Web 2.0 services interoperability for E-learning

Antonio CAPODIECI
University of Salento
Lecce, Italy

and

Angelo MARTELLA
University of Salento
Lecce, Italy

and

Roberto PAIANO
University of Salento
Lecce, Italy

ABSTRACT

The use of Web 2.0 technologies in learning has introduced a new generation of E-learning that goes by the name of E-Learning 2.0.

In E-Learning, the contextual use of Web 2.0 tools, like Second Life and Twitter, designed and developed for different use cases, causes problems of lack of interoperability, from which came a phenomenon of fragmentation of knowledge.

This paper reports the results of an experiment in integration and interoperability between Second Life and Twitter, applied to a training course for university staff. This experiment has been successful in facilitating the exchange of messages, and thus knowledge, among the course participants through the use of Twitter in the virtual environment of Second Life.

Keywords: Web 2.0, Interoperability, E-learning 2.0, Collaborative Virtual Environment, Twitter, Second Life, FOAF.

1. INTRODUCTION

The advent of Web 2.0 has brought the emergence of new forms of communication and interaction between Internet users such as micro-blogging and collaborative virtual environments (Collaborative Virtual Environment, CVE). Both systems share the fundamental goal that enable communication and social interaction between people who pursue the same interests, but have different models and paradigms of interaction.

Micro-blogging is a form of small publishing content on the Web in the form of text messages (usually up to 140 characters), images, video, MP3 audio, but also bookmarks, quotes and notes. These contents are published in a social network, visible to everyone or only to people in a community.

Instead, a CVE is a real meta-World where users, represented by avatars, can meet and interact with other users or objects in the virtual world.

The adoption of these systems has impacted so strongly on E-Learning tools to amplify the spread of the new concept called "E-Learning 2.0."

In this Paper will be introduced an experiment designed to integrate two WEB 2.0 services: Twitter and Second Life. The obtained system will be used in a training course for university staff.

2. STATE OF THE ART

As part of learning technologies and methodologies, the use of Web 2.0 tools, such as blogs, podcasts, media sharing and wiki, has impacted in a significant way to bring out a new concept, called e-learning 2.0 [1], [2]. E-learning 2.0 relies heavily on the collaborative nature of learning, in order to simplify the transition from the traditional view of e-learning to a new, technologically driven, knowledge transfer. In this sense, e-learning 2.0 could easily be used to support the learning processes, that involve teachers and students, in order to facilitate and promote collaboration and participation among them. In this context, our research is focused on the concept of collaboration; this target has been pursued introducing in a training course, aimed at technical staff of the university, a social networking tool integrated with a Multi-User Virtual Environment – MUVE [3].

In particular, the tools used for this purpose are:

- **Social network - Twitter** [www.twitter.com]: Twitter is a social network and a micro-blogging service that gives users a personal page updated via text messages (called tweets).

Currently, there are various enterprise tools to interact with Twitter that can also be integrated in ERP and CRM systems, such as:

- **Tweet2tweet** – ([tweet2tweet.appspot.com](http://tweet2tweet.appspot.com)) allows to put in communication two Twitter users and to follow the entire discussion between them, it is similar to that proposed by wall to wall on Facebook.

- **Yammer and GroupTweet** - allows to send a tweet to a group of people identified by the user. All this represents a valuable tool for companies that can send information to an entire community of users. ([www.yammer.com](http://www.yammer.com), [www.grouptweet.com](http://www.grouptweet.com))

Today, Twitter is often used as a communication tool, while it remains rarely used in e-learning context, where it can be implied for a more effective use.

- **Multi-User Virtual Environment - Second Life** ([secondlife.com](http://secondlife.com)) is a MUVE launched in June 2003 by the American company Linden Lab. A free client program, called Second Life Viewer, enables users, represented by avatars, to interact with each other.
The system provides its users, called "residents", the tools to add and create, in the Second Life’s "virtual world", new graphic content including: objects, landscapes, shapes of the characters, audiovisual content, services, etc. Residents of the system can also explore, socialize, meet other residents and manage individual or group activities, create partnerships and implement projects.

SL is considered a platform and a new medium, in many areas (such as learning, training, art, business, etc.) that make use of synchronous and asynchronous communication tools, which are integrated in the same system.

