OpenSocialGov: A Web 2.0 Environment for Governmental E-Service Delivery

Alexandros Dais, Mara Nikolaidou, Dimosthenis Anagnstopoulos

Department of Informatics and Telematics
Harokopio University of Athens
70, El. Venizelou Str, 17671, Athens, Greece
{adais, mara, dimosthe}@hua.gr

Abstract. To achieve transformational government (T-Gov), the interaction between citizens and government should be explored under a new perspective enhancing citizens' active participation. Web 2.0 paradigm could support many of the key features of such T-Gov interaction model. In this paper we explore the potential to utilize OpenSocial API to build *OpenSocialGov*, a set of libraries to construct Web 2.0 environments for the delivery of governmental e-services. In practice, OpenSocialGov enables citizens, businesses and public agencies to participate in a "governmental network", similar to a social network, which facilitates them to establish different kinds of relationships between them, as in the real world, and collaborate to perform specific tasks, composed by simple e-services developed by public agencies or even third party entities. Basic OpenSocialGov features and implementation issues, as the extension of OpenSocial API to meet the requirements of the proposed T-Gov interaction model are also discussed.

Keywords: Transformational Government (T-Gov), Web 2.0, Cross-organization Service Composition and Delivery, OpenSocial API

1 Introduction

The transformation of the IT governmental services (T-Government or T-Gov) has become an emergent challenge, posing new requirements to governmental e-service delivery to citizens and businesses, public agencies co-operation and shared e-service culture [1]. To achieve T-Gov, the interaction between citizens, businesses and government should be explored under the web 2.0 perspective [2] which emphasizes on the active participation and collaboration of all of them. Such a model should take into account all types of government interaction (C2G, B2G, G2G) and the notion of intermediates acting on citizen's behalf. Web 2.0 environment can affect government to citizen interaction and electronic government strategies, accelerating government transformation [3], [4]

In [5] we suggested the adoption of Web 2.0 technology model for the provision of governmental e-services to enhance citizen-centric service delivery and improve citizen collaboration. The proposed model handles citizen's data from a different perspective. Most government processes and services are based on the assumption

that government agencies own all information about their constituents and they are the only data steward to ensure compliance with data protection laws. In this dominant model, the citizen has limited or no access to the data that he/she is involved. Following the Web 2.0 concepts, citizens can hold their social data in profiles owed and managed by their own, thus retaining control of who can or can not access their data. In other words, citizens provide the content to the model and allow or forbid any governmental interaction.

The corresponding interaction model takes into consideration any kind of relationship between the constituents of T-Gov, namely citizens, business, government agencies and any type of intermediates, and allow them to communicate in a simple fashion, similar to one they are used to in real life. Likewise, Government-to-Government interaction necessary to perform a complex task is accomplished through multiple Citizen-to-Government and/or Business to Government interactions. Business-to-Government interaction is also citizen-centric, since properly authorized Citizens administer Business profiles and interact with Government Agencies to accomplish business' tasks. Such a perspective places the citizen in the centre of the interaction model leading to a transformation from a provider-centric (multiple citizens are related to a public agent) to a citizen-centric (multiple public agents are related to a citizen) context.

Existing social networking platforms could support many of the key features of the proposed model. Firstly, the relations mechanism of social networks could be used to implement the concept of the intermediate citizens. Additionally, citizens may execute applications provided by public agencies or third party entities in a collaborative fashion.

In this paper we explore the potential to utilizing existing social network development environments to build *OpenSocialGov*, a set of libraries to construct Web 2.0 environments to support the suggested interaction model for T-Gov services. OpenSocialGov libraries are based upon OpenSocial [6], developed by Google, as a common API for social applications across multiple websites. OpenSocial aims at becoming a "de-facto" standard for social network application development. Utilizing OpenSocial API, Public Agencies may create applications using standard JavaScript and HTML, that can be executed within any "governmental network", distributed in every administrative level (European, Federal, Local). For example, an application created by the Greek Ministry of Economics may be loaded in a Spanish "governmental network" and vice versa.

