Mashup Patterns applied to e-Government*

Michele Stecca
Computer Platform Research Center (CIPI)

Massimo Maresca
Computer Platform Research Center (CIPI)

About the Authors
Michele Stecca received his Master’s degree from the University of Padova (Italy) in 2007 and he is currently a PhD student at CIPI (Computer Platform Research Center, University of Genova and Padova). His research interests include JAVA technologies for Service Delivery Platforms (HTTP/SIP Servlet, JSLEE, J2EE, JMS), Web Services composition and Mashups.

Massimo Maresca received a Research Doctorate in Computer Engineering (1986) from the University of Genova, spent some years as a Post Doctoral scientist at the IBM T. J. Watson Research Center, Yorktown Heights, New York and at the International Computer Science Institute, Berkeley, California, became an Assistant Professor and an Associate Professor at the University of Genoa (Italy) and finally, in 1994, took a position of Professor of Computer Engineering at the Department of Information Engineering of the University of Padova. Dr. Maresca published several tens of papers in refereed journals, books and conference proceedings, was in the program committees of several scientific workshops and conferences as well as an Associate Editor of the Proceedings of the IEEE. His research interests are in the area of parallel/distributed computing, networking and services.

Corresponding Author Mailing Address, Phone and E-mail
Michele Stecca
c/o DIST University of Genova
Via Opera Pia 13, 16145 Genova
Telephone: +390103532709
E-mail: m.stecca@cipi.unige.it

* This paper is a revised and extended version of a paper that was presented at the “IEEE First International Workshop on Socio-Technical Aspects of Mashups” [1]
Mashup Patterns applied to e-Government

Michele Stecca
Computer Platform Research Center (CIPI)

Massimo Maresca
Computer Platform Research Center (CIPI)

Abstract
In this paper we focus on the structure of Server Side Mashups, that is of composite services executed and orchestrated in managed platforms, and on the application of Server Side Mashups in e-Government applications. Unlike Client Side Mashups, which run in client platforms under direct user control, Server Side Mashups run in background in remote platforms. The fact that they do not require direct user control enables a set of applications which are not supported by Client Side Mashups.

We first present a set of Patterns based on the type of user-service interaction, then we show how these Patterns can be used as a reference to implement value added applications in a number of application domains such as business, entertainment, e-health, etc. The classification of the Mashup Patterns that we describe in this paper stems from our experience gained during the participation to the European Community co-funded project called OPUCE (Open Platform for User-Centric Creation and Execution, contract n. 34101) which aimed at providing a platform for the easy creation and the efficient execution of Mashups.

Finally we focus on one specific application domain, namely e-Government, and show how the proposed Patterns might be used to create new applications that increase the quantity and improve the quality of Public Administration services. We also provide a preliminary analysis of a business model that might be enabled by the realization of a “Mashup ecosystem”.

Keywords: Mashups, Patterns, e-Government, 3rd Party, SME
Introduction

The availability of contents and services due to the advent of the so called Web 2.0 paradigm leads to the need of new tools that allow combining different contents and services in composite applications called Mashups (a.k.a., Service Compositions). While contents are usually provided through portals (e.g., YouTube for videos), social networks (e.g., Facebook, Twitter, LinkedIn, etc.) and blogs, services are usually provided through Web Services (e.g., Weather Information, Financial Information, Maps, personal communications, etc). One of the best known Mashup example is the presentation of traffic information (e.g., source Yahoo!Traffic) on a map (e.g. source Google Maps). A very interesting class of Mashups is that of Enterprise Mashups [2]. Enterprise Mashups combine contents and services belonging to private enterprise domains (e.g., Customer Relationships Management data, order status, etc.) with public domain contents and services.

Many of the existing Mashups (see [3]) were developed as computer applications. To overcome the limitation that developing a Mashup requires programming skills, the scientific community and the industrial community have addressed the design and the development of visual programming tools aimed at making Mashup creation easier and faster. Thanks to such tools even non-skilled programmers (e.g., business/marketing people) can create their own personalized Mashups, targeting the so called Web 2.0 long tail (see for instance [2]). Examples of such tools are Presto JackBe [4], IBM Mashup Center [5], OPUCE platform [6], etc.

In spite of the big hype around Mashups, what is still missing in the literature is a clear classification of what Mashups can do and cannot do. In this paper we propose an answer to this question by introducing a list of Services and a list of Mashup Patterns. The Services represent the atomic components that can be used to create Mashups while the Mashup Patterns represent common ways to combine Services.

Mashups can be successfully used in many different domains (e.g., business, entertainment, e-health, etc.). We focus on e-Government applications, which can take advantage of the huge amount of services and contents that governments are making available to citizens, as a consequence of the recent directives aiming at enforcing transparency, participation and collaboration (e.g., see the US Open Government Directives [7]).
After introducing the list of Services and the list of Mashup Patterns we describe a business model in which 3rd Party companies, with respect to server platform operators and end-users, may find a role in a “Mashup ecosystem” in a way similar to the well-known Apple App Store ecosystem [8].

In summary, the main contributions of this paper are the following:

1. We propose a classification of both the basic Services and the Mashups that can take advantage of a Server-side execution platform. This classification stems from our experience gained during the participation to the European Community co-funded project called OPUCE (Open Platform for User-Centric Creation and Execution, contract n. 34101) which aimed at providing a platform for the easy creation and the efficient execution of Mashups.