In Second Life, virtual physicals are made out of one or more prim. A primitive (or prim) is a single-part object; multi-part objects will have multiple parts ("prims"). Each prim is represented by a set of parameters, including shape/type, position, scale/size, rotation, cut, hollow, twist, shear, etc. These parameters are sent from a server to the viewer running on the resident's desktop, where the local video card is used to render the visual appearance of everything. Prims can be linked together into link sets, and, in turn, they can also be attached to avatars.

There are currently other efforts aimed at managing the interoperability between systems: Moodle [moodle.org], Second Life and Twitter.

In particular, the M3 project [M3 Project, 2008] at the University of Southampton explores the potential arising from Second Life and Twitter integration, with an organized course on Moodle e-learning platform. The project aims to:

- Assess the potential of virtual learning environment (VLE), Moodle, Twitter and Second Life, between three groups of people within the school community;
- Compare the integrated use of these tools.

The key aspect of the project aims to introduce social networking and 3D virtual learning environment tools within a Moodle course; Twitter plug-in for Moodle represents the core of the project.

Of the three groups of users in the design of a training course, two are made up of many communities of students enrolled in the course respectively in face-to-face or online (distance / on-the-move).

The remaining group consists of professors, researchers and Second Life residents who are interested in the course.

The integration proposed in the M3 project provides a special viewing window in SL to interact with the Twitter portal. This solution provides the user a sense of disorientation, as well as some discomfort in forcing him to switch context passing from one environment to another. In contrast, our experiment introduces into the world of SL, a real object, the prim or Twitter_prim, by which the avatar interacts with Twitter, invoking all the features offered by this, from anywhere and without the need to stand in a special area.

3. SCENARIO AND CHALLENGES

Web 2.0 applications, such as Twitter and Second Life, are intended primarily to the publication and sharing of information generated in their respective areas.

Currently, the use of these systems is extensive, even if separately, in the field of Education as means of teaching support [5], [6].

In this sense, the web application created in this research work aims to integrate services whose nature is heterogeneous and disjoint (Second Life and Twitter), in order to encourage the sharing of experiences on the web; in particular, the web application, using the communication channels available in an "aggregate" way, exploits for an efficient and an effective use of web 2.0 in e-learning [7].

In this way, the user has the option of using the same web interface to interact with the Second Life’s meta-world and with the social network Twitter; in fact, the browser window is divided into two sections in which is present respectively the typical SL viewer and the list of messages sent and / or received by Twitter.

Therefore, it is explored the possibility of integrating a social networking tool in an environment of "immersive" E-Learning, in order to enrich the user experience of the participants, including the use of an emerging technology such as Second Life.

The positive implications of this approach are:

- Strengthening and accelerating the exchange of information between users;
- Development of user skills, including the ability to store and retrieve information at a later time to the course.

From the practical point of view, the avatar has got a wearable object type (such as a bracelet) by which sends messages, while provides the latter with the utmost mobility.

Therefore, the location to interact with Twitter is not bound to the presence of an object, eg. a totem-informative. The sent messages are displayed on the web interface allowing to the avatar to read his tweets, without having to stop at a distinct land or in front of a display.

The requirements that the application must meet in order to enable this are:

- the ability to access to a single master account for both services. Once registered to the system, the user must be able to fill out his profile by entering the login credentials of both Twitter and Second Life;
- the ability to establish a channel of communication between Second Life and Twitter using a prim. This requirement has been met by implementing a middleware capable of intercepting commands from SL and to translate them into appropriate calls written by the Twitter API.

System Architecture and Integration

The system architecture will be compliant with the open source web infrastructure at three levels (see Figure 1) referred to as client, service and web 2.0 layers.

- **Web 2.0 Layer**: it consists of a set of mutually loosely integrated Web 2.0 applications. The integration of these applications will be guaranteed by a set of wrapper classes, providing a common interface for integration and communication with the upper layers. In particular, the wrapper has the task of translating the commands received from the service layer into commands written in the appropriate API. At present, it has decided to provide support only to Twitter micro-blogging system.

- **Service Layer**: it consists of a web server upon which is
installed a module, called middleware, and a MySQL [www.mysql.com] database. The middleware module connects the world of SL with Twitter, intercepting commands from SL and translating them into appropriate commands written by Twitter API.

The Figure 2 shows the class diagram of the middleware.

