Mashup Services to Daily Activities – End-user Perspective in Designing a Consumer Mashups

Zhenzhen Zhao, Sirsha Bhattarai, Ji Liu, Noel Crespi
Institut Télécom, Télécom SudParis
91000 Evry, France
{zhenzhen.zhao, sirsha.bhattarai, ji.liu, noel.crespi}@it-sudparis.eu

ABSTRACT

Mashups have been gaining wide popularity over the past few years. Several tools and platforms exist to support user-created mashups, however working with them is still complex, and their inability to directly impact existing activities and daily lives of end-users provide little motivation for their adoption and sustained use. This paper aims to design and implement a user-centered mashup system which provides greater motivation for mashups usage, by relating every-day calendar events to useful gadgets. The system offers high level of abstraction to end users, which eliminates the need for programming and the burden of knowing about data flows from one service to the other. The platform exhibits context-orientation, personalization and socialization features which are believed to improve user experience in the system. Strong focus on functionality integration rather than data integration is believed to create greater usefulness and motivation in using the system. The system is evaluated by 131 end-users to test for usability. Also, the system is used as a representative example in proposing a user-acceptance model for consumer mashups.

Categories and Subject Descriptors

H.5.3 [**Information Interfaces and Presentation**]: Group and Organization Interfaces – *evaluation/methodology, organizational design, web-based interaction.*

General Terms

Design, Experimentation, Human Factors.

Keywords

Calendar, Consumer mashups, Event, Personalized service, User acceptance model, User-centered design, UTAUT.

1. INTRODUCTION

The future internet is envisioned as an open garden for services. Most services are created for a specific domain and their functionalities are usually limited, there is an increasing demand for composing individual and heterogeneous services into more complex or new services to meet user's needs. Mashups are web applications which combine data, content and application functionality from multiple sources, to result in a single value-added application, and have been gaining wide popularity over the past few years. Due to inherent programming difficulties required in integrating data and services from multiple sources, mashups have largely been a programmer's affairs [1]. In order to allow non-expert users to be engaged in this innovative practice, perceived usefulness and perceived ease-of-use are the primary factors to consider since they have a direct and positive impact on users to accept and use any technology.

The aim of our work is to design a "useful" and "easy-to-use" consumer mashups system for the web-savvy users without computational thinking. Followed by a literature review of overall studies on end-user's perspective on service mashups, this paper presents a consumer mashup framework based on daily activities, to firstly acquire the context information through user generated daily event, followed by the recommendation and aggregation of precisely relevant contextual services. Compare to the existing web mashup approaches, our main contributions are four-fold:

Firstly, the proposed framework strong emphasizes on functionality integration rather than data integration, in which services can communicate with daily events, in order to make the usefulness of our system apparent to users.

Secondly, the proposed framework presents a high level of abstraction to end-users, as it targets the relatively less program-savvy population. Users can visually select services from the recommended pool of services by a simple click of button, without having to worry about programming like visual data flow diagrams.

Thirdly, the proposed framework is context-oriented in the sense that the service recommendation logic is performed taking into account the overall parameters of the event details to analyze related services, i.e. the user plays a more active role in the context acquisition through event creation.

Fourthly, the proposed framework integrates social sharing and collaboration features, in which users can not only share their created events and personalized service mashups, but also participate in each other's events, which are believed to reduce the learning curve and create greater motivation in using the system, and also keep user interested in the community in the long run.

The system is evaluated by 131 end users to test for usability. Results show that users perceive the system to have high values for perceived usefulness and ease-of use. Also, the system is used as a representative example in proposing a user-acceptance model for service mashups, the model being based on the popular Unified Theory of Use and Acceptance of Technology (UTAUT) [2].

Various hypotheses are proposed to show relations between the variables proposed in the model, which ultimately are used to model 'intention to use'. Almost all proposed hypotheses have been statistically verified to be true, by means of correlation.

The paper is structured as follows. We first summarize and classify different terminologies of web mashups in Section 2. The related work on mashup developments and end-user research on consumer mashups are discussed respectively. Section 3 presents the system design principle. A usage scenario and design details are described in Section 4 and Section 5 respectively. Section 6 discusses the usability test methodology and results. Finally, Section 7 concludes the paper.