2. We show how the proposed Patterns can be successfully applied to a very important application domain: the e-Government case study.

3. We propose a “Mashup ecosystem” which describes the economical relationships created among the different actors (e.g., end users, Mashup developers, 3rd Parties, Platform Owner, etc.) involved in the usage of Server-side platforms.

The paper consists of seven Sections: the first Section establishes the reference scenario used in the rest of the paper, i.e. the Server Side Mashup scenario, the following two Sections respectively introduce the Service Patterns and the Mashups Patterns, the fourth Section describes some of the Mashup application domains and the fifth Section focuses on the e-Government scenario. The sixth Section describes a “Mashup ecosystem” focusing on the role of Small Medium Enterprises. We finally conclude in the last Section.

Server Side Mashups

Client Side Mashups vs. Server Side Mashups

Depending on where the Mashup service components are actually “mashed-up”, that is on where service composition takes place, a Mashup is denoted “Client Side”, when its execution takes place within a client platform such as a browser or a Smartphone, while it is denoted “Server Side”, when its execution takes place in a service provider platform [9]. There are some kinds of Mashups in which the client part is very important (see, e.g., the Map visualization
Pattern showing some information on a Map like Google Maps) but in this paper we focus on the Server-side approach for the following reasons:

- **Long Running Mashups:** While Client Side Mashups typically exhibit short processing times and run under the direct control of a user, Server Side Mashups are typically characterized by long processing times and do not require the direct control of a user. On the contrary they can be permanently active, able to issue events when a configured situation occurs (e.g., a mail monitor service continuously monitors the mail server and issues a new event when an e-mail message matching a given filter is received). In a Server Side Platform applications remain active even when the user terminal is not on line (for example because of battery exhaustion, explicit switch off, poor radio coverage, etc.).

- **Privacy Protection:** While Client Side Mashups receive and process raw contents from content providers in an external and uncontrolled location, in Server Side Mashups content composition takes place in a protected environment (i.e., the Server Side Mashup execution platform). Client systems only receive the composition of contents that the Mashup authorizes. For example when a Mashup combines contents protected by a privacy policy (e.g., hospital patient information, government citizen information, company customer information) the enforcement of the global privacy policy can be managed at the Mashup level by a trusted third party managed by the Platform Operator.

- **System Protection:** While Client Side Mashups freely access the content/service provider platforms to retrieve contents and/or to activate services and as a consequence exhibit no protection against attacks, on the contrary Server Side Mashups protect the platforms by interposing a processing layer that inspects, authorizes and, if necessary, blocks content/service requests.

- **User Terminal power consumption:** While Client Side Mashups consume the energy power of the end-user terminals (e.g., the battery of a Smartphone), on the contrary Server Side Mashups show minimal impact on client devices. For instance, the presence of Monitoring Services (e.g., Monitor RSS Feed, Monitor e-mail inbox, etc.) based on polling mechanisms may affect the performance in terms of duration of the user terminal battery as proved by a recent study by Ericsson Labs [10].
**Server Side Mashup Platform**

A Server Side Mashup Platform is based on a Service Execution Platform (SEP) operated and managed by a Platform Operator. A Service Creation Platform (SCP) can be downloaded and used to create Mashups. Then, to be executed, the Mashups must be uploaded and deployed in the SEP, according to the Cloud paradigm [11]. Uploading, deploying and executing Mashups must comply with a set of security policies established by the Platform Operator.

The SCP is a graphical tool that presents to the service composer a set of basic service components called Services (Svc) typically acting as mediators of external services related to external contents. In the SCP the Svcs are represented by blocks that execute actions, issue events and possess input/output properties. Mashups are created by dragging and dropping Svc blocks and by linking them through edges (see Figure 1). The semantics related to one edge can be expressed as follows: “when an event occurs in a Svc then a set of actions are invoked in the following Svc(s)” If necessary, some of the output properties of the Svc from which an event originates are copied into some of the input properties of the Svc(s) to which the event is directed. It is worth noticing that the SCP allows non-skilled programmers to easily and quickly create Mashups.

The Service Execution Platform (SEP) takes care of the execution of the Mashups. At run time the edges linking the basics blocks are executed as sequences of component interactions. Interested readers may refer to [12] for further details.

**Service Patterns**

In this Section we list some of the most significant Service Patterns that can be used to create Mashups.

**Monitor Svc Pattern**

A Monitor Svc is a long running Svc that monitors a specific resource looking for new items that match a set of filters specified through the Svc input properties. Whenever a Monitor Svc finds a new item matching the input filters, it issues an event (note: if all new items are of interest, it is possible to set the input filter equals to “*”). We propose to classify the Monitor Svc Pattern into five sub-categories, namely:
• Message Monitoring: the Svcs belonging to this sub-category manage the incoming messages of an inbox and issue an event when a message matching the input filters is received. Different messaging technologies can be supported (e.g., SMS, MMS, e-mail, etc.).