Though, OpenSocial API should the extended to meet the requirements of our T-Gov interaction model. More specifically, the following issues should be explored:

- Profile and Relationship management for all participants (citizens, businesses and public agencies)
- Governmental Service integration in the Web 2.0 interaction platform
- A Registry for the Governmental Service Discovery

The rest of the paper is structured as follows: Section 2 provides a brief introduction regarding OpenSocial features. OpenSocialGov basic features and provided services reside in section 3, along with the necessary extensions of OpenSocial API in order to implement it. Conclusions and future work are discussed in section 4.

2 OpenSocial Scope and Features

OpenSocial is a set of APIs for implementing interoperable social networks. These social networks, known as OpenSocial containers, allow OpenSocial Gadgets to access information stored within the social platform. Applications are built using OpenSocial APIs, which expose methods for accessing social data (e.g. information about people, groups of people and their friends), application data (e.g information created by Gadgets executed within a user profile, for example High scores of a Gadget) and activities (e.g. a list of activities that the user has recently accomplished), within the context of a container. The same OpenSocial Gadget can run on more than just one container, e.g. social network platform as iGoogle [7] or MySpace [8]. The latest version of OpenSocial API is 1.1.

Depending on usage requirements and programming style, OpenSocial Gadgets may be implemented as a *Social Mashup*, which is a lightweight OpenSocial Gadget executed within the social network container or a *Social Application*, executed within the social network container but relying on an external server for processing and rendering data. In any case, OpenSocial allows secure API authorization via OAuth protocol [9]. OAuth defines a method of signing requests so that a recipient can verify that the request was not tampered with while in transit. OpenSocial uses this approach to allow third party developers to verify that social data passed to their servers was sent from a specific container. So it provides a secure way to pass social data from an OpenSocial Gadget to an external server and vice versa, which is a crucial feature when implementing governmental e-services.

Perhaps the most fundamental aspect of OpenSocial API is *Data Requests*, since it handles the information exchange between Containers, Gadgets, external servers and users. They can be categorized in the following groups, all of which are utilized in OpenSocialGov implementation:

- People Requests handling social data (such as *FetchPerson* and *FetchPeople* returning information for a single individual and a group of individuals respectively)
- Application Data Requests handling application data (such as *UpdatePersonAppDataReques*t, or *FetchPersonRequest* storing and retrieving application data respectively)
- Activities Requests handling activities data (including *FetchActivitiesRequest* that returns a collection of activity objects previously generated)

3 *OpenSocialGov:* Features, Provided Services and Implementation Issues

OpenSocialGov is a set of libraries to build a Web 2.0 environment utilizing governmental e-service delivery, implemented as an OpenSocial container. In practice, *OpenSocialGov*, enables citizens, businesses and public agencies to participate in a “governmental network”, similar to a social network, which facilitates them to establish different kinds of relationships between them, as in the real world,

and collaborate to perform specific tasks, composed by simple e-services developed by public agencies or even third party entities.

3.1. OpenSocialGov Basic Features

Government, Citizens and Businesses are considered the main constituents of a T-Gov system. The Citizen Centric Interaction Model that is proposed is tagged as Web 2.0 as the citizen retains all the control of the information that is exchanged in the model. Conceptually, Citizens provide the content to the model, using the necessary Gadgets called applications. Overall, it is up to them to decide whether or not to install an application that can access their private data.

The interaction model between citizens, businesses and public agencies supported by OpenSocial and the relationships established between them is summarized in figure 1.

