Contextual Spaces with Functional Skins as OpenSocial Extension

Evgeny Bogdanov, Christophe Salzmann, Denis Gillet
Computer, Communication and Information Sciences
Ecole Polytechnique Fédérale de Lausanne
Lausanne, Switzerland
{*evgeny.bogdanov, christophe.salzmann, denis.gillet*}*@epfl.ch*

***Abstract*—Portability, flexibility and extensibility are essential features of social media platforms. When such Web platforms are able to take user's context into account, they provide better user experience and enhance the effectiveness of users' actions. In this paper, we discuss an extension to OpenSocial standard, namely contextual space, that shapes the framework, in which people carry out online activities. The proposed contextual space extension defines how a set of OpenSocial widgets are aggregated as a Web environment for a given purpose and with a given functional skin as a user interface. Additionally it allows to create contextualized widgets. In this paper we discuss the proposed extension in details and provide the examples of its use based on real life scenarios. Finally, we detail an implementation scheme.**

***Keywords*-context, contextualized widget, contextual space, functional skins, portability, OpenSocial, social media platform, Web application, Web environment.**

I. Introduction

The number of social media platforms is rapidly increasing. After the advent of popular platforms such as Friendster, Facebook and MySpace, which are oriented mainly on management of users' list of friends, a great number of new social media platforms appeared and continues to appear. These new platforms are often of a narrow orientation: Twitter focuses on micro-blogging, Youtube specializes on online video streaming, Last.fm is a network for people that like music. Despite their different goals, all of these platforms integrate a notion of friendship and/or connections between people.

Some of these social media platforms started to provide APIs to allow external developers to create widgets (small applications that can be embed into a Web page). Such widgets represent usually a combination of HTML and Javascript code and can easily be added by users into their pages. They can as well take into account information about a user from its hosting platform. Widgets added into users' pages extend the default functionalities of a hosting platform and bring greater personalization experience to users. Thanks to these widgets, social media platforms can attract external developers to augment their functionalities.

OpenSocial Foundation provides a standard API [5] to retrieve information about a user from a social media platform. A special widget standard (OpenSocial gadget) was first proposed and implemented by Google and later spread in other platforms. These widgets can be found everywhere ranging from simple blog sites such as Blogger to social media platforms (Myspace, Orkut, Friendster, etc.) and business-oriented networking solutions (LinkedIn, Oracle, XING, etc.). A social media platform that implements support for OpenSocial can ensure that any widget implemented according to the standard will run properly when plugged into this application. In other words, the widget becomes portable and can be run in different social media platforms. With the advent of a reference open source implementation for OpenSocial API (Apache Shindig), any social media platform can quickly start hosting OpenSocial widgets.

Despite the fact that OpenSocial solves the problem of social media platform's extensibility and widgets portability, it has two major limitations. First, it is user-centric but does not take the user's context into account. In many social media platforms context is a crucial component and if widgets were able to retrieve the user's context (from the hosting platform), it would greatly improve user's experiences [9], [15].

To understand the second limitation, one should distinguish between two types of Web applications, namely, Web widgets and Web environments. While Web widgets are usually considered to be small Web applications (weather forecast widget, translator, calendar), a Web environment is a relatively complex Web application such as a forum, a text editor or a personal learning environment. These Web environments can be seen as meta-components since they might have the capability of managing other Web applications. OpenSocial standard provides good support for Web widgets, but lacks support for Web environments.

In this paper, we introduce a notion of a contextual space as an extension to OpenSocial. This extension enables the definition of portable Web environments combining widgets that can be presented to users with different functional skins to take context and preferences into account. In addition, being the representation of a user's context, space permits the development of contextualized widgets. In this paper, we present the extension details and show the scenarios, in which such extension could be beneficial. Furthermore, we describe an implementation that uses the concept of spaces and we demonstrate portable Web environments as proof-of-concept.

There exist different standards for widgets: W3C standard [4], OpenSocial gadget specification and some other proprietary standards. In this paper, we will be using indifferently "widget" to refer to all standards. Similarly, we will refer to OpenSocial widgets for OpenSocial gadgets.

The paper is organized as follows. Section II discusses the OpenSocial specification. In Section III, we discuss the details of th proposed extension. In Sections IV,V,VI, contextualized widgets, portable Web environments and functional skins are defined respectively. Finally, the implementation of the proposed extension is discussed in Section VII. Section VIII concludes the paper.