<table>
<thead>
<tr>
<th>Svc</th>
<th>Resource</th>
<th>Item</th>
<th>Filters Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>Monitor Gmail</td>
<td>Gmail Inbox</td>
<td>E-Mail</td>
<td>“Object field contains…” “Body field does not contain…”</td>
</tr>
<tr>
<td>Monitor SMS</td>
<td>SMS Inbox</td>
<td>SMS message</td>
<td>“Sender number equals…” “Content contains…”</td>
</tr>
</tbody>
</table>

Table 1 Message Monitoring Service examples

• Information Provider Monitoring: the Svcs belonging to this sub-category monitor the evolution of the contents provided over the network by a variety of sources such as Newspapers, Blogs, Government Offices, Public Transportation Companies, Job Search Agencies, Weather Forecasting Services, Road Traffic Management bodies, etc. These types of information are usually made available over the network through RSS Feeds/Atom technologies. A Monitor RSS Feed Svc enables the access to this huge set of external contents.

<table>
<thead>
<tr>
<th>Svc</th>
<th>Resource</th>
<th>Item</th>
<th>Filters Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>Monitor RSS Feed</td>
<td>RSS Feed</td>
<td>RSS Feed Item</td>
<td>“Title contains…” “Author is…”</td>
</tr>
</tbody>
</table>

Table 2 RSS Monitoring Service example

• Scheduled Events Monitoring: the Svcs belonging to this sub-category manage events like deadlines, appointments, anniversaries, etc. The technologies through which these Svcs are provided may be different (e.g., Google Calendar, internal companies/public institutes systems, etc.), depending on the specific application.

<table>
<thead>
<tr>
<th>Svc</th>
<th>Resource</th>
<th>Item</th>
<th>Filters Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>Monitor Google Calendar</td>
<td>Calendar</td>
<td>Calendar Event</td>
<td>“Event date equals…” “Event name contains…”</td>
</tr>
<tr>
<td>Monitor Tax Payments</td>
<td>Government Database</td>
<td>Database Entry</td>
<td>“Deadline date equals…”</td>
</tr>
</tbody>
</table>

Table 3 Scheduled Events Monitoring Service examples
Presence Monitoring: the SvcS belonging to this sub-category monitor the Presence status of a user and issue events accordingly, thus triggering the execution and/or the progress of a Mashup (e.g., a Mashup performs some operations if the user status is “busy” and some other operations as soon as the user status becomes “available”). XMPP (eXtensible Messaging and Presence Protocol) is the best known protocol, used for instance in the Google Instant Messaging system (i.e. GTalk), that supports presence monitoring.

<table>
<thead>
<tr>
<th>Svc</th>
<th>Resource</th>
<th>Item</th>
<th>Filters Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>GTalk Presence</td>
<td>User Presence</td>
<td>Presence</td>
<td>“Presence equals…”</td>
</tr>
<tr>
<td>System</td>
<td></td>
<td></td>
<td>“Presence contains…”</td>
</tr>
</tbody>
</table>

Table 4 Presence Monitoring Service example

User Localization Monitoring: the SvcS belonging to this sub-category retrieve the user geographic coordinates. They typically rely on the GPS receivers currently available in most Smartphones or on Internet services such as Yahoo Fire Eagle [13] or on Telecom Operator Services (see for example Vodafone 360 [14]).

<table>
<thead>
<tr>
<th>Svc</th>
<th>Resource</th>
<th>Item</th>
<th>Filters Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>Yahoo Fire Eagle</td>
<td>User Localization</td>
<td>User Location</td>
<td>“User is close to…”</td>
</tr>
</tbody>
</table>

Table 5 Localization Monitoring Service example

Notification Svc Pattern

A Notification Svc allows users to send contents to other people by means of different technologies and/or different networks. The typical input parameters are: 1. sender credentials (e.g., Mail Box login/password), 2. receiver ID (e.g., telephone number, e-mail address), and 3. content of the notification message.

Here is a list of possible Notification SvcS:

- Personal Communication Notification SvcS (e.g., “Send SMS”, “Send Instant Message”, “Send Mail”, “Send MMS”, “Make phone call”, “Text to Speech Call”);
- Social Network Notification SvcS (e.g., status line update, new note, link sharing, etc. on Facebook, new tweet on Twitter);
- Calendar Event Notification SvcS (e.g., Google Calendar);
Presence Notification Svcs (e.g., GTalk).

Data Connector Svc Pattern

A Data Connector Svc is a “proxy” that accesses a content source, extracts a content, converts it appropriately and fills out the Svc output properties. Here follows a list of possible Data Connector Svcs:

- Database Data Connector Svcs (e.g., MySQL Connector, MS Access Connector);
- Spreadsheet Data Connector Svc (e.g., MS Excel Connector, Open Office Spreadsheet Connector);
- XML Data Connector Svcs.

This Pattern is of particular relevance for Data-Mashup, i.e., applications that combine data provided by different sources and/or based on different formats. The e-Government case study that we describe later demonstrates the usefulness of this Pattern because many government agencies provide huge amounts of data but these information are saved in many different formats (e.g., Spreadsheets, CSV files, plain text files, XML files, etc.).

General Purpose Svc Pattern

In addition to the Patterns mentioned in the previous paragraph the following non exhaustive list of General Purpose Svcs can be used to manipulate data and/or to influence the execution of a Mashup:

- Formatting Svcs: the output information of a Monitor Svc may be re-formatted to satisfy the need of the final user better;
- Filtering Svcs: if the filtering functionality provided by a Monitor Svc is not sufficient, a more specific filtering Svc can be added;
- Sorting Svcs: if the output information of a Monitor Svc is a structured data set (e.g., an RSS Feed), it is possible to define some Svcs that sort the items included in the data set;
• Conditional Svcs: it is possible to influence the execution of a Mashup by introducing a conditional Svc, that evaluates a condition indicated in the input properties and selects one execution branch to be executed among a set of alternatives;

• Merging Svcs: it is possible to merge structured information elements retrieved even from different data sources.

