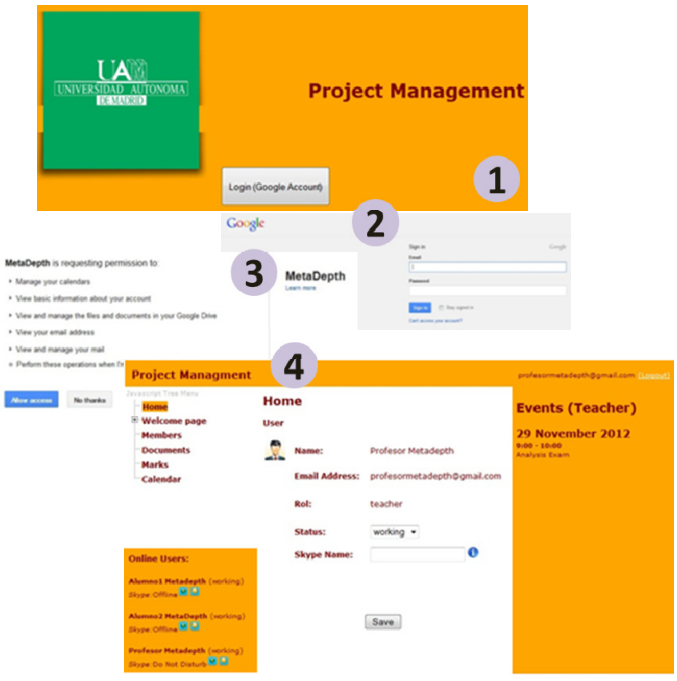

Figure 7. Login process (*teacher* role)

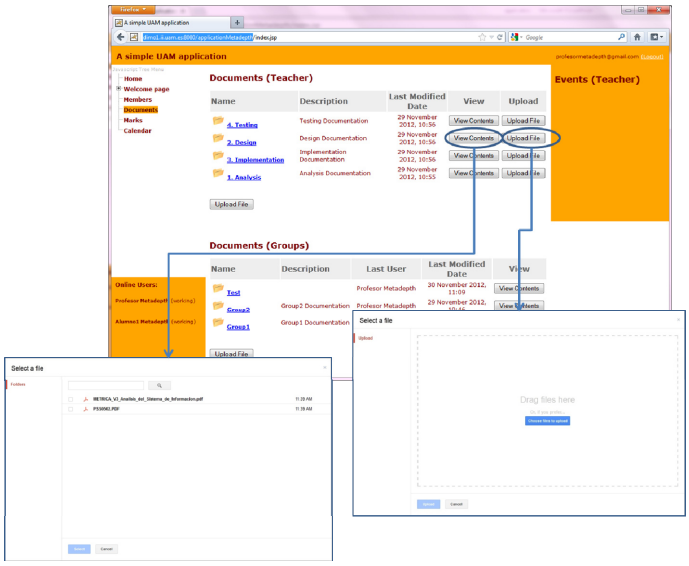

Figure 8. Handling documents (*teacher* role)

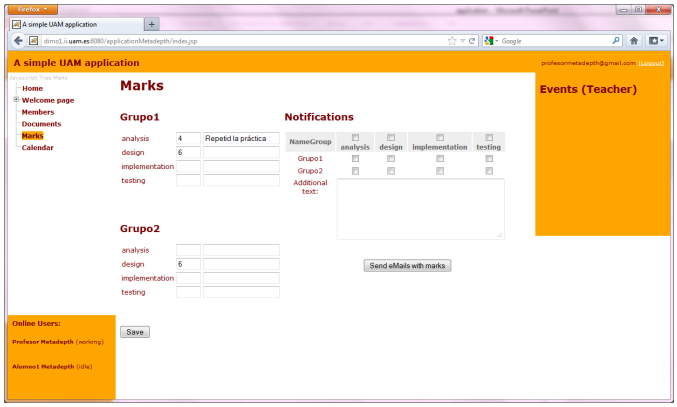

Figure 9. Grading (*teacher* role)