The organization of classes is compliant with the architectural requirements of the following pattern:

![Class Diagram]

**Figure 1: System Architecture and Integration**

- **Model-View-Controller - MVC**: separates the modeling of the domain, the presentation, and the actions based on user input, into three separate classes.
  - **Model**: The model manages the behavior and data of the application domain, responds to requests for information about its state (usually from the view), and responds to instructions to change state (usually from the controller).
  - **View**: The view manages the display of information.
  - **Controller**: The controller interprets the mouse and the keyboard inputs from the user, informing the model and/or the view to change as appropriate.

Between the available framework on the market, we chose Microsoft ASP.Net MVC.

- **Repository**: separates the logic that retrieves the data and maps it to the entity model from the business logic that acts on the model.

- **Dependency Injection**: the end goal is Loose Coupling providing external dependencies to objects through some external mechanism as opposed to:
  - **Require** to the same objects to set up their own dependencies or even worse,
  - **Require** processes that use the objects to set up the required dependencies.

Generally, component’s dependencies are given through constructors, methods, or directly into fields;

- **Decorator**: The decorator pattern is used to extend the functionality of a particular object at runtime, regardless of any other instances of the class itself. This is achieved through the development of a new decorator class that wraps the original class.

Applying the principles of the MVC pattern, we have identified, ignoring components view, the following model (see Fig. 3).

**Model Classes**

**Users**: is the generic User object;
**Session**: is the generic Session object;
**SL2Twitter_Dbentities**: is the delegate class for interfacing with the MySQL DBMS.

**Controller Classes**

In our research work, controller classes are derived from the abstract class **Controller**, implemented by ASP.NET MVC framework.

In particular, the controller classes implemented are:

- **UserController**: manages the client action relative to the operation on a generic user object. The available main methods are: Create, Details, Edit, GetFriendline, GetPrivateMsg, Search and SearchFiltered.
- **AccountController**: manages the client action relative to the operation on a generic user account object. The available main methods are: Logon, LogOff, Register and ChangePassword.

Applying the principles of the Repository pattern, we identify the following classes:

- **IUserRepository**: interface that defines methods for:
  - obtaining details of a user profile
  - adding a new profile
  - updating an existing profile
  - obtaining the list of a user profile.

**UserRepository**: is the implementation of the interface **IUserRepository**

- **ISessionRepository**: interface defines methods for:
  - obtaining details of a general session
  - obtaining details of a particular user session
  - opening a new session
  - closing a session
  - getting the list of sessions

**SessionRepository**: is the implementation of the interface **ISessionRepository**
Applying the principles of the Dependency Injection pattern, we introduce two service classes, respectively one between the UserController class and the IUserRepository class and the other between the UserController class and the ISessionRepository class. In particular:

UserService: implements all business logic concerned with user object, including the validation logic.

SessionService: implements all business logic concerned with session object, not including the validation logic.

Finally, Decorator pattern is used to make service classes independent from the class presenting the validation errors to the client, in our case ModelStateDictionary. To meet the pattern requirements, ModelStateDictionary class is wrapped in the ModelStateWrapper class, which is the implementation of the interface IValidationDictionary.

Now, the service classes no longer depend on ModelStateDictionary class, but only use the interface IValidationDictionary.

The middleware is written in PHP, and use a library, called Twitterlibphp [jdp.github.com/twitterlibphp], to interface Twitter.

Since the original version of the library is not provided of a feature to search Twitter posts, it has been integrated in that way, in order to allow the avatar of SL to do that.

Whenever the client layer generates a command received from the SL viewer, the middleware checks whether the avatar is registered and whether the application has an active session. To achieve this control, the middleware queries the database that stores the account information and the corresponding open sessions, and only if the control results are positive, the middleware will act on the signal received.

The commands that the middleware is able to process are:
- **status**: allows to update the status of the avatar;
- **direct**: allows to send a message to another user of Twitter;
- **search_general**: allows to search for all tweets that contain the keywords specified by the avatar. The search function, called searchTweets, saves the results into a file with .atom extension;
- **search_foaf**: allows to search for all tweets that contain the keywords specified by the avatar and to filter the results by the author of the tweet, in order to display only messages sent by users in the FOAF profile of the avatar, as well as friends of the SL user.

Client Layer: it consists of a web browser within the SL viewer is forced to run. The encapsulation is achieved using the Xenki [opensimulator.org/wiki/Xenki] solution designed by Adam Frisby.