Geographical Map Svc Pattern

The Geographical Map Pattern, or more simply the Map Pattern, is a specific Svc while it is not a generalization of a subset of Svcs. We present it as a standalone Pattern because of its relevance. A Map Svc takes the longitude and the latitude of a set of points as an input and displays them on a Map through a marker. It provides the Map visualization functionality used in a variety of applications. Google Maps is the best known Map service currently available on the Web.

Mashup Patterns and Real Life examples

In this Section we propose a set of Mashup Patterns and a set of examples taken from real life.

Pattern #1: Resource Monitoring and Notification

Figure 1 Resource Monitoring and Notification Mashup

Figure 1 shows the Resource Monitoring and Notification Pattern. It involves two Svcs and allows the Mashup developer to interconnect different information sources to different notification channels. The semantics related to this Mashup can be expressed by the following statement: “when the Monitor Svc finds a new item matching the input filters, the Monitor Svc issues a new event that triggers the ‘send notification’ action in the Notification Svc”. Once a set
of highly reusable basics blocks are implemented (see in particular the Monitor and the Notification Service Patterns), this Mashup Pattern can be used to create at least three different kinds of Mashups, namely:

**Reminder Mashups**, such as for example:

- Book Lending Deadline reminder: The day before the deadline, send an e-mail (or any other kind of notification) to the person who is supposed to return the book to the library (Monitor Book Lending Deadline Svc + Send Email Svc);
- Appointment reminder: Check the date of a specific appointment on your Google Calendar and send an SMS (or any other kind of notification) the day before that date (Monitor Google Calendar Svc + Send SMS Svc).

**Alert Mashups**, such as for example:

- Public Transportation Delay Alert: Monitor the progress of a train trip and send an SMS in case of delay (Monitor Train Delay Svc + Send SMS Svc);
- Traffic Jam Alert: Monitor the traffic around a specific location and notify congestion through an SMS (Monitor Traffic RSS Feed Svc + Send SMS Svc).

**Automatic notification of significant events Mashups**, such as for example:

- Announcement of an exhibition on a Museum Web Site: “Tweet” the announcement on a Twitter profile to notify the followers about such an event (Monitor Museum RSS Feed Svc + Tweet Svc);
- Announcement of a Rock Concert on Facebook: Notify the announcement issued by a Facebook subscriber through a SMS (Monitor Facebook Events Svc + Send SMS Svc).
Pattern #2: Resource Monitoring, Processing and Notification

Figure 2 Resource Monitoring, Processing and Notification Mashup (Time Dependant Notification example)

It is an extension of the Pattern introduced in the previous Section. Here one or more Svcs are inserted between the Monitor Svc and the Notification Svc in the so called “Processing Part” of the Mashup.

This Pattern can be used to create several types of Mashups. We propose three basic examples:

*Time Dependent Notification Mashups:* This Mashup allows to personalize the Notification Channel based on the time of the day. A processing layer retrieves the time of the day at which the new announcement was added, compares such a time with a configured value and selects the Notification Channel accordingly.

*Location Dependent Notification Mashups:* The Mashup selects the Notification Channel depending on user location. The processing part of the Mashup depicted in Figure 2 is a Localization Svc + Conditional Svc which selects the SMS communications channel if the user is near home, or the call otherwise. “Location Dependent Notification” Mashups are relevant for example in touristic applications, as they enable the creation of Value Added Services that notify tourists when they get close to points of interests (museums, monuments, restaurants, etc.).

*Presence Dependent Notification Mashups:* The Mashup selects the Notification Channel depending on user Presence Status. The processing part of the Mashup shown in
Figure 2 is a Presence Svc + Conditional Svc which selects the SMS communications channel if the user is “at work”, the text to speech call otherwise.

**Pattern #3: Resource Monitoring, Processing and Map visualization**

Figure 3 Resource Monitoring, Processing and Map Visualization Mashup

Figure 3 depicts the typical Web Mashup Pattern in which some information elements are shown on a Map. In this case the event that triggers the execution of the Mashup is the Monitoring Svc, which is a long running service. Some examples of this kind of Mashups are listed next:

- When a new event is published in a given Facebook account, the Mashup visualizes the location of such an event (Monitor Facebook Events Svc + Google Maps Svc) on Google Map;

- When a new Event is published on Google Calendar with a given keyword (e.g., jazz concert), the Mashup visualizes the location of such an event (Monitor Google Calendar Svc + Conditional Svc + Google Maps Svc) on Google Map.

**Pattern #4: Data Aggregator**

Figure 4 Data Aggregator Mashup
In the data aggregator Pattern information elements retrieved from different sources are merged together to create new contents. The simplest case is the one in which all the different sources follow the same format (e.g. RSS Feed): in this case the Aggregator Svc only has to merge the inputs. An example of a more complex case is depicted in Figure 4 where the two different data sources follow different formats. The Aggregator Svc translates the different formats in a common “internal” format and checks the compatibility of the inputs (e.g., same column numbers and names for the Excel and Access tables).