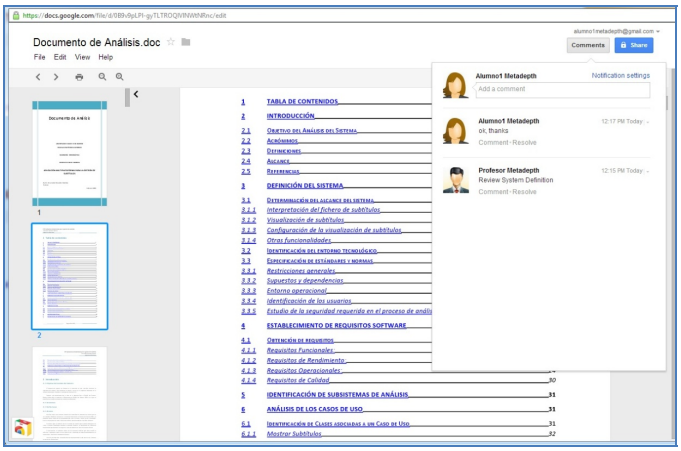

Figure 10. Collaborative work with documents (*student* role)

Finally, Figure 11 shows the management of calendar events by a student. In this view, the student can check the events set by the teacher (as the teacher calendar is shared among all participants), and set new events for his group.

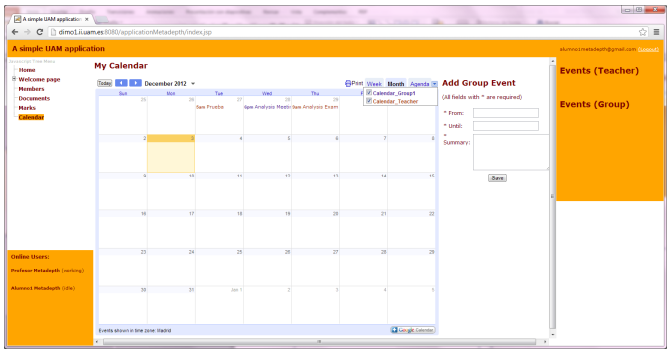

Figure 11. Managing events in the group's calendar (*student* role)

C. *Supporting Learning Analysis*

The use of Google services enables the analysis of the interaction by using Google Analytics. This service permits monitoring and measuring the accesses to each Google service included in the application. A screenshot of a typical report is shown in Figure 12.

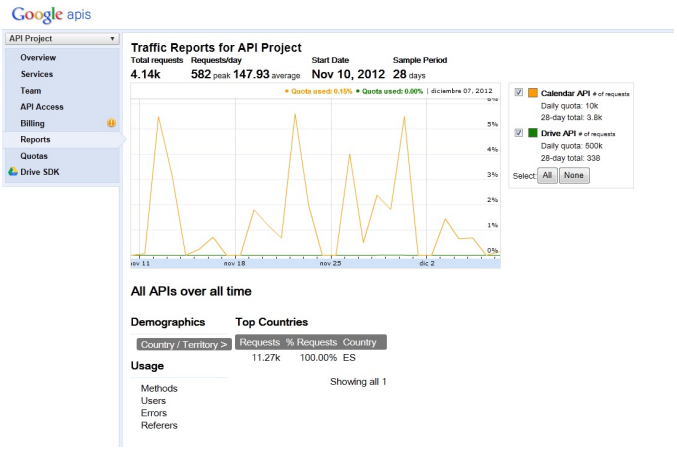

Figure 12. Using Google analytics to analyse service access.

The report shows an aggregation of the use of the Drive and Calendar APIs. This information could be used to analyze the student activity, and to perform optimizations of the environment. However, we are working in providing an improved support for analytics, including a detailed view of student actions, and automated suggestions for optimization of the environment organization. Please note that this would also require from queries not only to the Analytics API but also to e.g., the Drive API, to get the history of the documents uploaded by the students.

V. Comparison and Discussion

The use of open services, especially the related with social aspects, requires a well-management of information privacy. Both teachers and students prefer maintaining separate personal and academic roles [16]. There are some mechanisms to manage this separation, for instance, the Facebook Groups allow putting together people who don't have a friendly relationship, or the Google Plus Circles allow easily grouping contacts. But we believe that these mechanisms are not enough,

so, we recommend defining a separating layer among registered accounts and the services, which ensuring a settable privacy space and taking advantage of social interaction benefits [6].