This solution makes possible to implement the integration of a virtual world, like Second Life and OpenSim [opensimulator.org] with a web browser using the XBAP technology [www.xbap.org] from ASP.NET [www.asp.net]. The source code of this technology is not open source, and requires, in contrast to other solutions such AjaxLife [www.ajaxlife.net], the installation of Second Life client.

![Second Life - Twitter - Interoperability](image)
XBAP Security Considerations

By default, XBAP applications run in a sandbox in partial trust mode (partial reliability). For this reason, the code of these applications may access to confidential information, or perform unauthorized actions on the client. However, in certain situations, such as intranet, may be asked to XBAP applications to run in a full trust mode. In our case, consider that the application runs in the browser, and then in a partial trust environment, we have to enable the full-trust mode. To this end, it was created and installed on the web server, a security certificate, by which certifies the identity of the application’s vendor.

Interoperability

To ensure interoperability between the meta-world of Second Life and the micro-blogging service Twitter (see Fig. 4), we have to grant the following requirements:

- Integration of SL client with the Web browser;
- Interaction of SL with Twitter through an object of wearable type, which grants the maximum mobility for the avatar (Twitter_prim);
- Presentation of the Twitter’s tweets on the web interface.

Following are presented the details of communication diagram related to the Twitter operations of update status and messages research.

Communication Diagrams

Status Update: The diagram in figure 5 shows the messages exchange between the three layers of the architecture during the update status operation. The communication starts off with the submission by the avatar of the command "status", followed by the channel (300) and the status to the Twitter_prim (1). The Twitter_prim, using an HTTP request, sends to middleware the parameters to create a request by the Twitter API (2). Middleware using the API Rest of Twitter updates user status, awaits the outcome of the operation (3) (4) and concludes transferring the outcome of the command to Twitter_prim (5). The Twitter_prim displays the message received from the middleware server (6).

Twitter Message Search: The following diagram (see Fig. 6) shows the exchange of messages between the various entities involved in the search for tweets. The communication starts off with the submission by the avatar of the command "search" followed by a keyword on the channel 300 to Twitter_prim (1). The Twitter_prim, using an HTTP request, sends to middleware the parameters to create a request by the Twitter API (2). Using the API Rest of Twitter, middleware does the search, waits for the results of the operations (3) (4) and concludes transferring the outcome of the command (5) to Twitter_prim. The Twitter_prim visualizes the message received from the server middleware (6). Finally, the web application "asks" and obtains the search results from the middleware (7) and (8).

4. A CASE STUDY

In this section, we introduce the application, also with some screenshot, illustrating the main client scenario.

The use case was developed through the organization of a virtual classroom, consists of 15 university employees who have attended a training course on Java programming. Employees belonging to academic structures distributed, also geographically, on campus. Each participant had his own Twitter and Second Life account.

The high level goals, required by the system, are used to obtain, through a process of refinement, the scenarios and use cases of the application (Fig.7).

Figure 6: Twitter Message Search - Communication Diagram

Figure 7: High-level target diagram

By requirements analysis, we have the following use cases:

- New Profile
- Update Profile
- Update Status
- Direct Sending Message
- Tweet Search
- Filtered Tweet Search.

Without dwelling on scenarios such “New Profile” and “Update Profile”, which dynamics are well-known in the literature, we will discuss the aspects of the most important use cases for the integration between Second Life and Twitter.

In particular, the use case “update status” (see Fig. 8) provides that, for user authentication, the system checks for the username entered in the database, and then, if it is successful, acquires detailed information of the user. At this point, the user starts the SL viewer, and through his avatar updates his status on the prim. Then, the system forwards the message to Twitter and displays the result of the operation. If the avatar hasn’t got the prim for sending messages, he can recover it from its inventory.
Learning system, fulfilling the dual functions of user and developer, breeder and controller of the knowledge sharing in the community.

Other similar experiments attempted to implement interoperability mechanisms, limited to viewing only in Second Life, content retrieved from Twitter [4], but users of the virtual world is impossible to communicate through social networking tools such as. In contrast, in our experiment the avatar can communicate with other avatars and Twitter users through the use of prims inside Second Life.

Finally, possible future developments might explore further different forms of interoperability between social network (first of all Facebook) and E-Learning tools (such as Moodle) with systems used to support academic teaching. In essence, the main target we pursue is to increase the dissemination of information, including educational materials, supplied by the teachers, in order to expand sources and channels of information itself by exploiting the student social networks.

6. REFERENCES