TM: A Development Technique for E-government 2.0 Portals

FARAS ZUHEIR MUSTAFA MOHAMED[1], SARAVANAN S/O MUTHAIYAH[2], ARMAN K. NASSIRTOUSSI[3]
Faculty of Information Technology
Multimedia University
Persiaran Multimedia, 63100 Cyberjaya, Selangor
MALAYSIA
[1]faras@mmu.edu.my, [2]saravanan.muthaiyah@mmu.edu.my, [3]armankhnt@gmail.com

Abstract: - The purpose of this paper is to introduce the software community to TM (Total Mashup): a conceptual software development technique for building and developing e-government 2.0 portals as total mashups. The range of current development approaches and technologies are compared and a particular conceptual development technique with a specific set of technologies is proposed to be instrumental to the development process of e-government 2.0 portals. TM is tested via a real-life prototype that targets the government research community. The outputs of the prototype are discussed and analyzed. TM has various added advantages to the conventional development processes in terms of time efficiency and collaboration. It is a unique technique to enhancing the e-government 2.0 portals development process.

Key-Words: - E-government 2.0, Web 2.0, portal, mashup, Java, and API.

1 Introduction

Generally speaking, e-government initiatives have plateaued over the past the few years. This can be attributed to ineffective governance, lack of Web-related capabilities, and reluctance to allow users' participation in the creation of applications and content [1].
E-government 2.0 or government 2.0 is a legitimate child of web 2.0 influence on numerous domains including e-government or e-government 1.0. Siblings of e-government 2.0 include enterprise 2.0, e-library 2.0, etc.
E-government 2.0 portal development shares more complex features than any current web 2.0 project. This is due to the versatile nature of the many parties constituting the term (fig.1). Web 2.0 concepts can be applied in many segments of e-government – in both back- and front office domains [2].
It is predicted that Web 2.0 will enable the transformation of public administration services, development of better policies, suppression of silos and reorganization of public administration [3].
With the fast pace of development on the web, it has become commonplace more than ever before. Web 2.0 along with its social aspects and its many capabilities, which were simply not possible a few years ago, are ruling the cyberspace nowadays [4]. As a result an SOA (Service Oriented Architecture) approach has become popular under the umbrella of SaaS (Software as a Service), which has increasing possibilities in terms of available services usage on the Internet for customized purposes.

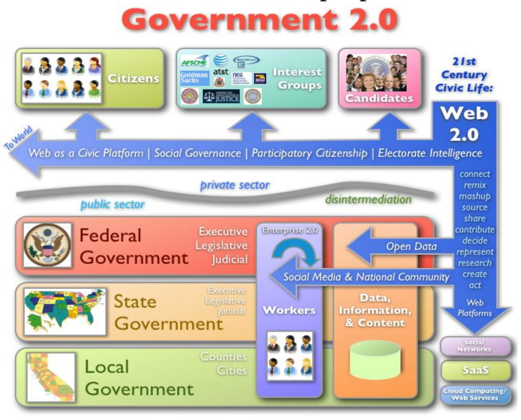


Fig.1: An e-government 2.0 (government 2.0) blueprint [5]

This is reflected in the increase in the popularity of mashup applications. It is more evident with the presence of ubiquitously useful services like Google earth that is mixed up in various mashups from real estate websites to traffic control [6].
According to Aaron Boodman, quoted in BusinessWeek online, "The Web was originally designed to be mashed up. The technology is finally growing up and making it possible." The mashup concept is centered around the integration of different sources within one displayed interface,

usually using the development approach of AJAX or Asynchronous JavaScript and XML.

The terms “portal” and “mashup” have been demarcated from each other (based on the standards they embrace), when discussing technologies for web development, even though both of them are classified under the concept of content aggregation. Despite all of that, to many researchers, the term “portal” is widely and conceptually understood as a common entry to into multiple distributed repositories, using the analogy of a door as a common entry into knowledge resources. A portal provides a common user interface, which can often be customized to certain preferences [8].

TM’s conceptual understanding of a mashup refers to a web component that achieves functionality by combining two or more external sources so as to create a new service. Hence, “total mashup” refers the intensive use of mashups and web services in a substantial and undecorated way.

Researchers expect that Web 2.0 influence will cause a shift from service-oriented to web-oriented technologies like web services and mashups, due to its data reuse features [7].

With the great availability of services, maybe it is about time to use mashups more aggressively and for total production of a range of common web sites and portals. A total mashup software development technique needs to be embraced by software architects from scratch; this will revolutionize how web sites and portals are made in governments and enterprises.

Several research questions are posed; for instance, what is the logical process or technique followed to produce intensively mashup-based e-government 2.0 portals? And how easy would the use of mashups be in the development process? This implicates factors like the needed knowledge by the developers, the availability of resources, the possibility of integration and components reuse within the popular platforms like Java and .Net, and which platforms are most-friendly and most-efficient for development.

Generally, there are four research objectives of this work:

(1) The introduction of a total-mashup conceptual development technique for e-government 2.0 portals that is pragmatic and easy to understand.

(2) Explaining the deficiencies of the current development techniques and the strength points of TM.

(3) Exploring the possibility of easing the TM technique components reuse process.

(4) Evaluating the conceptual nature of the TM technique via a test portal that carries much of the genes of a typical e-government 2.0 portal. This is very important in extrapolating the concepts introduced.

The structure of the paper follows both the conceptual and logical progression of the TM technique. It incepts by discussing the different features, elements, and current development mindset influencing the domain of e-government 2.0 portals. The paper discusses the potential influence of mashups on the society in general and on e-government in specific. It discusses several points of similarities and differences with enterprise 2.0 and SOA respectively. It then tackles where would the mashup be classified among web services and recommends a particular development platform for the TM technique after a comparative survey on their relevant characteristics.