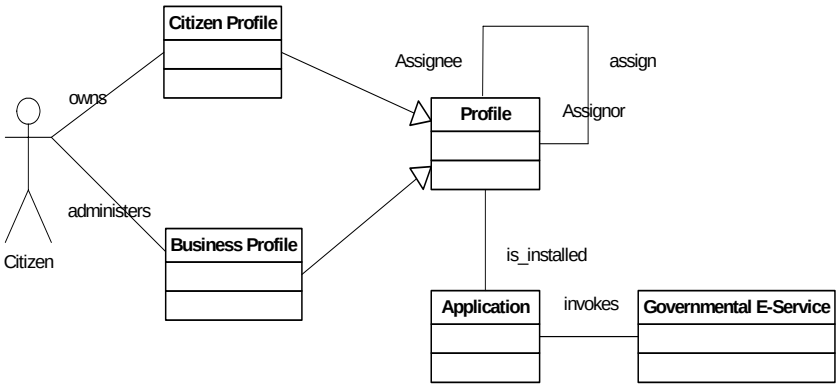

Fig 1. OpenSocialGov Interaction Model

Citizens and businesses are represented by profiles, while a business profile must always be administered by an authorized citizen, as in the real world. Thus, a single sign-on is used by the citizen when logging in a OpenSocialGov container, enabling him/her to access his/her citizen profile and all the business profiles he/she administers. Applications, developed by Government Agencies or third party entities, are installed and executed in Citizen and Business profiles. They invoke web-services implementing specific governmental services, executed in Government Agency IT infrastructure (outside OpenSocialGov container). Citizens are able to install applications found in the Application Registry. Consequently, applications publish information in their citizens private profile. Cross organizational tasks can be accomplished by installing several applications from different Public Organizations in a profile and sharing the information that resides into the profile (application data section) among them. This is accomplished with the pub/sub mechanism that provides operations for publishing and subscribing to message channels. [10]

When dealing with the Government, often both businesses and citizens use intermediates. Intermediates are Citizens or Business that accomplish tasks on behalf of other Citizens or Business. The assignee can install and execute applications on

behalf of the assignor. The application appears on both profiles, with the published information appearing on both the assignor and the assignee profile. The assignor should retain the control of the actions of the assignee. Overall, the Assignor Profile is the reference point for the coordination, the information extraction and the control of the applications.

This is depicted by the *assign* relationship, established among citizens and businesses. The *assignee* can install and execute applications on behalf of the *assignor*. The application should appear on both profiles with a difference in the title bar; in the assignor profile the application declares the name of the assignor and in the Assignor Profile the application declares the name of the assignee. The Assignor Profile is the reference point for the coordination, the information extraction and the control of the applications.

The assignor authorizes the assignee to do *something* for *some period.* The citizen/business declares a domain for the tasks that the intermediate can handle. The other parameter is the period of the authorization. An authorization can be granted permanently or for a specific task to be completed. The fundamental notions of the model regarding Business to Government are presented in [11]

3.2. Provided Services and Implementation Issues

A) Any OpenSocialGov container utilizes existing major OpenSocial features such as:

- *Handling profiles and relationships*. Basic OpenSocial operations support adding and removing relationships (as the “friend” relationship supported by most social networks) and adding/ removing applications in profiles.
- *Data Persistence*. OpenSocial defines a data store that applications can use to read and write per-user and per-application data, otherwise known as "app data”, utilized by *OpenSocialGov* to also handle citizens and businesses (either assignor or assignee) application data.
- *OAuth support.* Protecting Citizen’s data is of vital importance. To prevent unauthorized use of citizens’ data, OpenSocialGov container uses OAuth parameter signing to create the signatures on requests. OAuth client libraries can be used to verify the requests received from the container.

B) To implement OpenSocialGov containers, OpenSocial API is extended to support the following features:

An extended relationship mechanism facilitating the “assign” relationship.