2. RELATED WORK AND MOTIVATION

2.1 Mashup Classification and Development

There has been a large interest over the past few years in mashup technologies, and many mashup development platforms/tools have emerged on a fast pace. Mashups are developed for various purposes: some are developer-centric and some are user-centric. It is useful to draw similarities and differences on the names and corresponding functionality which mashups have been given (Figure 1). There are different basis on which mashups have been classified. The first one is "*Client Mashup*" and "*Enterprise Mashup*". Client Mashups are usually created for a personal use for situational problem solving, but could be shared among peers. Enterprise Mashups are developed for problem solving in businesses and enterprise domain, and require greater collaboration among people for carrying our business processes in a coordinated manner. Often, analogous terms "*Consumer Mashup*" and "*Business Mashup*" are used to describe Client and Enterprise mashup respectively.

Another approach for mashup classifications is: "*Web Page Customization Mashup*" and "*Process Mashup*". Web page customization mashups are used to change websites by removing elements, adding additional widgets and changing the user interfaces (UIs) of websites. Process mashups allow for aggregation of data, content and services, and include them in automated sequential processes [3].

A similar way to categorize mashups is as "*Front-end Mashup*" and "*Back-end Mashup*". Front end mashups help to build web front ends like dashboards using widgets/gadgets and little to no programming (iGoogle, Netvibes, PageFlakes). Back-end Mashups combine web-accessible data and services into more useful web services that can be called easily using a RESTful interface (Kapow, Yahoo Pipes).

Another similar terminology for mashup classification is "*Horizontal Mashup*" and "*Vertical Mashup*". Horizontal mashups can be seen as a process of grouping sets of typically similar or complementing services to aggregate their outputs, in which there is no interaction between service modules, but the customizable front-end offers more value to the end-users to solve a particular task. Vertical mashup, on the other hand, is a process of orchestrating outputs from some services into the inputs of other services, where service modules are connected together, and the parameters are passed between the modules to get a new enhanced service.

Extensive research works have been done in web page customization mashups, process mashups and enterprise mashups.

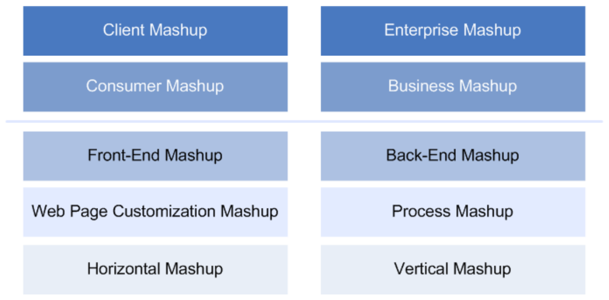


Figure 1. Mashup classification.

In front-end side, authors in [4] design a university-oriented personalizable web 2.0 mashup portal called iNIU, the portal exploits the Web 2.0 technology to mash-up a variety of existing Web services that NIU students frequently use, including Blog, Weather, Clock, Calendar, News, Facebook Profile Viewer, and Search. In [5], authors propose the idea of gadget creation so that extracted data can be immediately reused on personal portals through an unsupervised web data extraction approach. [6] proposes a widget based service exposure and service creation tool: a tool creates links between loaded widgets automatically, while additional functionalities are added automatically to existing widgets as long as the end-user loads other widgets to his personal environment.

To make Process Mashup development simpler and to enable even non-experienced end-users to mash up their own web applications, a number of development tools and frameworks have been proposed. [7] introduces a recommendation tool called MashupAdvisor. Based on the current state of a mashup, the MashupAdvisor exploits a repository of mashups to estimate the popularity of specific outputs (goals), and makes suggestions using the conditional probability that an output will be included. When a suggestion is accepted, MashupAdvisor uses a semantic matching algorithm and a metric planner to modify the mashup to produce the suggested output. Similarly, [8] presents an efficient syntactical approach for actively discovering web service candidates for service mashups. The authors use syntactical, natural language approaches to predict when the underlying web services messages are related.