The conceptual nature of TM is put to the test by an implementation of a researchers portal, whose members are commissioned to undergo research for the government. Findings and limitations are used to critique the overall work and draw directions to future research.

2 Literature Review

There are a number of common components and functions that are usually implemented in every e-government 2.0 portal; for instance the following features are seen frequently: calendar, chat (messaging), communication platforms/weblog, bulletin/announcement board, file storage and management, reference management, wikis, scheduling, project management and graphical reporting.

More, a need for functionalities of the following kinds is emerging; mostly categorized as the Web 2.0 trends [9]: social networking, collaboration, constant communication, collaborative content development, and collective knowledge representation.

It is important to note that many of the above popular features exist also on public websites like Google, Delicious, Blogger, Box.net, etc. Many of the potential users of any portal that are to be created, are already enjoying these services elsewhere and in a separate setting and environment than the newly created e-government 2.0 portal.

According to W3C, semantic Web standards in particular lend themselves to data aggregation — mashups — and thus to collaboration (planned and unplanned) among government agencies and with other e-government actors. Semantic Web technology also helps in the management of accountability, which can help reduce errors and mistakes and build trust [10].

2.1 The Current Development Mindset

Currently the de facto development mentality is still designing and creating the necessary components and features from scratch. Or maybe with the use of some pre-made libraries but usually the components developed inside a portal are of little or no contact with the outside world and specifically focused on the users' affairs inside the company. They are simply to be found only on their portal of origin [11].

The major problematic parts of “reinventing the wheel” are:

(1) Time Consuming:

It takes a considerable amount of time until the under development portals reach, if ever, a level of higher functionalities like collaborative features, for instance.

(2) Organizational Resistance:

Every new format of components and every new solution encounter a degree of organizational resistance and will require time, for the users to get familiarized with.

(3) Different Designs:

If an e-government 2.0 project has different portals for different departments, which is the case a lot of the times due to the evolutionary nature of the organizational growth, the users who need to have access to different resources from different departments end up using similar features with different designs for each one. Say, in case of departmental event calendars, each department will have “one of its own” and integration of any kind can be time consuming, nevertheless inevitable.

Hence, we suggest a shift of architectural mindset from an organization level design to the Internet cloud level design and we call it the “Total Mashup” technique.

Many enterprises have recently considered including web 2.0 technologies in their enterprise portal, but that happened through third party mashup providers, like IBM [12], who probably separate themselves from the enterprise and assume different responsibilities. This paper suggests that architectural separation should not exist at least for e-government projects that are developing their portals from scratch; What the IT departments should do instead, is considering the TM techniques that already has many mashup concepts in place, uses many external services, and is designed with the mentality that accommodates creation and implementation of ad-hoc mashups any where necessary in the future.

The similarities between e-government 2.0 and enterprise 2.0 are striking. Both concepts have similar obvious objectives, but the scale and context are different. Hence, what applies on enterprise 2.0 can carefully be extrapolated to e-government 2.0.

Enterprise social software, also known as Enterprise 2.0, is a term describing social software used in enterprises [13]. It includes Web 2.0 modifications to enterprise portals.

Traditional enterprise software imposes structure prior to use, while enterprise social software tends to encourage use prior to providing structure [14].

The latter is the mentality that is required to exist nowadays; however, the shift is not much seen yet because it goes against the ingrained design habits. Nonetheless, a total mashup mindset is going to facilitate and secure many of the Web 2.0 changes that an enterprise will be forced to embrace due to the external Web 2.0 phenomenon-taking place in the society.

The Association for Information and Image Management (AIIM) defines Enterprise 2.0 as "a system of web-based technologies that provide rapid and agile collaboration, information sharing, emergence and integration capabilities in the extended enterprise" [15].

We argue that neither e-government 2.0 nor enterprise 2.0 features will be regarded as a modification and extension only, but more techniques and approaches are needed for a total integration of these characteristics in the e-government and enterprise portal fundamentally.

Some of these characteristics are as follows:

(1) Implementation of Web 2.0 technologies within an e-government/enterprise.

(2) Existence of rich Internet applications within an e-government/enterprise project.

(3) Provision of software as a service using the web as a general platform.

“All of these things (wireless, next generation search engines, weblogs, instant messaging, file sharing, grid computing, web spidering] come together into what I'm calling "the emergent Internet operating system." The facilities being pioneered by thousands of individual hackers and entrepreneurs will, without question, be integrated into a standardized platform that enables a next generation of applications. The question is, who will own that platform?”[16].

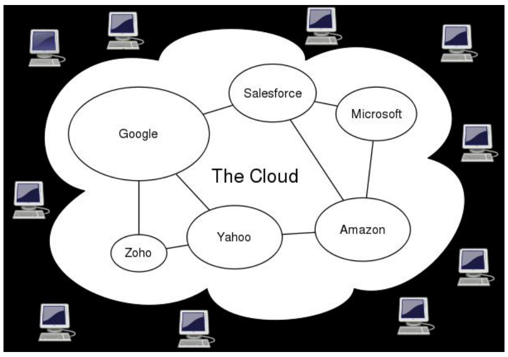


Fig. 2: The cloud, as portrayed by Sam Johnston

As Tim O'Reilly had correctly speculated in 2002, Internet has taken a big shift in its style, affairs, and usage. With Web 2.0, which was also coined by him [17], the web has become more friendly, social, and far more commonplace.
Some challenge an OS (operating system) look at Internet cloud, because they claim, it has not fully replaced an operating system [18].
We suggest that it is not a matter of an operating system physical existence or its size, but that the cloud can be named an OS as soon as the Internet platform fulfils some of the main responsibilities that an OS has. A brief account of some of these responsibilities follows:
(1) Provision of running applications on the platform, e.g., Google docs.
(2) Provisions of APIs to control processes and create further applications, e.g., all openAPIs like Google APIs.
(3) Provide file storage and its management, e.g., box.net.
(4) Provide global user management, e.g., OpenID.
As one can notice from the given examples, all of the above already exist and is becoming more and more sophisticated. Therefore, the e-government 2.0 architects should consider it seriously in their total designs. Reluctance towards use of these services prejudicially and without thorough and open-minded exploration can be of a disservice to the usability and success of future e-government portals.