II. OpenSocial specification

OpenSocial specification [5] provides a set of common APIs for social media platforms. This API standardizes the way information about people, their friends, application data and people's activities are retrieved. This standard solves two problems at the same time. First, a social media platform developer does not have to create a new naming scheme for his/her API and, secondly, widgets developed according to the OpenSocial specification become portable and can run on any Web platform that implements OpenSocial.

The only requirement for a social media platform is that it has to support OpenSocial specification. Thanks to Apache Shindig [1] (open source reference implementation of OpenSocial specification) any social media platform can quickly add support for OpenSocial and be able to instantiate OpenSocial widgets. By implementing this specification, a social media platform automatically gets the opportunity to extend its own functionality by plugging in OpenSocial widgets.

III. Contextual space and OpenSocial

OpenSocial specification focuses mainly on users. It provides APIs to retrieve list of friends for a given user, list of user's activities, albums, list of user's widgets, etc. It is known, however, that depending on user's context, a list of context-specific tools is needed. A list of people can change from one context to another. The same person might have a completely different role in different contexts. Such context could be a workspace, a forum's topic, a friends discussion or a trip planning. Social media platforms often support the context concept in one form or another. As an example, there might be different discussions and sub-discussions in the same forum, each one with its own topic, own list of participants and own list of resources. Groups or events of Facebook is another example of a context. In case of Learning Management System, every user can have several contexts - one for every course the user takes part in. It is important that Web applications get access to user's context and are able to take it into account.

One of the main limitations of OpenSocial is that it does not support user's context. Such concept simply does not exist in OpenSocial. Currently with OpenSocial it is impossible to model, for example, a university course with participants and resources. Contextual spaces as OpenSocial extension are intended to provide support for context to OpenSocial specification and make OpenSocial widgets more useful for people.

Later in the paper, we will narrow down the definition of the context to a contextual space. We define a contextual space as an aggregation unit that includes list of applications that are to be used in the context, a list of people with different access rights sharing the context, resources that can be used inside such context, and possibly some other subspaces that belong to this context. As the 3A-model [11] suggests, such mapping can be done for any social application [8]. Any social media platform's structure can be mapped into three different entities: Actors, Activities and Assets, where Actors represent people, agents, applications; Assets represent resources, documents, etc.; Activities represent an aggregation unit that combines different Actors, Assets and another Activities together into a context. Activity is a contextual space in our definition and represents the user's context. A contextual space can be further enriched with additional fields to better represent the user's context.

It should be noted that support for people, resources and tools already exists in OpenSocial. However, all these entities are centered around the user and not around the user's context. To avoid confusion, we should note that "Activity" exists in OpenSocial specification but this concept is completely different from 3A-model concept. While in 3A-model word "Activity" means an aggregation unit or a contextual space, in OpenSocial it means an action done by people (user sent a message, user became friends with somebody, etc.), which are two completely different concepts.

Table I presents the proposed OpenSocial extension with contextual spaces. In OpenSocial the "People" service is responsible for retrieving a list of people connected to a user. With the contextual space extension, there is additionally a list of people for every space (for example, list of group members). In order to handle both scenarios we suggest to add a field "type" to the API. By doing this, a pair (gid,type) defines either a space or a person, for which a list of connected people has to be retrieved.

"AppData" service of OpenSocial can be extended in a similar way. By default, this request returns application data for a widget that belongs to a user. With the new extension, a widget can belong to either a person or to a space. Thus application data can be retrieved for widgets belonging to either people or spaces. "Type" parameter in the request allows to specify for which item (space or person) application data should be retrieved.

In addition to above extensions of existing OpenSocial services, we suggest to introduce two new services, namely, "Spaces" and "Applications". Requests for "Spaces" are similar to those of "People" service. The first request in

People	**OpenSocial: /people/{guid}/@all** - Collection of all people connected to user {guid} **Extension: /people/{gid}/@all/{type}** - Collection of all people connected to item with id {gid} and with {type} in ("space","person")
AppData	**OpenSocial: /appdata/{guid}/@self/{appid}** - All app data for user {guid}, app {appid} **Extension: /appdata/{gid}/@self/{appid}/{type}** - All app data for item with id {gid} and with {type} in ("space","person"), app {appid}
Spaces	**Extension: /spaces/{spaceid}/@self** - Profile record for space {spaceid} **Extension: /spaces/{gid}/@all/{type}** - Collection of all spaces for item with id {gid} and with {type} in ("space","person")
Applications	**Extension: /applications/{appid}/@self** - Profile record for application {appid} **Extension: /applications/{gid}/@all/{type}** - Collection of all applications for item with id {gid} and with {type} in ("space","person")

Table I. OpenSocial extension with contextual spaces

the third row allows to retrieve detailed information about a space. The second request allows to retrieve list of spaces or list of people connected to a space. The last request is similar to the extension of "People" request, but instead of returning list of people, it returns list of spaces. With these two requests one can get lists of all people and spaces connected to a specific person, as well as lists of all subspaces of a space and lists of people for a space (or space members).