In Enterprise Mashup domain, authors in [9] discuss the design principles of the Enterprise mashup architecture, upcoming intermediaries and mass collaboration. The same authors in the following year propose a web based mashup/gadget development tool that allows for different options to realize Business to Business (B2B) collaborations via mashups. In their work, five patterns for the development of enterprise mashups are identified and characterized [10]. [11] proposes a new widget aggregator that enables the end-user to personalize a business process by chaining widgets according to his/her needs and habits without computing skills.

2.2 End-user Studies on Consumer Mashups

Several, but not abundant studies have been performed in user research. Such research starts from reviewing the current mashup tools. Survey from [12] states that not all of these tools are easy to understand and use, especially for normal end-users who lack the art of computational thinking. [3] identifies six aspects of mashups, from an end user perspective, to review current tools, viz: Levels of abstraction, Learning support, Community features, Searchability, UI design, and Software Engineering techniques. For less technical users, generally a high level of abstraction, abundant learning support, and community features are desired.

[12] further states that end users do see benefits in mashups for searching, integrating, and sharing information. The authors present the term "web-active user" to define users who are considered to be active online. They are users who use internet on a daily basis, and who try to find out new ways to integrate their online activities, although they don't have the programming expertise to create mashups. A survey [13] was conducted with over 200 students who were considered to be web-active. When explained about mashups, the users gave the feedback that usefulness is more important to them than the perceived difficulty of action.

[14] identifies two major factors - "Usefulness" and "Technology initiative" from an end-user survey to have an effect on user's motivation to create mashups. The study has found that people with different levels of technology initiative have interest in creating and using different types of mashups, with low-initiative users preferring mashups related to people and social activity, and high-initiative users preferring more complex data mashups.

2.3 Limitations and Motivations

From the above studies we can conclude that: on the one hand, process mashup is still considered quite complicated and discouraging for non-expert end-users. Survey from [14] states that among the three processes of composing mashups, viz. data gathering, data manipulation and data presentation, end-users usually find the data manipulation stage most confusing. In the development of the markets, process mashup systems are far from being popularized among the ordinary users due to the complexity of understanding data flows between the services. End-users who lack the art of computational thinking [14] are not able to fully leverage the value and benefits of mashups which require the use of APIs, RESTful services, Atom and RSS feeds. In order to achieve a greater user motivation for mashups use, there is the need for existing mashup platforms to offer solutions by bringing enough value to their existing activities and to meet their daily life needs [9], while offering greater simplicity and usability features.

On the other hand, web page customization mashup systems are seen to provide a convenient way for users to aggregate selected services, referred to as widgets or gadgets in their personal dashboard for the creation of a personalized environment. These methods, however, lack flexibility since the gadgets cannot communicate with each other or with any other web service. Moreover, such systems exhibit large service databases and often permit access to third party for increased system functionality, which is not necessarily providing a better solution and quality of experience for the user. Often, when services are aggregated, users need to search for required services in a pool of services, including many of them are not particularly useful.

Concerning user research on mashups, although user studies have been made for development of mashup tools, little research endeavors have examined the needs of the less programming savvy end-users [12]. Moreover, formal model for user acceptance of mashup technology is still missing.

Our work aims to design and implement a user friendly mashup platform which provides greater motivation for mashups usage, by relating every-day calendar events to useful gadgets. The platform offers high level of abstraction to end users, which eliminates the need for programming and the burden of knowing about data flows from one service to the other. The platform exhibits context-orientation, personalization and socialization features which are believed to improve user experience in the system. Strong focus on functionality integration rather than data integration is believed to create greater usefulness and motivation in using the system. Also, the system has been used as a representative example in proposing a user-acceptance model for service mashups, targeting the relatively less program-savvy population.

3. DESIGN PRINCIPLE

This section focuses on the design principle of our system. The basic research question in this paper is "How end-user perspectives help in designing a useful and easy-to-use consumer mashups?" We have learned the end-user perception from the related user research that: on the usefulness side, end-users require a mashup system designed to offer solutions by bringing value in organizing end-users day-to-day and social activities [9][14]; while on the ease-of-use side, comparing to "vertical mashups", the "horizontal mashups" is more accepted by the less programming user group.