2.2 Web 2.0 Mashups in Society and E-government
By far the impact of Internet on our lives has doubled if not more by the occurrence of the Web 2.0 trend and its components. Every citizen of us is somehow involved in a social networking site, or uses web services in one way or another [19]. Google Docs for instance has become an undeniable part of the lives of a great majority of the knowledge workers and the Internet users. Wikipedia has broken a definite historical record in multitudes in terms of a source that is referred to by people of all kinds constantly and around the globe.
The use of Web 2.0 technology has become so habitual by the younger generation that some companies are starting to act on it, based on the sheer belief that "If change is happening on the outside faster than on the inside the end is in sight." - Jack Welch.
We firmly believe in the above and hence encourage web 2.0 technologies to be embraced and integrated by the e-governments. By far the most practical and ubiquitous aspect of the Web 2.0 wave is the appearance of mashups that are effective and have solved many problems effectively; earthquake-forecast is one example [20].
A great number of those mashups, especially Google maps have gained a continuous presence. It turns out that data representation on a map, particularly if it is real-time is of great popularity and usefulness [21]. Hence, there are all sorts of mashups created from pandemic pin-pointers to on-the-run-prisoner locators, from traffic monitors to real-estate websites almost anywhere possible; Google maps were injected and many of them facilitated tasks, which could not have been carried out otherwise.
Like mentioned earlier, the mashup mentality is based on driving usefulness out of what already exists and creating applications or solving problems many times more quickly.
This rate of problem solving is something that people are now used to in the society and will find it frustrating if they enter for instance an e-government portal whereby the same principles and methodologies do not exist.
Concepts of Web 2.0 like social networking prove to be very useful inside e-governments, too. They can facilitate knowledge dissemination in a unique manner and can help citizens find answers for their questions from social network. Citizens and businesses can also stay connected and keep updated.
The question of whether mashups are the Excel of our era is a very valid question and a few lessons need to be learnt in terms of their similarity as user applications and their adoption process by the enterprises and the software companies.
Excel applications gave the non-developer users the possibility to create easily ad-hoc applications that solved some of their daily problems very quickly and efficiently and that is why, despite many efforts to control the user desire to do so and to restrain

things by the help of substitute and more sophisticated BI (Business Intelligence) software that is unified across the enterprise [22], the users continued to use excel applications as they preferred form of BI, until the companies were left with no choice but embracing the change and valuing it.

We argue that the same attitude exists towards mashups currently in e-government portals, but they will find their way, as they have in many cases. However, embracing the required changes in the architectural design, that enables mashups from ground up, can make this process a lot less painful and a lot more fruitful, while aimed at a productive future backed by user creativity.

3 The “Total Mashup” Technique

We coin the term “Total Mashup” or TM technique for e-government 2.0 projects, typically portals that encompass the following characteristics:

(1) Extensive use of external services, e.g., Google Calendar, anywhere possible.

(2) Inclusion of the current Web 2.0 services and trends that the potential users of the portal are using already on the outside as much as possible in cooperation with their original sources.

(3) Creation of an architecture that facilitates later accommodation of mashups and further extensions. A lot of this relies on the choice of platform and the integration approach for mashups, which is explained further in the coming sections.

(4) Implementation of procedures and input/output passages that will facilitate controlled data accessibility.

(5) Provision of appropriate security and speculation over the probable security risks.

3.1 Mashups vs. SOA

SOA (Service-Oriented Architecture) has been around for quite some time and has become with principles like abstraction, autonomy, composability, discoverability, formal contract, loose coupling, reusability, and statelessness [23].

These principles are supposed to serve to faster application development with more flexibility in terms of mixing the services. The services are also autonomous and treated as separate entities, so that if one is lagging, the others can still work fine and independently, hence, the entire system does not go down at once. In short, some of the best practices of software architecture are summarized in these principles.

Mashups however, do pretty much all the above and in a more pragmatic way, that is also sensible by the citizen. They actually realize many of the promises of SOA at a practical level and they also have many advantages. Hence, they have become radically more popular compared to SOA composites especially in the eyes of the businesses [24].

The main differentiations of mashups that should be considered to SOA:

(1) Mashups are Web-based:

SOA is not necessarily web-based and web compatible, which makes it different and somewhat more primitive compared to mashups.

(2) Mashups are relatively simple:

The SOA architecture with all its best practices is academically interesting but the complication overhead is undeniable, and makes it impractical at various levels.

(3) They can be user-developed and are sharable:

One of the most important features that came to practical existence and use, when it comes to mashups, is the fact that they are capable of being developed or composed by the users, which is the same factor that has made Excel applications extremely popular. Moreover, in an enterprise setting, once a user creates a mashup it can be easily shared with others, which makes it extremely useful and reusable.

(4) Easier data integration with other services:

Since mashups are web-based and its data collection and communication are much simpler and more standard, they are extremely successful when it comes to data integration with local services or other services from the web. This is practically not the case for SOA architecture because the access to data is more complicated and the methods are not simple or compatible with web-based methods.

(5) Distinct business value:

Since the mashups are easily and very quickly created, their business value is quite clear. They provide access to business insight that is the sheer result of putting different data sources together at a timely manner.

(6) Help emerge new business models:

Since mashups help deepen business insight and provide new sources of revenue accordingly, they lead to new business models [25]. These new business models are partially a result of the capability that mashups bring about in accessing and using data at a different granularity than ever before.