There is also the issue of security when using information stored in Google Drive accounts. For this purpose, a careful handling of permission is needed, and the Drive API provides rich permission handling capabilities.

An advantage of the extension approach for popular platforms, like Facebook or Moodle, is that the users are not confronted with a new tool, and the learning curve of new functionality is generally lower. On the other hand, a hosting platform may restrict the kind of functionality that can be added or types of interaction of the user, hence being more restrictive than the pure mashup approach. An extension approach can also use services (like the Facebook chat) that are provided by the hosting platform, and which otherwise could not be used in a purely mashup approach. The other way round, for some platforms, it may become impossible to integrate arbitrary open services due to its special requirements.

The use of a pure mashup approach drastically reduces the resources needed to host a learning environment. In the case study of section IV, all material was hosted externally in the Goggle Drive accounts of the participants. This requires fewer resources than hosting an installation of dedicated e-learning platforms, like Moodle.

VI. Related Work

Mashups can be seen as the result of applying software composition techniques to the development of web applications [1]. Much effort is being spent nowadays to propose models enabling rich integration of the different component functionalities. There is a growing need for the integration of educational services [5], and the idea of mashups have also been proposed in the educational domain [18, 19], however a much richer integration of the different services is needed in order to obtain integrated learning environemnts. The work we have presented in Section IV, is an step in that direction.

About social media in educational context, several researchers report using these platforms as a tool for disseminating content. For instance, Laru et. al. analyzed multiple social software tools and face-to-face activities for supporting activities in small groups of learners, where they found that the collective interaction probably increased individual knowledge acquisition during the course [17]. Furthermore, Eggers have analyzed the use of Youtube for sharing resources in the arts fields [11]. Dabner et. al. presents an experience where the University of Canterbury used Facebook for supporting communication activities in times of crisis, particularly a earthquake [9]. Meanwhile, Selami presents a review of Facebook in educational context from some aspects such as: users, uses, harmful effects; effects on culture, language, and education; and they have saw a serious lack of research on Facebook's use as an educational resource [2].

Some works have presented an experience of integrating Google Plus functionalities in higher education context. Particularly, the use of Circles function, which allow supporting a relationships-based approach that seems to improve privacy aspects in the learning environments [21]. But, we believe that there are still few experiences about integrating open services with learning environment.

VII. Conclusions

In this work we have described our approach to use open services for the construction of educational environments. The use of open services enables scalable solutions, appropriate for their use in MOOCs. We have discusses different strategies for the construction, extension and customization of learning environments using open services, and the typical interactions that arise. We have illustrated the approach with two case studies, one using an extension approach, where the learning environment is embedded into Facebook and uses open services from YouTube and the hosting platform. The other one uses a mashup approach, making heavy use of Google services. In both cases, we have discusses the use of learning analytics to improve the learning experience.

We are currently working on improving the tool support, and the generative architecture presented in Figure 6, as well as analyzing new useful services, like those of reference management systems (like Mendeley, http://www.mendeley.com/), Wikipedia (http://www.wikipedia.org/), or diagrammatic web environments like Cacoo (https://cacoo.com). In the future, we will also pursue an integration of the two presented applications.

Acknowledgments

This research was partly funded by the Spanish National Plan of R+D, project number TIN2011-24139; and by the Autonomous Community of Madrid, e-Madrid project, number S2009/TIC-1650.

References

[1] S. Aghaee, C. Pautasso, "*The mashup component description language*". Proc. iiWAS 2011, pp. 311-316

[2] S. Aydin, "*A review of research on Facebook as an educational environment*", Educational Technology Research and Development, 2012, 60 (6), pp. 1093-1106

[3] M. Bastian, S. Heymann, M. Jacomy, "*Gephi: an open source software for exploring and manipulating networks*", Proc. International AAAI Conference on Weblogs and Social Media. From AAAI. 2009.