This Pattern is useful when different public sources must be merged (e.g., news about the same topic from different newspapers, or a set of significant events - e.g. “all the jazz concerts”- from different cities, etc.) or when a heterogeneous set of information elements must be merged (e.g., company fusions, data migrations, etc.).

Application Domains
We now briefly describe a set of domains in which the proposed Svcs and Mashup Patterns can be successfully used to create new value added composite services or Mashups.

Enterprise Domain: Enterprise Mashups can be associated to the so called Web 2.0 Long Tail of applications, i.e., to the list of applications that might be useful within an Enterprise but that the Information Technology department has decided not to prioritize for development [2]. Enterprise Mashups are typically created by the employees by means of graphical Service Creation Environments [4] to merge different types of data, to extract information from RSS feeds, etc.;

Health Domain: In Health applications the patient information can be combined with data/services available over the Web (e.g., forecast services, epidemic diseases diffusion data, etc.). For instance, the “Pollen Journal” service [15] allows to automatically receive information from different sources on the basis of the geographical location of a user, to display allergy outbreaks on a map and to generate alerts depending on the pollen levels;

Business Domain: Financial information elements available on the Web (e.g., stocks values, currency exchange rates, raw material prices) can be combined by enterprises and people with internal/external data to perform operations like price comparison, revenue forecasting, etc. Even these information elements are often provided through RSS Feed technology;
Personal/Social Domain: Many Social Networks, like Facebook, Twitter, and Foursquare provide large amounts of information (e.g., users status, users uploaded contents, users generated events, users check-ins, etc.) that can be used to create new value added services, in combination with notification channels, localization services, presence services, etc. Social Networks information is often made available through RSS Feed technology;

Entertainment Domain: Entertainment applications may take advantage of the publication of new events (e.g., music concert dates, sport events, news, etc.) through RSS Feeds or Calendar Svcs to create personalized applications along with notification Svcs;

Tourism Domain: Location Svc is one of the key enablers for the implementation of Mashups in the tourism domain, in particular to personalize the information sent to tourists by a Content provider. For example the composition of a Location Svc with a database storing the historical information of a city supports the creation of a service that sends the description of the points of interest to the tourists terminals at the appropriate time;

Government Domain: Many Government agencies provide large amounts of contents to citizens (taxes, auctions, public services, etc.) through RSS Feeds, e-mails, etc. These information elements can be combined with other delivery technologies like SMS, MMS, Calendars, Social Networks, etc. to create new value added services. This application domain is widely described in next Section.

Application of the Mashup Patterns to the e-Government

The electronic-Government (e-Government, a.k.a. e-Gov, Government 2.0, and digital Government) initiatives aim at facilitating the interaction between the governments and citizens, enterprises and other governments through the adoption of innovative paradigms and through the use of advanced ICT tools.

Many governments, e.g., the USA [16], the UK [17], Australia [18], New Zealand [19], Norway [20], etc., and many local authorities located in large cities such as Washington DC [21], New York City [22], San Francisco [23], London [24], Toronto [25], etc. provide contents and services over the Web. The diffusion of the data exposure trend is being accelerated by the introduction of specific regulations as it happens for example in the USA, where “the three principles of transparency, participation, and collaboration form the cornerstone of an open
government” are currently enforced [7]. More specifically all the American national agencies are invited to provide as many data sets as possible in open formats on specific portals located at http://www.[agency].gov/open URLs. All the U.S. government data are collected in the White House aggregator portal [26] which counts 168000 data sets uploads until July 2010.

As described in [27], government data might be provided in different formats such as Relational Databases, Spreadsheets (e.g., Comma Separated Values – CSV or XLS files), XML files, Web Pages, Plain Text, RESTful Web Services and in general XML files or JSON objects. As suggested in [28], since these data might change and new contents might be uploaded on the portals by governments, it is desirable to take advantage of the RSS Feed technology to notify the public about the additions and modifications.

In any case the information and the services provided by governments can be used as atomic components to create Mashups (e.g., see [29], [30] and [31]). In the following we describe the application of the Mashup Patterns described in the previous Section to the e-Government domain.

*Mashup Pattern #1 applied to e-Government*

The “Resource Monitoring Svc + Notification Svc” can be successfully used in the e-Government domain because a huge number of government institutions provide useful information to citizens by means of specific technologies such as RSS Feed, e-mail, Social Network (e.g., Twitter, Facebook, etc.), etc. that can be monitored by a Resource Monitoring Svc to create personalized notification channels through the appropriate Notification Svc. Here are some example of resources that can be monitored:

- The USA Government Web Site provides RSS feeds in the areas of Business & Economics (e.g., Federal Reserve news), Health (e.g., Pandemic Flu news), Public Safety and Laws (e.g., FBI news), Defense (e.g. Army news), etc.
- The California State Web Site provides RSS Feed and e-mail notifications concerned with the Department of Justice, with the California Seismic Safety Commission, etc.
- Many governments follows the directive suggested in [28] by having their portals provide RSS Feeds reporting the changes and/or the additions of new information.
This kind of RSS might also be monitored and used to create “Resource Monitoring Svc + Notification Svc”.