However, some look at mashups not as a replacement of SOA but as a complement [26], especially in the user arena. We argue that this architectural look is not declined and the TM technique can be described as an SOA architecture that has a fully-fledged web-based end and is fully oriented and specialized for mashups' creation and management.

3.2 TM Technique is RESTful

A Web Service is defined by W3C as "a software system designed to support interoperable machine-to-machine interaction over a network" [27]; and that usually happens through APIs. There are other approaches with similar functionality as web services like Object Management Group's (OMG), Common Object Request Broker Architecture (CORBA), Microsoft's Distributed Component Object Model (DCOM) or SUN's Java/Remote Method Invocation (RMI), however, Web Services are most compatible with current web standards.

The three prevalent styles of Web Services are Remote Procedure Calls (RPC), Service-Oriented Architecture (SOA), and Representational State Transfer (REST).

RPC is operation-based, SOA is message-based, but REST is resource-based.

The suggested approach for the total mashup technique is RESTful.

The REST design transmits resources over HTTP without an additional messaging layer, as opposed to SOAP. As a matter of fact the best example for a RESTful system is the World Wide Web.

REST has got a number of benefits, namely, improved response time by caching of representations, less client-side software requirement, less dependant on vendor software, no need for a separate resource discovery mechanism and most importantly REST possesses unique scalability [28], which is extremely important for the context of e-government 2.0.

In the words of the inventor of REST, Roy Fielding, REST's effect on scalability is as follows [29]:

"REST's client-server separation of concerns simplifies component implementation, reduces the complexity of connector semantics, improves the effectiveness of performance tuning, and increases the scalability of pure server components. Layered system constraints allow intermediaries—proxies, gateways, and firewalls—to be introduced at various points in the communication without changing the interfaces between components, thus allowing them to assist in communication translation or improve performance via large-scale, shared caching. REST enables intermediate processing by constraining messages to be self-descriptive: interaction is stateless between requests, standard methods and media types are used to indicate semantics and exchange information, and responses explicitly indicate cacheability."

3.3 TM Development Platform

The choice of a development platform is critical to the extensibility and interoperability of an e-government 2.0 portal; both are vital requirements of its success.

There are many platforms available for such development, however, the following demonstrates why in the view of this paper, JavaServer Faces or JSF is one of the most suitable, and how it should be used.

3.3.1 Comparisons of Current E-government 2.0 Portal Development Platforms

Some of the most common technologies that are used for web development are Ruby on Rails (RoR), .Net and JavaServer Faces. Each of which has unique characteristics that make them suitable for specific usages.

Ruby on Rails for instance is a great automation of many of the configurations that developers need to carry out, it also generates a lot of the code and by the motto of "Convention over Configuration" [30], agile development becomes possible. Moreover, it is aimed for web development and, therefore, enjoys the advantages that come with specialization.

.Net is a heavy weight platform with many capabilities for huge development projects and monitoring them to the greatest details. It also enjoys the Visual Studio IDE for visual design that can facilitate the development process to a great deal.

However the two above platforms are less practical and efficient than JSF for the purpose of technique.

One of the important factors that need to be taken into consideration when contemplating a technology or platform, which is to be used at e-government level, and in total cooperation with the other segments of an e-government system, is interoperability. Not only is interoperability important with regards to systems where the e-government 2.0 portal needs to be run on, in terms of hardware and OS, but also, and probably more importantly, it is critical for an enterprise portal to be compatible and work with J2EE as, by far, the most dominant enterprise solution for many of the businesses. J2EE enjoys specialization in enterprise architecture, and therefore, is used at large scales and facilitates many of the needs of heavy-usage-environments, by appropriate and somewhat unique traffic and load management capabilities that are

achieved through possession of well-rounded properties that are required in distributed systems.
From the above perspective, JavaServer Faces, which is a component of the J2EE stack naturally, exceeds any other competing platform in terms of compatibility.
JSF applications enjoy one of the most successful design patterns for web development by the name of Model-View-Controller or MVC. MVC brings about separation of business-logic and presentation in a way that makes the applications a lot more efficient, reusable and maintainable; it also facilitates testing processes.
Moreover, JSF or J2EE, in general possess strong separation of layers of code through the use of JavaBeans. In JSF also, JavaBeans are excessively used and grant clear boundaries to concepts related to a Request, a Session or an Application by virtue of existence of a separate JavaBean for each. This, of course, improves code clarity and management, two of the vital enterprise-architecture-related characteristics.
On top of all, JSF comes with JSF Visual Designer that works in NetBeans. The visual designer is more sophisticated in many ways than the better-known Visual Studio IDE for .Net.
Last but not least, of the features that are critical to development of a Web 2.0 portal, is the Ajax-friendliness or the capability to use Ajax with the development technology. This is extremely handy and to a great extent automatic in JSF due to the presence of various Ajax component libraries like Woodstock Components, for instance. These components come with Ajax features out of the box that are usable with very little or no programming effort in Ajax. They are capable to do partial page updates on their own and based on the circumstances and events.
In comparison, Ruby on Rails is actually not designed for large-scale enterprise development like e-government 2.0 at all, and is most suitable for agile development of much smaller systems. It also does not have any visual design environment. But it enjoys being an MVC based platform, and is quite Ajax friendly.
.Net bears some of the typical criticisms that are usually out there against it, like its being a proprietary platform and therefore, again, less interoperable with any other technological phenomena.
Furthermore, .Net is not fundamentally based on a sound design pattern like MVC. Microsoft recently figured this out and made an effort to catch up by the release of MVC ASP.Net, however, this product is still at a hatching stage and is yet to be anywhere near comparable to the grass root MVC that exists in JSF.
ASP.Net Ajax satisfies the Ajax needs to some extent but the library options are not as many as the libraries that exist for JSF.
One last point that needs to be mentioned is the availability of API client libraries for the programming language of the platform in order to be able to follow the proposed methodology in this paper. By far the most available API client library for a language, in most of the services that this paper frequently refers to, is Java.
The following table summarizes some of the above characteristics comparatively among the three platforms.