The service "Applications" retrieves detailed information about a given application with specific identifier (id). The "Applications" request with "type" parameter in the fourth row allows to get list of applications for a person or a list of applications that belong to a space. This request is similar to previously described requests for "People" and "Spaces" services.

Interested readers can refer to Apache Shindig wiki [2] and [12], where implementation details are presented.

IV. Contextualized widgets

Assuming that the proposed contextual space concept is supported by the OpenSocial specification, widgets can retrieve user's context information (such as people, applications and resources in this context) from hosting social media platform. This would provide greater widgets personalization.

By taking context into account, one can create contextualized widgets, widgets that adapt their behavior to user's context. Such widgets can better extend the functionality of a hosting platform. Their visual interface, displayed data and functionality can be changed according to the context, in which user currently is interacting. The same widget might display different people and different resources depending on the user's context.

Let us look at a simple example coming from a learning scenario: a person is in the process of learning German and French languages. This person utilizes a widget that lists language documents from a space and another widget that lists space members. With the proposed OpenSocial extension it is enough for a user to create two spaces "Learn German" and "Learn French" in his/her social media platform. Then s/he can put German documents and friends learning German into "Learn German" space and French documents and friends learning French into "Learn French" space. When user enters "Learn German" space in the hosting platform, widgets display German documents and his/her friends from German space, and when user enters "Learn French" space, the corresponding French items are displayed in the same widgets. Thus these widgets become contextualized widgets.

V. Portable Web environment

The notion of a space and its support in OpenSocial bring us new scenarios, that were impossible to realize before, namely portable Web environments. Such Web environment can be moved between two different social media platforms [13]. With contextual spaces it is possible to model such concepts as group of people, event, discussion, course, conference, etc. All these concepts have the following characteristics in common: they all serve as an aggregation unit to join together people, resources, applications.

A contextual space defines a Web environment and once OpenSocial specification is extended with contextual spaces, a Web environment can be technically implemented as an OpenSocial widget (such Web environment can be seen as a meta-widget [6], that is a widget that can integrate several widgets). People are used to consider a widget as a small application running in their desktop or their browser page. However, as Wikipedia states, "in computing a Web widget is a portable chunk of code that can be installed and executed within any separate HTML-based web page by an end user without requiring additional compilation. They are derived from the idea of code reuse." Thus, even though a Web widget and a Web environment represent different concepts, they both can be implemented according to OpenSocial widget standard (as "portable chunk of code"), which ensures portability for Web environments.

Let us consider a real life example where such approach can be used. Imagine a Web environment *App1* (that is a groups management application) implemented as follows: the server side code is implemented according to OpenSocial specification extended with spaces. The client has a

widget container able to render OpenSocial widgets. Figure 1 depicts the Web environment *App1*. Learning groups are represented as horizontal tabs. When a tab with a group is active, the list of people for this group is shown in the right area and widgets for this group are displayed inside the left area side-by-side. This Web environment is technically implemented as a big OpenSocial widget that receives via OpenSocial list of groups (as spaces) for a logged in user, a list of people and widgets for every group. Since this Web environment is implemented as a widget, its functionality is portable and can be reused in another Web platform. If the owner of *App1* believes such widget (Web environment) to be useful for other people, s/he can add it into one of the available widget repositories (iGoogle, for example).