Some examples of the use of this Mashup Pattern are:

- **“Reminder” Mashups**, e.g., the Tax Payments Deadline reminder: The day before the submission deadline, send an SMS (or any other kind of notification) to the people who are supposed to pay a certain tax (Monitor Tax Payments Deadline Svc + Send SMS Svc);
- **“Alert” Mashups**, e.g., Public School Closing / Delay Alert: Provide updates about unscheduled public school closings, delays and early dismissals (Monitor Public School RSS Feed Svc + Send SMS Svc);
- **“Automatic notification of significant events” Mashups**, e.g., Add a “New Auction event” to Google Calendar: If the Government publishes a tweet on Twitter related to a new auction for a house then the Mashup automatically updates the Google Calendar application of the user by adding a new “Auction Event” to it (Monitor Twitter Account + Google Calendar Svc).

**Mashup Pattern #2 applied to e-Government**

This Pattern represents an extension of the “Resource Monitoring Svc + Notification Svc” Pattern. Examples of the use of this Mashup Pattern in the e-Government domain include:

- **Time Dependent Notification Mashups** that allow to personalize the Notification Channel based on the hour of the day. For example a Mashup similar to the one depicted in Figure 2 composed by the “Monitor RSS Feed Svc + get time of the day Svc + Conditional Svc + Send SMS Svc + Text to Speech Call Svc”, works as follows:
  
  o Keep Monitoring an RSS Feed provided by a Local Authority that reports the list of car accidents;

  o As soon as the new information is retrieved trigger the “get time of the day” Svc which retrieves the time of the day,
If the Conditional Svc classifies such a time as a working hour then send an SMS to the user with the info of the RSS item retrieved from the first Svc, otherwise invoke a Text to Speech Call Svc.

- **Location Dependent Notification Mashups** that allow personalizing the Notification Channel based on end user location. For example a Mashup similar to the one depicted in Figure 2 composed by the “Monitor Twitter Account Feed Svc + get user location Svc + Conditional Svc + Send SMS Svc + Text to Speech Call Svc”, works as follows:
  - Keep Monitoring a Government Twitter account that publishes the information related concerts scheduled in the evening in the city,
  - As soon as this information is retrieved trigger the “get user Location” Svc which retrieves the location of a user (e.g., by using the Yahoo Fireeagle APIs [13]),
  - If the Conditional Svc classifies such a location to be near the city of the concert then send an SMS to the user with the information related to the concert item retrieved from the first Svc, otherwise invoke a Text to Speech Call Svc.

- **Presence Dependent Notification Mashups** that allow to personalize the Notification Channel based on the Presence Status of the end user. For example a Mashup similar to the one depicted in Figure 2 composed by the “Monitor Twitter Account Svc + get user status Svc + Conditional Svc + Send SMS Svc + Text to Speech Call Svc”, works as follows:
  - Keep Monitoring a Government Twitter account that publishes the information related concerts scheduled in the evening in the city,
  - As soon as this information is retrieved trigger the “get user status” Svc which retrieves the presence status of a user (e.g., by using the GTalk SDK),
If the Conditional Svc classifies such a presence status to be “at the office” then
Send an Instant Message to the user with the information related to the concert
item retrieved from the first Svc, otherwise invoke a Text to Speech Call Svc.

Mashup Pattern #3 applied to e-Government

Figure 5 Screenshot of the Monitor London’s underground Mashup

The display of data on a map enables a number of e-Government applications. For instance, the
USA Government Portal provides a dedicated area [32] where users can browse a set of Maps
displaying statistics, weather and other types of data while the Government Portal of the Italian
city of Turin [33] allows users to create their own Maps. An interesting application of this
Mashup Pattern is the composite application (see [34]) that displays the positions of the trains of
the London’s underground in real time. In this case the monitored resource is the API providing
the actual train location at a given time (see [35]). A screenshot of this Mashup is shown in
Figure 5.

Mashup Pattern #4 applied to e-Government

This Pattern related to data aggregation, and more in general to data processing, is gaining
momentum because of the availability of larger and larger sets of data published by governments
on their official portals. We distinguish two types of Mashups that can be developed according to this Pattern, namely:

- **Internal Data Aggregator Mashup**: the adjective “internal” refers to the fact that Mashups aggregate data related to the same country. Browsing, e.g., the data.gov portal it is possible to retrieve files uploaded by different agencies and aggregate them to create new data of interest, to extract statistics, etc. For example, the Ohio Civil Rights Commission created a Mashup aggregating two different data sources, namely ethnic distribution and water distribution in the city of Zanesville. The result of mashing up these two data source was the discovery of a racism case because only the houses occupied by white people were reached by the water pipes.

- **External Data Aggregator Mashup**: the adjective “external” refers to the fact that Mashups aggregate data related to different countries. This Pattern allows to combine and compare data published by different governments, as shown in the following examples:
  
  - Digest of Energy Statistics: the comparison between production and consumption of coal, electricity, gas, and oil in different countries leads to interesting results both from an economical point of view and from an environmental point of view;
  - Road accident data: the analysis of road accident information and more in general of traffic information provides a quantitative support for the development of public traffic policies and for the planning of safety organization;
  - Expenditures: the knowledge of the way public money is spent enables the development of studies and comparisons aimed at improving service efficiency. For example, the UK government recently made available the COIN (Combined Online Information System) database containing a lot of statistical expenditure data.