	Visual designer	MVC	Client library APIs	Ajax Integration	Interoperability	J2EE compatibility	Open source
JSF	✓	✓	✓	✓	✓	✓	✓
RoR		✓		✓			✓
.Net	✓		✓	✓			

Table 2: Technology comparison

3.3.2 JSF Suitability as a Mashup Platform
JavaServer Faces is a very promising platform for development of large scale applications including e-government 2.0 and enterprise web applications, which is based on an MVC (Model-View-Controller) design pattern [31].
The view of the application is mainly constructed in JSP and Woodstock Ajax library, and the business-logic is created in Java and in a completely separate layer.
The separation of the business-logic is one of the principles of the MVC and its being developed in Java makes it the perfect development bed for using the Java API client libraries.
The most important element in existence of usable and viable mashups is the availability of standard usable APIs that can facilitate the services efficiently.
And if a Java client library is available for the APIs being used, the application development is made dramatically more efficient.

3.3.3 A Strong Visual Designer

JSF also possesses a strong visual web designer by the name of JSF Visual Designer that is plugged into NetBeans IDE.
The existence of a visual designer is undeniably vital to speed and productivity of application development, especially mashups development.
With JSF visual designer, one has almost complete control over the normal web components, and also Ajax ones like the Woodstock components. The Woodstock components are AJAXified in that they have individual update characteristics, independent from the page update. Hence, one can get Ajax properties by working with them without much JavaScripting.

3.3.4 Importance of MVC in Enterprise Development

Model-View-Controller (MVC) design pattern draws its success from the clear separation of functions. This is extremely significant for e-government 2.0 portal development because the one characteristic that is always there, is the fact that the application is huge and in order to manage and run an application of great magnitude, separation provides a breakdown that facilitates extensibility and also distributed execution of the application.
Fig.3 below is a snapshot of how MVC is reflected in JSF.

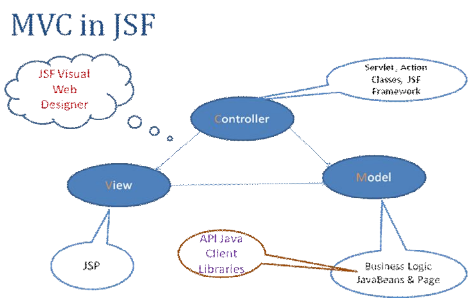
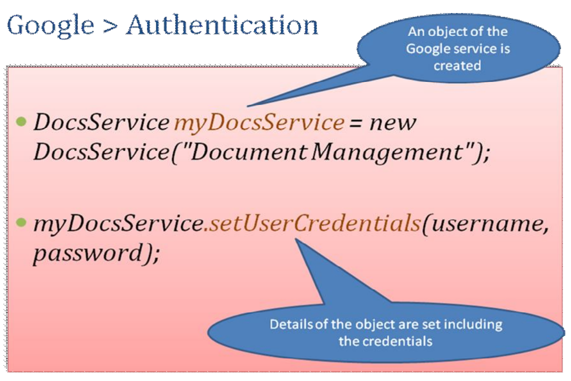

MVC in JSF
JSF Visual Web Designer
Servlet, Action Classes, JSF Framework
Controller
View
Model
JSP
API Java Client Libraries
Business Logic JavaBeans & Page

Fig.3: Implementation of MVC in JSF

3.3.5 Importance of a Java Layer

As you can see in the above figure the business logic is developed in Java, in JSF platform and in complete separation. Therefore, we suggest that the technique for designing mashups efficiently should be using this capability in order to communicate with the desired web services, wherever possible. Almost all available Web Services have Java Client libraries to work with; GData or the Google protocol and all Google services have almost all the required libraries in Java, and it is being updated constantly based on the collective user needs.
Some examples of how the Java client libraries are to be used are provided below in fig.4.

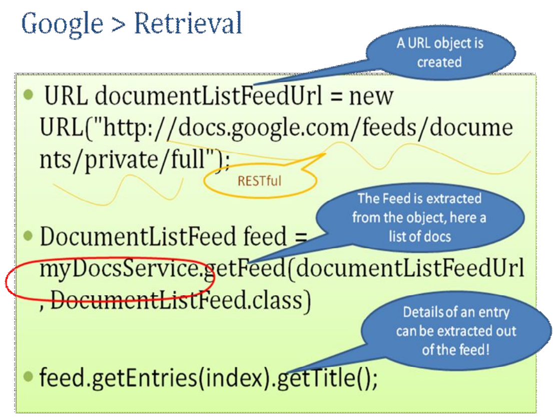

Google > Authentication
An object of the Google service is created
Details of the object are set including the credentials

```
• DocsService myDocsService = new
  DocsService("Document Management");

• myDocsService.setUserCredentials(username,
  password);
```

Fig.4: Google services authentication by API

As one can see above, Google services authentication can be as simple as above in fig.4.
Fig.5 below shows how data can be retrieved from a service, in this case, the Google Docs.


Google > Retrieval
A URL object is created
RESTful
The Feed is extracted from the object, here a list of docs
Details of an entry can be extracted out of the feed!