Followed by the end-users' perceptions, we direct and propose the system design principles as follows. Note that we target the group of less technology savvy end-user.

Usefulness: Consumer mashup framework is designed to organize users' day-to-day and social activities through life events.

Ease-of-use: Consumer mashup framework is designed as web page customization mashups at the presentation layer.

We add another term - intuitiveness - as a separate feature from ease-of-use, which is reported to be one of the powerful factors and almost a key for the success of consumer mashup system since it fosters real engagement of users [15].

Intuitiveness: The intuitiveness of consumer mashup system is improved by using the light-weight applications or components of user interface, in the form of gadgets, instead of service APIs.

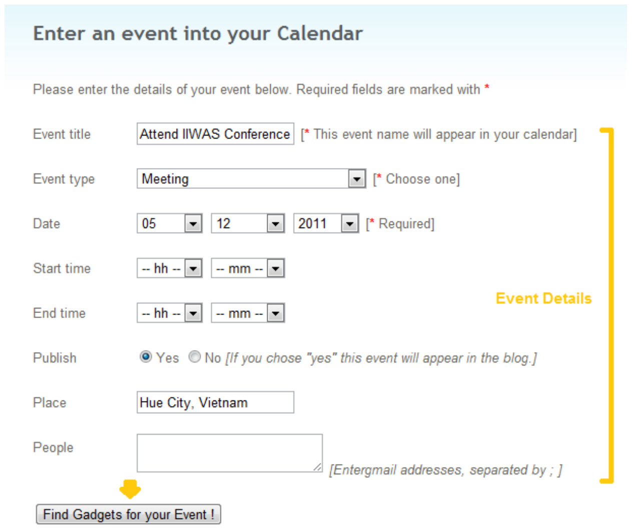
Gadgets, also known as widgets, are mini applications which provide a graphical, simple and efficient means of user interaction with the actual web resources (data, content, or application functionality, e.g. text/multimedia content, RSS feeds etc), abstracting the technical description from the functionality.

4. DEMONSTRATION SCENARIO

In this section we introduce our system - event based service provider (EBSP), a "Calendar-based Mashup" which makes use of the Google Calendar and iGoogle gadgets in providing valuable solutions in organizing day-to-day activities of end users, such as meeting, travelling, shopping, cooking, sports, games and many more. This system attempts to bring value to meet the daily life needs of the web-savvy yet non-programmer end-users.

Suppose a Google subscriber Fiona, is going to attend IIWAS 2011 conference, she decides to use EBSP system to organize her trip. The various steps involved in using the system are shown below:

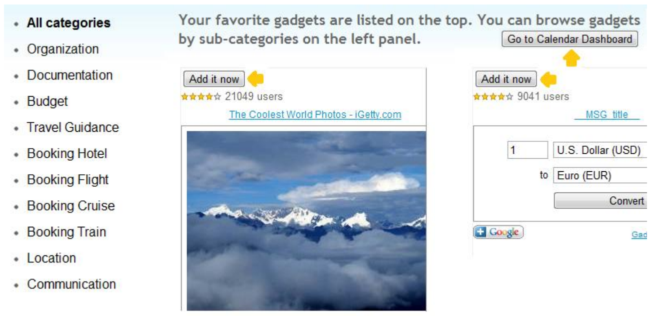
1. Fiona firstly browses the events blog, to see other users' activities. She doesn't find anything related, so she decides to create an event by her own.


Enter an event into your Calendar

Please enter the details of your event below. Required fields are marked with *

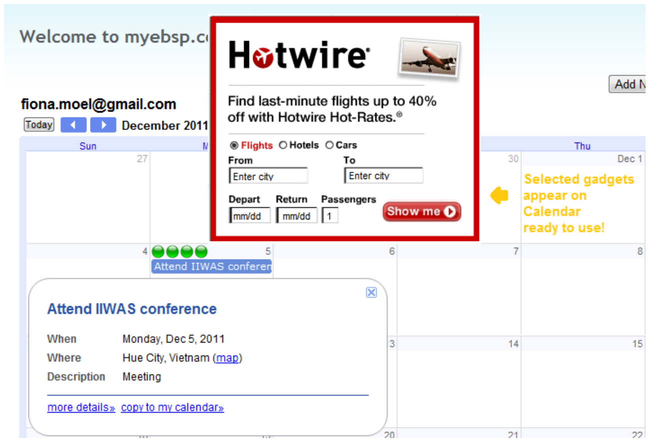
Event title	Attend IIWAS Conference	[* This event name will appear in your calendar]
Event type	Meeting	[* Choose one]
Date	05 / 12 / 2011	[* Required]
Start time	-- hh -- / -- mm --	
End time	-- hh -- / -- mm --	
Publish	[✓ Yes] No	[If you chose "yes" this event will appear in the blog.]
Place	Hue City, Vietnam	
People		[Entergmail addresses, separated by ;]