The external data aggregator Mashup Pattern can be successfully used to obtain interesting statistical data even though it must be recognized that at the current stage of the e-Government evolution it is not easy to take advantage of government data. The problem lies in the heterogeneity of the data available, which use different formats and different ontologies. In order to overcome such a problem the scientific community is working on ontology definition and Semantic Web technologies (e.g., Linked Data + Resource Description Framework as suggested by Tim Berners Lee in [27]). In any case it is already possible to implement such a Pattern even
if it requires the usage of additional, specific blocks to perform pre-processing operations (e.g., changing the names of some columns of a table stored in a XLS file) that allow to compare, merge, etc. data.

**Definition of a “Mashup ecosystem” and implications for the Small-Medium Enterprises**

The existence of a user-friendly Service Creation Platform and of a Service Execution Platform that support the creation and the deployment of Services and Mashups by means of friendly tools and interfaces leads to the rise of “Mashup ecosystems” in which the main actors are the Platform Operator, the end-users and the Mashup Developers, typically considered 3rd Parties with respect to platform operators and end-users. The most significant examples of similar ecosystems that involve 3rd Party developers are the Apple App Store, Amazon and eBay. In the case of the App Store the 3rd Parties provide applications, rather than services, while the App Store just makes the applications available to the end users through the network. End users buy the applications and download them in their terminals, where the applications ultimately run. In the case of both Amazon and eBay (as well as in other cases), a 3rd Party cooperation program is established between the Amazon/eBay and the 3rd Parties in force of which every time an end user accesses the Amazon/eBay portal through a 3rd Party portal and completes a commercial transaction, the 3rd Party portal receives an economic compensation.

We refer in general to the case in which the Mashups are developed by specialized 3rd Parties, typically belonging to the category of Small-Medium Enterprises (SMEs), which are the best suited candidates for these business opportunities as they exhibit the following characteristics:

- They have interest also in limited size business opportunities (e.g., of an order of magnitude of tens/hundreds thousands dollars);
- They are flexible enough to plan investments in short time frames;
- They are able to attract and hire skilled people that master service composition technologies;
- They are flexible enough to adapt their business models to those proposed by the Platform Operators.
Identification of cooperation schemes

The interaction between a 3rd Party and a Platform Operator is the key issue for the development of this kind of ecosystems. As 3rd Parties must have a clear picture of the cooperation scheme exposed by Platform Operators, the Platform Operators must establish the cooperation rules and publish a manual that reports such rules. In particular the issues upon which cooperation is based include:

- The Service Creation Platform costs for 3rd Parties: for example it may be assumed that the Service Creation Platform is provided under a free of charge basis;
- The hosting costs for 3rd Parties: for example it may be assumed that the hosting service is provided under a free of charge basis;
- The criteria to be followed to define the price that the end-users are going to pay for Mashup usage: in the case a 3rd Party implements a Mashups and deploys it in a Platform, the Platform Operator and the 3rd Party must agree on the criteria to be adopted to define such a price;
- The criteria to be followed to define the price of Services: in the case a 3rd Party implements a Service and deploys it in the Platform, the Platform Operator and the 3rd Party must agree on the criteria to be adopted to define such a price;
- The direct revenues sharing scheme in a “pay per use” framework: in the case a Mashups is used under a “pay per use” scheme, an allocation of the price paid by the end-user must be agreed. In addition the Platform Operator may have sharing schemes based on the volume of Mashups used;
- The direct revenues sharing scheme in a “flat price per bundle” framework: in the case a Service Composition is used under a “flat price per bundle” scheme, it may be the case that the contributions of the Mashups included in a bundle to the value of the bundle as a whole cannot be easily evaluated. In this case a framework must be established to this purpose by the Platform Operator;
- The indirect revenue sharing scheme: in the case a Mashup is used under a “free of charge” scheme, the possible indirect revenues (e.g. from advertisement) for the Platform Operator and for the 3rd Parties must be considered and a sharing scheme must be negotiated;
• The Mashups ranking criteria: the Platform Operator must guarantee a mechanism through which the Mashups end-users are invited to provide an evaluation of the Mashups they have used so as to support an automatic ranking system to be used by other end-users to select the services that they may need.

Market Segmentation

Market segmentation is to be performed not only by 3rd Parties but also by all the actors that participate in the Mashups value chain. One noticeable criterion to segment the market is related to the identification of user categories. The most significant categories of end users are listed in the following:

• Social Communities. For example youngsters or mature people wishing to establish and/or maintain relationships of different kinds;
• Professional Communities. For example professionals needing to interact inside enterprises and public bodies;
• Thematic Communities. For example people interested in specific themes such as for example, cinema, TV, sport, travel, books, and liking to share views, opinions and evaluations;
• Mobility Sector. For example tourists, car drivers, truck drivers and more in general people moving around and potentially needing information and services to improve the quality of their life;
• Public Service Sector. For example people working in public services, such as police, fire brigades, civil protection, health care, who may need information and services related to their specific professional activity to improve the effectiveness and the efficiency of the public service;
• Private Service Sector. For example people operating with banks, insurance and finance, who need to be constantly in touch with financial news and economic information providers to react to events promptly.
Estimation of the profits for SMEs

It is difficult to provide an analysis of so an unpredictable market as that of Mashup Ecosystems. We report some sample hypotheses, leaving the evaluation of their appropriateness to the reader. The cases are based on the following assumptions:

- No price is charged by the Platform Operator to the 3rd Parties for Service Creation/Deployment;
- No price is charged by the Platform Operator to the 3rd Party for Mashup hosting;
- The revenue sharing rate is 50% for Platform Operator and 50% for 3rd Parties (notice that such figures are different with respect to the case of the Apple App Store, where the rate is 30% - 70%).