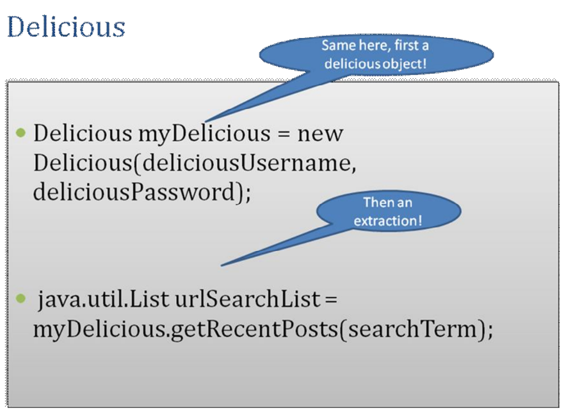
```
• URL documentListFeedUrl = new
  URL("http://docs.google.com/feeds/docume
  nts/private/full");

• DocumentListFeed feed =
  myDocsService.getFeed(documentListFeedUrl
  , DocumentListFeed.class)

• feed.getEntries(index).getTitle();
```

Fig. 5: Google services retrieval by API

The same is true for almost all other available Web Services through their Java client libraries, e.g., Delicious, the bookmark management service (fig.6).


Same here, first a delicious object!
Then an extraction!

Delicious

```
• Delicious myDelicious = new
  Delicious(deliciousUsername,
  deliciousPassword);

• java.util.List urlSearchList =
  myDelicious.getRecentPosts(searchTerm);
```

Fig. 6: Delicious services retrieval by API

3.3.6 Strengths of Using APIs' Java Client Libraries

(1) Compatibility with J2EE
J2EE as the de facto large scale development platform with its widespread popularity can be very well accommodating for the mashups with this proposed technique. As a matter of fact, the JSF platform is a part of the J2EE family, and since it now can be affectively used to create mashups, the proposed technique would be the natural one for this integration.
(2) Common approach
In order to have a successful e-government 2.0 portal development approach - although mashing up and ad-hoc development are being advocated here, the general approach for doing so should be unified for the sake of manageability and maintenance. We basically propose that a sound, reliable and common approach needs to be in place so that it enables further mashup development on the top. That is why standardizing the use of Java Client Libraries for all mashup activities would be strongly recommended and that is precisely what happens when JSF and Java Client Libraries are used almost at all times.
(3) Combination with business logic
Indeed, the real power of mashups comes from combining. And when the Java Client Libraries are used, the required logic-combinations with other services from the outside, and more importantly with the internal business-logic, become significantly easy.
Hence, with the use of TM, a platform will emerge with enough room for creative development on the top.

3.4 A Mashup Generator Platform

TM as a conceptual technique is materialized by the development of a mashup generator: an inter-component connection, where each component is an outlet to a Web Service (fig.7). It simply advocates that almost all components that are typically required in portals exist out there in the form of services and can be used again and again in different contexts. This brings the thought process to the need for existence of a platform that facilitates the reuse and organization of these components. These platforms could be called mashup generator Platforms or simply mashup generators.
A mashup generator facilitates the creation of a mashup without any programming knowledge required.
It basically allows the non-developer user to choose the components or widgets that they would like to have on their mashup from a set of available widgets with pre-set interconnectivity capabilities. In other words if one requires components like an outlet to their Google docs, an outlet to their Delicious bookmarks and one to their weblog, they can have all of them in form of widgets and all they need to do is to select them from a list of available widgets on the mashup generator and add them to their mashup page, in that way they will have a personalized mashup based on their unique needs. They might also require some sort of interconnectivity between these widgets, for instance one might need their calendar events to have the capability to add their favorite website links from the Delicious widget to the event description section of a calendar event by just choosing them from a pop-out list on the calendar widget.
There are also other possibilities, for instance, some widgets could exist for the purpose of advertisement for a particular brand, but they could have filtration criteria for product categories that could be set by the user or based on input from other widgets, and thereby users can have their preferred advertisement on their advertisement widget of choice on their personalized mashup.
Hence a mashup generator allows novice developers to take at least the following two actions:
(1) The users can choose their widgets of choice and add them to their personalized mashup.
(2) The users can set some criteria that some of the widgets have and thereby determine the type of output that they would like to see on those widgets on their mashups.

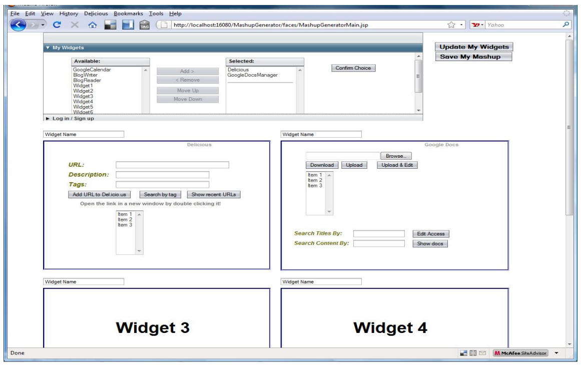


Fig.7: The mashup generator GUI

A mashup generator platform, however, should preferably have a set of tools that help its maintenance and improvement by developers or the developer community. Hence the following facilities should be available:
(1) Standards, guidelines and ultimately online tools for creation of widgets.
(2) Automated capability to use the guidelines for addition of the created widgets to the set of available widgets on the platform.
Hence, a mashup generator has the following benefits:
(1) User-development that has been covered in previous sections actually becomes a reality with the existence of a mashup generator platform, whereby users with little or no programming knowledge can form their personalized mashups for all purposes. This is extremely important because it enables user creativity in problem solving in their work environment and thereby the resulted mashups become very effective and to-the-point in terms of satisfying users' needs.
(2) The development team spends its time on creation and perfection of widgets and their communication ability without worrying too much about the context of use of the mashups, because the mashup generator could be used in any context, moreover, it will enjoy a much larger audience. Therefore, the most important software engineering task becomes the reusability architecture.

3.5 Testing the Implementation of a TM-based E-government 2.0 Portal

3.5.1 Test Overview

The Total Mashup Technique was tested through a practical and real-life implementation on an e-government 2.0 portal that targets researchers' collaboration commissioned by the government (fig.8).
With numerous portals whose requirements are met by the conventional portal architectures, everything should be created from scratch and normally independent from the outside world. Lengthy and man-hour consuming development would not necessarily accomplish all the collaborative needs of an e-government 2.0 portal specially sociability and sharability features.