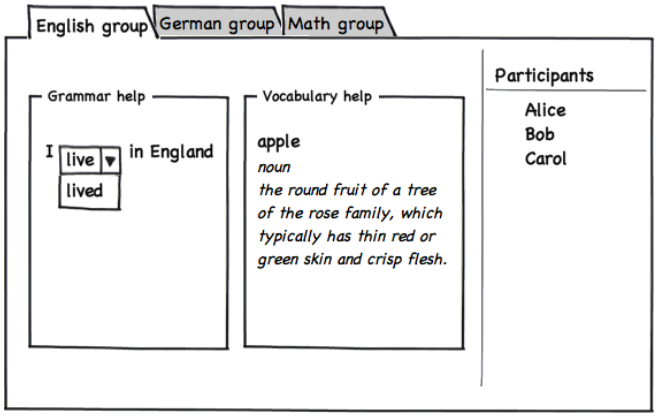

Figure 1. Web environment *App1*

Other people can reuse this Web environment according to the following scenario. A user plans to develop a new Web environment *App2* that will be showing news to people. The user would like to organize the news site in the following manner: s/he wants to have tabs that correspond to the different news topics (music, economics, sport). For every topic the user wants to add some RSS widgets that show news for the topic and a list of people subscribed to this topic. S/he would like to know if similar functionality is already implemented by someone else, so that she does not have to start from a scratch. S/he goes to a widget repository and finds a URI for the OpenSocial application. This application suits perfectly to his/her requirements. The structure and functionality is exactly as expected, however the data are different.

Assuming s/he already has the extended Apache Shindig installed on his/her server, s/he first creates topics (as spaces) and adds widgets and people to every topic. Then, s/he adds the retrieved widget and his/her site starts to function as planned (Figure 2).

This scenario shows that the Web environment functionality becomes portable and reusable. Even though the difference is transparent for an end-user, for Web developers

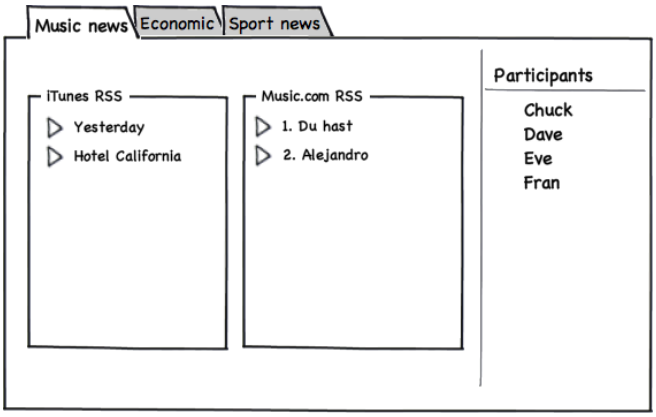

Figure 2. Web environment *App2*

this is a step forward in simplifying the development process. It is true that even without spaces Web widgets are reusable and portable, however without spaces it is difficult to create portable Web environments where the user's context is important (as in the described scenario).

The space extension and portable Web environments can have a great influence on end-user experience too. In the previous scenario we showed how different data can be represented in a similar way through the portable OpenSocial widgets with spaces. The next section presents functional skins, where the interaction with the same data structure can be offered to users differently through OpenSocial widgets extended with spaces.

VI. Functional skins

OpenSocial specification extended with spaces and Web environment (or meta-widget concept) allow people to easily change the way interaction with the information is offered. We define such different interaction schemes as functional skins, where data and data structure do not change, while visual representations and actions that user is allowed to perform with data (functionality) might differ from one functional skin to another.

The main idea behind functional skins is the same as for portable Web environments: Web application implemented as an Opensocial meta-widget that supports spaces and can integrate other widgets.

As an example, we consider the scenario where a knowledgable person in Computer Science decides to create a space to support learners in mastering Computer Science. For this goal, the mentor creates a space "Learn Computer Science". Then learners (Alice, Bob, Chuck) and some subspaces (Introduction, Basic Algorithms, Complex Algorithms) are added to the space. Then the mentor structures information and populates every subspace with widgets and resources helpful to master the Computer Science. The resulting view for a space "Basic Algorithms" is depicted in Figure 3.

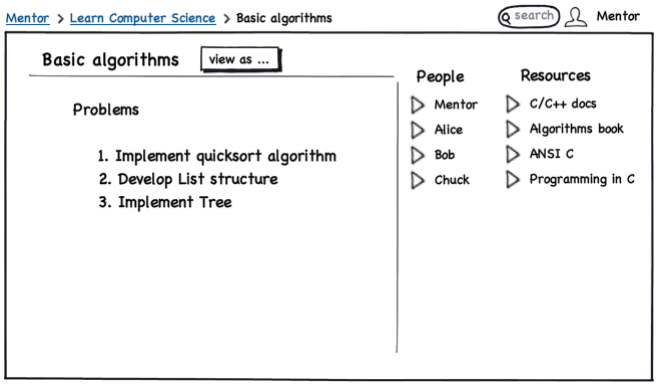

Figure 3. Default functional skin for Basic Algorithms space

During the studying process some of the visual parts are not needed for learners as they might be distracted from a learning process. Thus the mentor wants to provide a special view for learners that provides only the needed functionalities: two widgets displayed side-by-side and a list of resources as a column on the right (Figure 4).