This relationship is established between the assignor and the assignee citizen or business. The *assign* relationship is treated as a specialization of the *friend* relationship. In a *friend* relationship, connected friends have access to each other’s basic social data (including name, surname, list of friends). The *assign* relationship is more complicated. In contrast to the *friend* relationship that either exists or not, the *assign* one is characterized by a set of properties, such as the type (if it is permanent or not), the duration (if it will be dropped when a time period will be elapsed or not) and the domain it is valid for (e.g. corresponding Public Agencies or even specific applications). Assignees can have access and use parts of the application data of the

assignor they are authorized to represent, based on the domain this relationship is valid.

From the technical point of view, Applications installed by the assignee should be executed with the assignor data. So a mechanism, sharing the assignor data among assignees should be implemented. Furthermore, applications can be integrated in both the assignor and the assignee profile with different title bars. This way; the application declares the name of the assignor in the assignee profile and the name of the assignee in the assignor profile. Such feature can be implemented by creating groups of assignees (similar to the default *friends* group*)* that can access information stored in the assignor profiles.

Citizens' Interaction

Each citizen, either handing his/her profile or administrating a business one, and corresponding assignees should be informed of each other's activity. This is accomplished utilizing an extension of OpenSocial ActivityStream library [4]. Activity Streams provide an intuitive, sequential log of the activities of each user and others related to him/her. Basic operations supported in OpenSocial include storing activities and retrieving activity streams for himself and friends. OpenSocialGov container extends activity stream to provide an efficient way for the assignees to interact. Besides informing citizens regarding pending assignments and assignment acceptance, the same mechanism is used to inform each citizen for all the applications assignees are executing on his/her behalf and pending results. When an application is unable to find the necessary application data for its execution, a thread-activity named by the application is created and spread to assignees. The assignees responsible for to producing the application data needed (e.g. execute the appropriate applications) are informed and may participate in the overall process. Every action taken to enable the execution of the initial application, including a) the applications executed and b) the application data (pending or completed) loaded to the citizen/business profile, is added in the corresponding activity list.

OpenSocialGov Application Development and Deployment

Application development is based on Gadgets. Gadgets are autonomous software components, based on HTML, CSS, and JavaScript,. not aware of the existence of the other Gadgets. Inter-Gadgets interaction is be facilitated in the OpenSocialGov container. In the latest version of OpenSocial (1.1), Inter-Gadget Communication was introduced through OpenAjax Hub event management system [10]. By incorporating this technology into OpenSocialGov Container, application data management mechanism provides a publish and subscribe feature enabling applications to subscribe to their input data and publish their output data. Thus, the execution of a specific application is enabled when all necessary input data are available.

Assuming that an application, called *electronic offer submission*, needs data provided by *tax-clearance* and *insurance clearance* applications to start its execution. When invoked, it subscribes to its input data before actually executed. The corresponding thread-activity is created, if the subscripted application data are not available. Tax and insurance clearance application data are published by *tax-clearance* and *insurance clearance* applications, when executed by corresponding assignees (if any), enabling the execution of *submit a proposal* application.

Technically speaking, this feature can be implemented using the `<Require>` tag, included in gadget's header, which declares dependencies. For example, the following declaration is included in *submit a proposal* gadget.

```
<ModulePrefs title="electronic offer submission”>
          <Require feature="opensocial-1.1" />
          <Require feature="tax-clearance" />
          <Require feature="insurance-clearance" />
</ModulePrefs>
```

Tax-clearance and *insurance-clearance* are input data labels found in the application registry, discussed in the following.

C) Any OpenSocialGov container should also incorporate the following services, implemented as external applications:

Application Registry

The registry facilitates application discovery. When registering an application, the developer should describe it using metadata fields, necessary to identify the application. The basic metadata needed to describe applications [12] are summarized in Table 1.

Table 1: Metadata characterizing Applications

Field	Description
Identifier	An unambiguous reference to the Application.
Title	A name given to the application. It should be the formal title of the service provided.
Version	Applications can be modified and changes from one version to another must be tracked.
Creator	A Public Agent or third party organization or even a citizen primarily responsible for making the Application.
Type	The genre of the service offered by the application (e.g payments, certificates).
Description	A high level description of desired operation expressed in natural terms.
Input Data	An application may require some input data, either citizen's personal data or data produced by other applications.
Output Data	The output of the application described in terms of data produced by its execution.