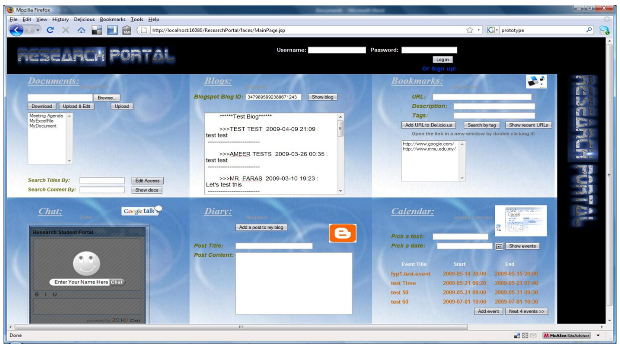


Fig.8: A screenshot of a TM-based e-government 2.0 portal for research collaboration (displayed features include documents management, blogging, social bookmarking, calendar, diary, and messaging)

Hence, the non-conventional technique of Total Mashup seems to be the right choice and is put to test. This adoption should not only result in fully functional features with sophisticated levels of social-networking and sharability, but also it makes it possible for a portal of such magnitude to be accomplished many times faster, hence successfully impact the ROI (Return on Investment).
The Mashup Generator Platform was instrumental in evolving the created research portal into a platform whereby registered users (software architects, developers, etc.) can choose the widgets that they desire and put them on their personalized mashup in the allocated slots, in the order they wish. As it is seen in fig.7, the first two slots are assigned with widgets, one is the Delicious widget and one is the Google Docs widget and the other two are empty and can be used for placing other widgets.
The observed significance of the Mashup Generator Platform according to the test is:
(1) The implementation of the required flexibility for choosing the widgets in the platform is easily feasible with the help of the utilized technology (i.e. JSF).
(2) The interconnectivity of the widget works mostly as it was in the case of the research platform itself.

(3) Considerable user development has been enabled.
The components that could be added to this platform in the future are:
(1) Widget creation facilities.
(2) Automation of widget addition.

3.5.2 Outcomes of the Test

To sum up the achievements of the Total Mashup Technique for the test research portal, the following could be enumerated:
(1) All required features are available as services on the web.
(2) The time is spent on creating mashup functionalities rather than the basic components. Mashup functionalities that are results of pulling different services together and creating new combinational services, can be very useful and creatively developed at ease. Nevertheless, the time of pulling these services necessitate performance testing because this greatly influences the continuity of the citizen with the e-government 2.0 portals.
(3) Efficient development was by far the most significant and immediate impact of taking the TM technique on this project. A project of this magnitude could have never been implemented with limited resources otherwise.
(4) Development is also very flexible in TM; and addition and replacement of features, is much more convenient and straightforward.
(5) The components of the research portal primarily represents the impact of Web 2.0 on e-government back-office domain, typically knowledge management and cross-agency collaboration (social bookmarking, blogging, messaging, calendar sharing, and documents exchange). The test did not involve the front-office domain like service provision, political participation, and law enforcement, which nevertheless are realized differently within other contexts by other researchers. Nevertheless, the front-office domain widgets could be added using the Mashup Generator.
(6) End-users find it easier to deal with the generated applications. This is attributed to the fact that many of them are already familiar with the third-party components' individual use. Hence, less resistance, training, and data duplication are imposed.
(7) The use of the mashup generator highly supports generatively: the ability to generate content without any inputs from the originators.
(8) Testing TM as open-source facilitates growth and innovation.

3.5.3 Challenges of the Total Mashup Technique According to the Test

Three of the most critical areas of the technique are:
(1) The great dependence on third party service providers is an issue, since the availability of services is completely dependent on them. The issue of availability is a sensitive matter for numerous e-government agencies, nevertheless it is quite acceptable to rely on established and reliable web services like the ones from Google, Apple, Amazon, etc., especially with the growing trend towards Cloud Computing.
(2) Corollary to the challenge of dependence in point 1 follows the issue of IP (Intellectual Property) protection. Nevertheless, signing non-disclosure agreements can alleviate this, since many of these third-party service providers like Google are very keen in expanding their business to government.
(3) Security might also be a challenge but not as big as many would like to portrait when it comes to mashups, especially when TM is embraced, because when TM is in place the total architecture is designed to accommodate mashups and critical security measures can be put in place from ground up, although this new technique might require some creativity in security, but it is plausible that they are easily achievable. More and admittedly, we witness both security and end-user convenience clash. For instance, one concept that is used within the this test to achieve end-user convenience is SSO (Single-Sign On); SSO enables a single action of user authentication and authorization that allows a user to access all computers and systems where he has access permission, without the need to enter multiple passwords. In its simplest form, SSO is implemented by internally storing the authentication information (mostly, username and password) of every third-party service provider (Google, Delicious, etc.) and later using these credentials for log in (fig.4). Obviously, having an SSO feature should always be associated with high measures protecting the end-user's single authentication information.
Moreover, when speaking about security in a macro-perspective the net-gain should be compared against the net-loss, which in this case the amount of increased overall productivity due to relevance and functionality of the mashups, is arguably and significantly higher than possible security breaches, when taking into account all the possible ways of breaching into a conventional portal or the possible ways of data-leak through users' negligence.

4 Conclusion

This work is an effort to emphasize the necessity and practicality of mashup design integration, as a part of the Total Mashup technique, into the architecture of e-government 2.0 portal projects. It differentiates itself from many other enterprise 2.0 efforts in that they suggest the mashup capability as a third party, and as an additional extension to current architectures at enterprises. The research carried out appreciates the progressive thought in that direction yet deems that it might not be enough.