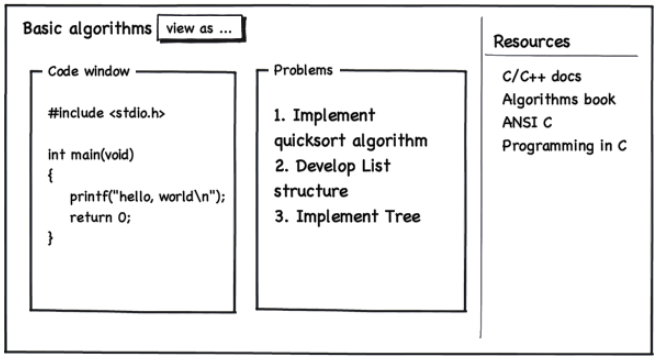

Figure 4. Learning-focused functional skin proposed by mentor

Mentors are provided with a way to change the visual representation of data structure through widgets. The mentor can go to a publicly available widget repository and find a widget that provides a functionality similar to Figure 4 (such widget could be developed either by mentors itself or by some other developers). URI for this widget is retrieved and added into the list of URIs for "view as ..." button in "Basic algorithms" space. Since this widget implements the extended OpenSocial specification, it correctly processes information about space, widgets and resources inside and it displays information as illustrated in Figure 4.

The mentor is only required to know the URI for this widget and no additional implementation is required on the user's side to add this widget as a functional skin for his/her space. Thus, the mentor can easily share this functional skin with other people. In the case of a different Web platform (iGoogle for example) that implements OpenSocial specification, this functional skin can be used also. Another interesting point is that with such technique, the mentor can greatly extend the default functionality by using third-party Web applications.

Moreover, learners can add and change functional skins themselves. If, for example, a learner would like to have an additional area showing a list of people in the bottom-right area under Resources (see Figure 4), s/he can either look for another functional skin in widget repositories or develop one him/herself. Afterwards, a new widget's URI can be added to the list for "view as ..." button, and learner can simply switch between different functional skins.

These new widgets are indeed skins, since, for example, a space can have several different functional skins that would display the learning space and enable interaction in different ways. However, we add the word "functional" because it does not only change visual appearance as skins normally do but functionality might also change from one skin to another.

This technique allows users to have "functional skins" for their Web environments. It provides both a way to change visual representation of data and actions to be performed with this data. One of this approach's main benefits is that user can easily change the provided default functionalities to work with data. This allows easier code reuse since the same functional skin can be used by different people for their own Web environments. People themselves can find and add new functional skins to work with data. Functional skins can be used in different Web platforms, by providing users with the flexibility in choosing a Web platform, in which they prefer to work.

VII. Implementation

Graaasp is a social Web platform with support for widgets and the proposed space concepts (Figure 5). Every user has a list of connected people (friends), list of widgets and list of spaces s/he is a member of. Every space has a list of members, a list of sub-spaces and a list of widgets connected to this space.

Apache Shindig is used to provide spaces extension for OpenSocial. However space extensions are not implemented exactly as described in Table I to preserve compatibility with current OpenSocial specifications. Instead of using "type" parameter, we added a prefix "s_" for spaces and "p_" for people into the "guid" parameter. Then on the server side we take prefix into account and return either a list of items for a space or a list of items for a person.

Graaasp's OpenSocial API extended with spaces is used in another Web platform, namely, *Rolespace* [3]. *Rolespace* implements a visual interface that is completely different from *Graaasp* user interface, however it uses the same data structure, since it takes spaces, people, widgets from *Graaasp* via OpenSocial API. The *Rolespace* can be seen as a functional skin for *Graaasp*'s data structure.

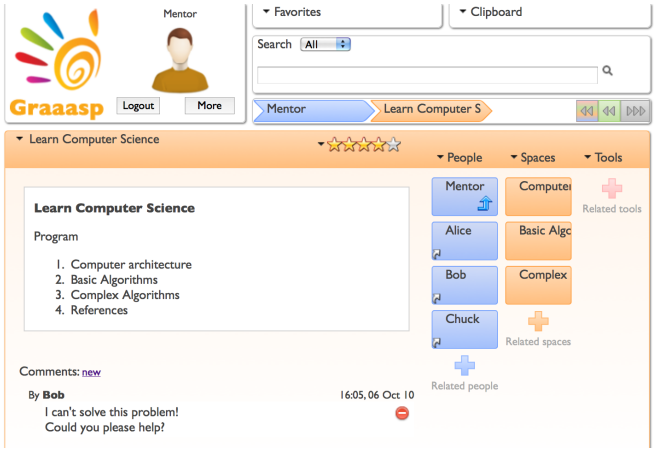

Figure 5. *Graaasp* Web platform

In addition, we developed for *Graaasp* a proof-of-concept implementation for meta-widget that integrates several small widgets. This meta-widget can run on different Web platforms that support OpenSocial widgets.