[4] E. Bogdanov, F. Limpens, Na Li, S. El Helou, C. Salzmann, D. Gillet, "*A social media platform in higher education*", IEEE Global Engineering Education Conference (EDUCON), 2012, pp. 1-8.

[5] M. Caeiro-Rodríguez, M. Manso-Vázquez, L. Anido-Rifón, "*Design of Flexible and Open Learning Management Systems using IMS Specifications*". Journal of Research and Practice in Information Technology, 2012, 44(2), pp. 151-165.

[6] MKC. Cheung, P. Chiu, M. Lee, "*Online social networks: Why do students use Facebook?*", Computers in Human Behavior, 2011, 27(4), pp. 1337-1343.

[7] I. Claros, R. Cobos, "*An approach for T-Learning Content Generation based on a Social Media Environment*", Proceedings of the 10th European conference on Interactive tv and video, ACM, New York, NY, USA, 2012, pp. 157-160.

[8] N. Dabbagh, A. Kitsantas, "*Personal Learning Environments, social media, and self-regulated learning: A natural formula for connecting formal and informal learning*", The Internet and Higher Education, 2012, 15 (1), pp. 3-8.

[9] N. Dabner, "*'Breaking Ground' in the use of social media: A case study of a university earthquake response to inform educational design with Facebook*", The Internet and Higher Education, 2012, 15(1), pp. 69-78.

[10] M. Dougiamas, P. Taylor, "*Moodle: Using learning communities to create an open source course management system*", World Conference on Educational Multimedia, Hypermedia and Telecommunications 2003.

[11] L. Eggers, "*What's on the tube? : art educators on YouTube*". Master's thesis, University of Texas at Austin. 2012. Available electronically from http : / /hdl .handle .net /2152 /ETD -UT -2012 -05 -5279

[12] RM. Gagné, "*The conditions of learning*", Nueva York, Holt, Rinehart & Winston, 1977.

[13] DW. Johnson, RT. Johnson, "*Social skills for successful group work*", Educational Leadership, 1990.

[14] J. de Lara, E. Guerra, "*Deep Meta-modelling with MetaDepth*". Proc. TOOLS (48) 2010, pp. 1-20

[15] J. de Lara, E. Guerra, R. Cobos, J. Moreno-Llorena. "*Extending deep meta-modelling for practical model-driven engineering*". 2012. The Computer Journal (in press).

[16] NS. Lau, L. Lam, "*An Investigation of the Determinants Influencing Student Learning Motivation via Facebook Private Group in Teaching and Learning, Hybrid Learning*", Lecture Notes in Computer Science Volume 7411, 2012, pp. 35-44

[17] J. Laru, P. Näykki, S. Järvelä, "*Supporting small-group learning using multiple Web 2.0 tools: A case study in the higher education context*", The Internet and Higher Education, Volume 15, Issue 1, January 2012, Pages 29-38, ISSN 1096-7516, 10.1016/j.iheduc.2011.08.004.

[18] M. Llamas-Nistal, M. Caeiro-Rodríguez, J. González-Tato, J. Álvarez-Osuna, "*Integrating Web Services with Gadgets to Support an i-Google PLE*". Proc. ICALT 2012, pp. 381-382.

[19] F. Moedritscher, G. Neumann, V.M. Garcia-Barrios and F. Wild, "*A web application mashup approach for eLearning*", OpenACS and LRN Conference 2008: international Conference and Workshops on Community-Based EnVironments, pp. 105-110. 12-16 Feb 2008.

[20] MOOC wikipedia entry: en.wikipedia.org/wiki/Massive_open_online_course

[21] B. Oberer, A. Erkollar, "*Social Media Integration in Higher Education. Cross-Course Google Plus Integration Shown in the Example of a Master's Degree Course in Management*", Procedia — Social and Behavioral Sciences, 2012, 47, pp. 1888-1893.

[22] M. Vardi, "*Will MOOCs destroy academia?*", Communications of the ACM, 2012, 55(11), pp. 5.

[23] RT. White, RE. Mayer, "*Understanding intellectual skills*". Instructional Science, 1980, 9 (2), pp. 101-127.