Before analyzing some business cases of interests, we summarize the potential revenues by distinguishing two cases, namely:

- The case in which Mashup subscriptions generate revenues. Table 6 reports the revenue generated by a Mashup as a function of a flat annual amount and of the number of end-users.

<table>
<thead>
<tr>
<th>n. of end-users</th>
<th>10K end users</th>
<th>100 K end users</th>
<th>1000K end users</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 €/year</td>
<td>10 K€/year</td>
<td>100 K€/year</td>
<td>1 M€/year</td>
</tr>
<tr>
<td>5 €/year</td>
<td>50 K€/year</td>
<td>500 K€/year</td>
<td>5 M€/year</td>
</tr>
<tr>
<td>10 €/year</td>
<td>100 K€/year</td>
<td>1 M€/year</td>
<td>10 M€/year</td>
</tr>
<tr>
<td>20 €/year</td>
<td>200 K€/year</td>
<td>2 M€/year</td>
<td>20 M€/year</td>
</tr>
<tr>
<td>40 €/year</td>
<td>400 K€/year</td>
<td>4 M€/year</td>
<td>40 M€/year</td>
</tr>
<tr>
<td>100 €/year</td>
<td>1 M€/year</td>
<td>10 €/year</td>
<td>100 M€/year</td>
</tr>
</tbody>
</table>

Table 6 Flat payment schema

- The case in which each Mashup executions generate revenues. Table 7 reports the revenue generated by a Mashup as a function of a per-execution revenue and of the number of executions activated per year.
Based on the above hypotheses, we consider the following business cases.

**Business Case n. 1:** We envision a scenario in which 10K end users subscribe to a Mashup that costs them 10 €/year (corresponding to 0,83 €/month). That leads to a revenue of 100 K€/year. If we suppose that the revenue sharing ratio is 50% for Platform Operator and 50% for 3rd Party, the 3rd Party is obtaining a revenue of 50 K€/year.

**Business Case n. 2:** We envision a scenario in which 100K end users use a Mashup that costs 0,02 € each time it is executed and that each user executes the Mashup 100 times a year on average. That leads to a revenue of 200 K€/year. If we suppose that the revenue sharing ratio is 50% for Platform Operator and 50% for 3rd Party, the 3rd Party is obtaining a revenue of 100 K€/year.

**Business Case n. 3:** We envision a scenario in which 100K end users subscribe to a professional Mashup (e.g. a Business/Financial Application) that costs them 20 €/year (corresponding to 1,66 €/month). That leads to a revenue of 2 M€/year. If we suppose that the revenue sharing ratio is 50% for Platform Operator and 50% for 3rd Party, the 3rd Party is obtaining a revenue of 1 M€/year.

**Business Case n. 4:** We envision a scenario in which 500K end users (supposing for example that the Mashup is available in more than one country) use a service that costs 0,10 € each time it is executed and that each user executes the service 100 times a year on average (corresponding to an annual cost for the end user of 10 €/year). That leads to a revenue of 5 M€/year. If we suppose that the revenue sharing ratio is 50% for Platform Operator and 50% for 3rd Party, the 3rd Party is obtaining a revenue of 2.5 M€/year.

<table>
<thead>
<tr>
<th>Number of Mashups Executions/year</th>
<th>1M Exec. per year</th>
<th>10M Exec. per year</th>
<th>100M Exec. per year</th>
<th>1G Exec. per year</th>
</tr>
</thead>
<tbody>
<tr>
<td>from 100 executions/year/end user with 10K end users</td>
<td>0,01 €/exec.</td>
<td>10 K€/year</td>
<td>100 K€/year</td>
<td>1 M€/year</td>
</tr>
<tr>
<td>to 1000 executions/year/end user with 1M end users</td>
<td>0,02 €/exec.</td>
<td>20 K€/year</td>
<td>200 K€/year</td>
<td>2 M€/year</td>
</tr>
<tr>
<td>0,10 €/exec.</td>
<td>100 K€/year</td>
<td>1 M€/year</td>
<td>10 M€/year</td>
<td>100 M€/year</td>
</tr>
</tbody>
</table>

Table 7 Pay-per-use payment schema
Conclusions
We have considered the case of Server Side Mashups, that is of Composite Services (or Mashups) that run in managed platforms, and in particular have focused our analysis on Server Side Mashups.

We have examined the services used in these Mashups and have proposed a taxonomy based on service functionality and user interaction. We have then explored the main application environments in which Server Side Mashups can be used and have proposed a set of Mashup Patterns that can be taken as a reference when designing Server Side Mashups. For each Pattern we have presented examples taken from real life to help the reader to understand how the Pattern can be successfully used in real applications.

We have presented an analysis of how the proposed Mashup Patterns can be applied to the e-Government domain, which benefits from the increasing amount of data and service exposed by governments and public administrations. Although the creation of data-Mashups that merge information provided by different governments is still not trivial, it represents a very relevant trend, driven by the regulations that impose the publication of government data.

Finally we have described the organization of a “Mashup ecosystem” business scenario and have estimated the potential profits for 3rd Party SMEs that intend to invest in service creation.

References


[34] http://traintimes.org.uk:81/map/tube/