VIII. Conclusion

This paper presented contextual space as an extension to the OpenSocial specification. It summarizes the limitations of current version of the OpenSocial specification and highlights its user-centric focus without the ability to describe user's context. OpenSocial specification extended with spaces tackles these limitations and provides greater flexibility to users.

These OpenSocial extensions are described in details and three scenarios are presented. As a result, widgets can be contextualized, meaning that they have access to users' context and might adapt themselves to it.

The meta-widget concept combined with the space extension makes it possible to take advantage of functional skins (different representations and modes of interaction with the same data) and ensures portability for Web environments (same representation and mode of interaction with different data).

We conclude the paper with the details of our proof-of-concept implementation for the space extension and meta-widget. Further work is under way to provide a complete implementation for the proposed space extension that will lead to contextualized widgets, functional skins and portable Web environments.

Acknowledgment

The research work described in this paper is partially funded through the ROLE Integrated Project; part of the Seventh Framework Program for Research and Technological Development (FP7) of the European Union in Information and Communication Technologies.

References

[1] Apache shindig. http://shindig.apache.org/index.html 13.12.2010.

[2] Apache shindig wiki. https://cwiki.apache.org/confluence/display/SHINDIG/Index 13.12.2010.

[3] Rolespace. http://graaasp.epfl.ch/role_project 13.12.2010.

[4] W3C set of standards for widgets. http://www.w3.org/2008/webapps/wiki/WidgetSpecs 13.12.2010.

[5] Opensocial specification v0.9. apr. 2009. http://www.opensocial.org/Technical-Resources/opensocial-spec-v09/OpenSocial-Specification.html 13.12.2010.

[6] M. Blattner, E. Glinert, J. Jorge, and G. Ormsby. Metawidgets: towards a theory of multimodal interface design. *Computer Software and Applications Conference, 1992. COMPSAC '92. Proceedings., Sixteenth Annual International*, pages 115 –120, sep. 1992.

[7] E. Bogdanov, S. El Helou, D. Gillet, C. Salzmann, and S. Sire. Graaasp: a web 2.0 research platform for contextual recommendation with aggregated data. *CHI*, pages 3523–3528, 2010.

[8] E. Bogdanov, C. Salzmann, S. El Helou, and D. Gillet. Social Software Modeling and Mashup based on Actors, Activities and Assets. *ECTEL - MUPPLE Workshop*, 2008.

[9] A. K. Dey, G. D. Abowd, and D. Salber. A conceptual framework and a toolkit for supporting the rapid prototyping of context-aware applications. *Hum.-Comput. Interact.*, 16(2):97–166, 2001.

[10] K. Fruhmann, A. Nussbaumer, and D. Albert. A psychopedagogical framework for self-regulated learning in a responsive open learning environment. *International Conference eLearning Baltics Science (eLBa Science 2010)*.

[11] S. E. Helou, N. Li, and D. Gillet. The 3a interaction model: Towards bridging the gap between formal and informal learning. *ACHI*, pages 179–184, 2010.

[12] R. Lopes, H. Akkan, W. Claycomb, and D. Shin. An opensocial extension for enabling user-controlled persona in online social networks. *4th International Workshop on Trusted Collaboration (TrustCol 09 - in conjunction with CollaborateCom 09)*, pages 1–5, nov. 2009.

[13] M. Palmér, S. Sire, E. Bogdanov, F. Wild, and D. Gillet. Introducing qualitative dimensions to analyse the usefulness of Web 2.0 platforms as PLEs. *IJTEL*, 2010.

[14] S. Sire, E. Bogdanov, M. Palmér, and D. Gillet. Towards Collaborative Portable Web Spaces. *ECTEL - MUPPLE Workshop*, 2009.

[15] M. Wolpers, J. Najjar, K. Verbert, and E. Duval. Tracking actual usage: the attention metadata approach. *International Journal Educational Technology and Society 11 (2007) In*, pages 1176–3647, 2007.