When trying to accomplish a complex task, applications are coordinated based on their input and output data described in the Registry. Though, applications are registered by different organizations, thus the compatibility of terms used to describe application data should be ensured. Semantic interoperability between different organization vocabularies is based on a hybrid system, which involves a *predefined taxonomy* maintained by public agencies and a *folksonomy* formed by citizens using the applications. The creator of the application lists the input and output data from a

vocabulary, which has been previously populated from public agencies, while citizens may tag input and output data using their own terms.

The taxonomy embedded into the Application Registry should provide a standard list of meaningful terms, which may be used to describe application data, and publish the preferred terms in order to accomplish semantic interoperability between Public Agencies at any level (for example European, Federal, Local). It can be implemented collaboratively by the Public Agencies involved. Generally, Public Administration is separated in domains (such as Business and Industry, Economics and Finance, Environment, etc) reflecting the social needs. A generic hierarchical structure can be applied, based on established public sectors vocabularies, such as UK IPSV (Integrated Public Sector Vocabulary) [13]. Each public agency proposes a set of formal terms that describe procedures involving citizens (e.g Value Added Tax clearance certificate), which may be used to describe application input and output data.

The registry should also facilitate citizens to express themselves regarding the applications they use, through the Citizens Collaborative Tagging System. As a application is integrated in the profiles, Citizens are asked to describe the application freely with their own terms. In contrast to the Taxonomy, the Citizens Collaborative Tagging System has no hierarchical structure. Free tagging allows citizen to use non-preferred terms for the Public Agency applications, such as synonyms (for example equivalent terms in different languages) or acronyms – abbreviations (such as VAT clearance for Value Added Tax clearance).

Recommendation mechanism

The Recommendation mechanism is invoked by both the OpenSocialGov Container and the Application Registry. Recommendation mechanism may assist citizens to find and add in the proper profile the necessary applications for a cross-organization task to be accomplished. When a citizens installs an application in a profile, the application searches the corresponding profile (either citizen or business, either current or the assignor's if it is executed by an assignee) for the necessary input in order to be executed. If the required input data is missing, the recommendation mechanism may propose some applications that could fetch that data.

The recommendation mechanism may also be used when registering an Application to assist the developer to effectively tag input and output data using both the taxonomy defined by the government and the folksonomy formulated by citizens.

3.3. OpenSocialGov Architecture

OpenSocialGov architecture is illustrated in figure 2. Any *OpenSocialGov* container, providing extended OpenSocial functionality, may be used to implement a "governmental network". Registered users, e.g. citizens handling their personal or business profiles, as owners or intermediates, may install OpenSocialGov applications in their profile and invoke them through a web browser. Each time an application is invoked the corresponding OpenSocialGov API is used to pass related information in the *OpenSocialGov* container through a JSON-RPC call.

OpenSocialGov container allows communication with External Govermental Services via OAuth protocol. External Governmental Services are running on their IT infrastructure and invoked as web services based on the concepts of Web Oriented Architectures (WOA) [14], though designated application proxies.