The conceptual nature of TM is evaluated through the deployment of real e-government 2.0 portals that emphasizes high degree of collaboration among researchers.

The TM technique demonstrates the possibility of an innovative design technique and sets the path for further development initiatives of similar scenarios in the future. Several outcomes and challenges are unearthed.

TM is highly driven by the shift towards to Cloud Computing; hence it inherits some of its research challenges like dependency on their services providers, IP, and security.

We view TM as a propitious technique that should be considered by software architects who are commissioned to develop e-government 2.0 portals.

The future of e-government 2.0 is bound to involve a great number of mashup applications; hence methods, approaches and design patterns need to be devised to cater for this need. The blend of e-government 2.0 activities with Internet cloud services is inevitable and the e-government agencies that realize that before others will have the bigger wins or at least avoid losses (first-movers advantage).

We perceive that a detailed operational framework for the Total Mashup technique is obviously required. More, there is a dire need to undergo research on methods to maintain availability of services upon which mashups are designed. Finally, empirical research on performance and stress testing might be beneficial to unearthing scalability issues.

References:

[1] J. Baumgarten, and M. Chui, E-government 2.0, *McKinsey Quarterly - Public Sector Practice*, www.mckinseyquarterly.com/E-government_20_2408, 2009.

[2] D. Osimo, Web 2.0 in Government: Why and How? *European Commission, Joint Research Centre*, Institute for Prospective Technological Studies, 2008.

[3] P. Klein, Web 2.0: Reinventing Democracy, *CIO Insight Magazine*, 2008, pp. 30-43.

[4] P. Deitel, H. Deitel, *Ajax: Rich Internet Applications and Web development for programmers*, Prentice Hall, 2008.

[5] D. Hinchcliff, *Enterprise Web 2.0,* http://blogs.zdnet.com/Hinchcliffe, 2009.

[6] C. Klocker, *Mashups: A New Concept in Web Application Programming*, VDM Verlag, 2008.

[7] Gartner Inc., *Government and Web 2.0: The Emerging Midoffice*, Stamford, 2007.

[8] I. Becerra-Fernandez, A. Gonzalez, R. Sabherwal, 2003, *Knowledge Management: Challenges, Solutions, and Technologies*, Prentice Hall, 2003.

[9] J. Feiler, *How to Do Everything with Web 2.0 Mashups*, McGraw-Hill Osborne Media, 2007.

[10] W3C Press Release, *W3C eGovernment Activity to Help Empower Citizens*, http://www.w3.org/2008/06/egov-pressrelease, 2009.

[11] , [12] IBM Corporation Website, *Why Mashups Matter*, ftp://ftp.software.ibm.com/software/lotus/lotusweb/portal/why_mashups_matter.pdf, 2009.

[13] P. McAfee, Enterprise 2.0: The Dawn of Emergent Collaboration, *Sloan Management Review*, Vol.47, No.3, 2006, pp. 21-28.

[14] Online Public Encyclopedia, Wikipedia Corp., *Enterprise2.0*, http://en.wikipedia.org/wiki/Enterprise_2.0, 2009.

[15] AIIM official website, *What is Web 2.0?*, http://www.aiim.org/What-is-Web-2.0.aspx, 2009.

[16] T. O'reilly, *Inventing the Future* , http://www.oreillynet.com/pub/a/network/2002/04/09/future.html, 2008.

[17] T. O'Reilly, *What Is Web 2.0: Design Patterns and Business Models for the Next Generation of Software*, http://www.oreillynet.com/pub/a/oreilly/tim/news/2005/09/30/what-is-web-20.html, 2008.

[18] A. Mohamed, *Moving closer to the Internet as an operating system*, http://www.computerweekly.com/Articles/2008/05/23/230812/moving-closer-to-the-internet-as-an-operating-system.htm, 2008.

[19] V. Barker, (2009), Older adolescents' motivations for social network site use: The influence of gender, group identity, and collective self-esteem, *CyberPsychology & Behavior*, Vol.12, No.1, pp. 209-213.

[20] F. Gorder, Grid Computing Yields Earthquake Forecast, *IEEE Computing in Science &*

Engineering, January-February, Vol.9, No. 1, 2007, pp. 6–10.

[21] L. Vincent, Taking Online Maps Down to Street Level, *IEEE Computer Journal*, Vol.40, No. 12, 2007, pp.118 – 120.

[22] H. Elmeleegy, A. Ivan, R. Akkiraju, R. Goodwin, Mashup Advisor: A Recommendation Tool for Mashup Development, *ICWS '08 IEEE International Conference on Web Services*, 2008, pp.337–344.

[23] Y. Balzer, *Improve your SOA project plans*, http://www.ibm.com/developerworks/webservices/library/ws-improvesoa, 2004.

[24], [25] C. Weinhardt, A. Anandasivam, B. Blau, J. Stosser, Business Models in the Service World, *IEEE IT Professional*, Vol.11, No.2, 2009, pp. 28-33.

[26] E. Castro-Leon, J. He, M. Chang, Scaling Down SOA to Small Businesses, SOCA '07, *IEEE International Conference on Service-Oriented Computing and Applications*, 2007, pp. 99 – 106.

[27] W3C Working Group Note, 2004, *Web Services Glossary*, http://www.w3.org/TR/ws-gloss, 2004.

[28], [29] R. Thomas, *Architectural Styles and the Design of Network-based Software Architectures*, Doctoral dissertation, University of California, Irvine, 2000.

[30] V. Viswanathan, Rapid Web Application Development: A Ruby on Rails Tutorial, *IEEE Software*, Vol.25, No.6, 2008, pp. 98-106.

[31] C. Schalk, E. Burns, J. Holmes, *JavaServer Faces: The Complete Reference*, McGraw-Hill Osborne Media, 2006.