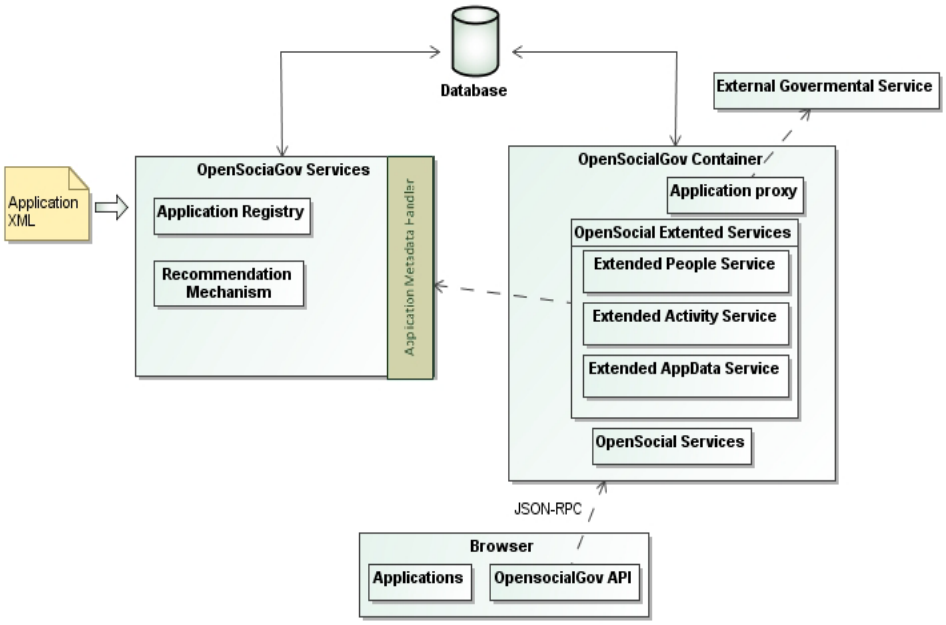

Fig. 2. OpenSocialGov Architecture

Any *OpenSocialGov* container consists of both regular and extended OpenSocial Services. Regular OpenSocial Services, as for example *OAuth support* or *People and Groups Services*, provide the main features of a social networking platform. They utilized to implement basic social network functionality, as profile management and external application invocation.

OpenSocialGov supports the concept of the intermediates citizens and collaborative composite T-gov service composition, thus OpenSocial's PeopleService, AppDataService and ActivityService have been extended. As previously discussed, the extended PeopleService practically handles the "assign" relationship, established between the assignor and the assignee profiles. The extended AppDataService extends the pub/sub methods that allow Applications to interact even thought they are installed in different profiles and specifically the assignor and the assignee ones. The extended ActivityService provides the necessary methods for assignors and assignees to communicate between them, in order to collaboratively accomplish cross-organizational tasks.

OpenSocialGov Services supplement OpenSocialGov container to provide integrated collaboration services. They consists of the Application Registry and the Recommendation mechanism. They both utilize *Application Metadata Handler* to manage the *predefined taxonomy* maintained by public agencies and the *folksonomy*

formed by citizens using the applications. The Application Registry serves as a repository for the Application XML files to upload.

3.4. A Simple Example

A simple example depicting the potential of the proposed interaction model as far as cross-organizational interaction is concerned is considered in the following. A citizen called Jason must receive a payment by a Public Agency. Thus, tax and insurance clearance is required.

Based on the "whole of Government approach" [15], an one-stop portal could provide such service. One-stop government portals gather governmental services and provide a common interface for the citizen to interact with all of them. In this case, Jason, properly authorized, would invoke the "Payment from Public Agency A" service, which would invoke both the "Tax clearance" service provided by Ministry of Finance and the "Insurance Clearance" service provided by the Ministry of Labor according to a predefined plan, store and process their outputs and inform the citizen whether the payment was feasible. In such case, the data exchanged between the services is known in advanced and statically described within the predefined plan, implemented using a standard web service composition language as BPEL. The services invoked are specifically defined, while the composite service is treated as a "black box" invoked by a single user, who has no notion of his/her personal information exchanged between Public Agencies (for example Social Identification Number) and is not asked for his/her consent.

Based on the proposed approach, utilizing an OpenSocialGov container, Jason may install in his profile the "Payment from Public Agency A" application from the Application Registry. As indicated by its inputs, the application requires both tax clearance and insurance clearance data to exist in his profile and prompts Jason to install the relevant applications, utilizing the Recommendation mechanism. More than one application recorded in Application registry may produce such data. Lets assume that "Tax Clearance" Application is provided by the ministry of Finance and "Insurance Clearance" application is provided by the ministry of Labor. Jason installs both applications that provide the necessary input for the "Payment from Public Agency" to be executed, thus is inform of the way Public Agencies interact to fulfill his request, while relevant data are gathered in his profile .

From the Public Agency point of view, OpenSocialGov platform provides a common framework for the application to be integrated avoiding complex interaction schemas, leaving the cross-organizational interaction to the citizens. Application could be developed and integrated in the platform at any level (for example European, Federal, Local) in a distributed way without predefined export/import data structures. Citizens, on the other hand, are provided with a relatively simple [16] interaction model, are equipped with an assignor-assignee mechanism and with a personal data vault that handles their private data in a Web 2.0 fashion.

4 Conclusions and Future Work

A Web 2.0 environment utilizing governmental e-service delivery has been presented. The novelty of the proposed approach relies on the establishment of a "governmental network" between Citizens, Businesses and Public Agencies, utilizing social networking concepts. The collaboration between them is facilitated in a way similar as in the real world, by applications executed in citizen and business profiles, while the assignment of specific responsibilities to intermediates is also supported. OpenSocialGov container, based on OpenSocial API, is proposed for implementation of the Web 2.0 environment. The way OpenSocial API should be extended for this purpose is also discussed, to establish the viability of the proposed interaction model.

We are currently working on the implementation of OpenSocialGov container, while OpenSocial API extensions have already been implemented. Future work focuses on the completion of a prototype implementation of OpenSocialGov and the exploration of a real-world test case to evaluate the proposed concepts in practice. In addition, security and trust issues must be thoroughly examined.

References

1. S. Sahraoui., A. Ghoneim., Z. Irani., S. Ozkan. T-Government for benefit realisation: A research agenda. In *Evaluating information Systems. Public and Private Sector* by Zahir I and Peter L. (eds), pp 359-373, Butterworth-Heinemann/Elsevier,*(* 2008)
2. T. Oreilly, What is Web 2.0: Design Patterns and Business Models for the Next Generation of Software. Communications & Strategies, No. 1, p. 17, (2007)
3. A. Maio, Government and Web 2.0: The Emerging Midoffice, *Gartner*, (2007)
4. A. Maio, The E-Government Hype Cycle Meets Web 2.0, Gartner, (2007)
5. A. Dais, M. Nikolaidou, N. Alexopoulou, D. Anagnostopoulos Introducing a Public Agency Networking Platform towards supporting Connected Governance, 7th Int'l Conf (EGOV 2008), LNCS 5184, Springer, pp 375-387. (2008)
6. http://code.google.com/apis/opensocial/, Accessed 22/03/2011
7. http://www.google.gr/ig, Accessed 22/03/2011
8. http://www.myspace.com/ Accessed 22/03/2011
9. http://oauth.net/, Accessed 22/03/201
10. http://www.openajax.org/member/wiki/OpenAjax_Hub_1.0_Specification_PublishSubscribe, Accessed 22/03/2011
11. A. Dais, M. Nikolaidou, D.Anagnostopoulos Facilitating Business to Goverment Interaction using a Citizen-centric Web 2.0 Model, IFIP 305 (I3E 2009), Springer Verlag. (2009)
12. European Interoperability Framework for European Public Services (EIF) ver. 2.0, (2010)
13. http://doc.esd.org.uk/IPSV/2.00.html, Accessed 22/03/2011
14. D. Sholler, Reference Architecture for Web-Oriented Architecture, *Gartner*, (2008)
15. T. Christensen, P. Lægreid, The Whole-of-Government Approach to Public Sector Reform, In *Public Administration Review*, Wiley, Vol. 67 Issue 6, pp. 1059 – 1066.
16. J. Maeda, The Laws of Simplicity (Simplicity: Design, Technology, Business, Life), The MIT Press, pp. 73-81, (2006)