Figure 2. User input in the event details.


- All categories
- Organization
- Documentation
- Budget
- Travel Guidance
- Booking Hotel
- Booking Flight
- Booking Cruise
- Booking Train
- Location
- Communication

Your favorite gadgets are listed on the top. You can browse gadgets by sub-categories on the left panel.

Figure 3. Service recommendation and user selection.



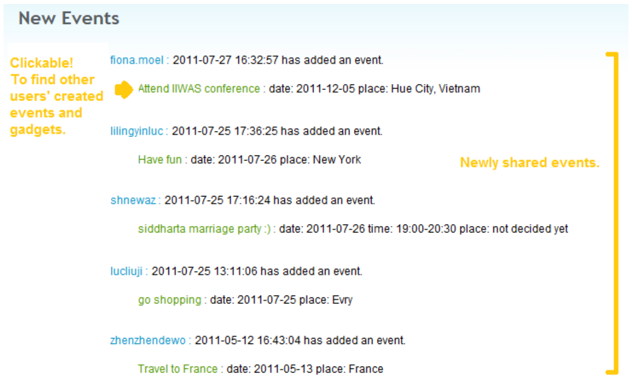
Attend IIWAS conference

When	Monday, Dec 5, 2011
Where	Hue City, Vietnam (map)
Description	Meeting

more details» copy to my calendar»

Figure 4. Service mashups on the calendar.

2. Once she starts to add an event, Google asks her to log in and grant access to the EBSP system.
3. Fiona enters the details of the event "Attend IIWAS conference" and agrees to publish this event to the public (Figure 2). Then she clicks "Find Gadgets for your Event!"


New Events

fiona.moel : 2011-07-27 16:32:57 has added an event.
Attend IIWAS conference : date: 2011-12-05 place: Hue City, Vietnam

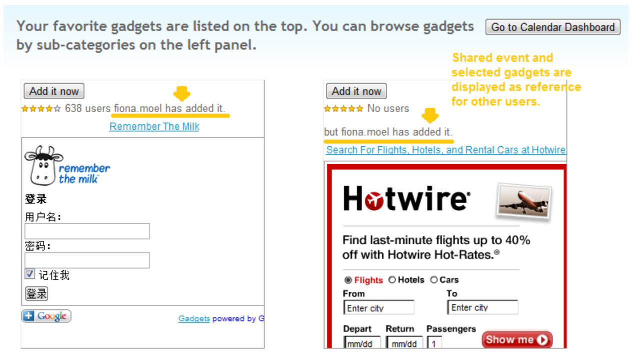
lilingyinluc : 2011-07-25 17:36:25 has added an event.
Have fun : date: 2011-07-26 place: New York

shnewaz : 2011-07-25 17:16:24 has added an event.
siddharta marriage party :) : date: 2011-07-26 time: 19:00-20:30 place: not decided yet

lucliuji : 2011-07-25 13:11:06 has added an event.
go shopping : date: 2011-07-25 place: Evry

zhenzhendewo : 2011-05-12 16:43:04 has added an event.
Travel to France : date: 2011-05-13 place: France

Figure 5. Public events blog.


Your favorite gadgets are listed on the top. You can browse gadgets by sub-categories on the left panel.

Go to Calendar Dashboard
Shared event and selected gadgets are displayed as reference for other users.
Add it now
638 users fiona.moel has added it.
Remember The Milk
remember the milk
登录
用户名:
密码:
记住我
登录
Gadgets powered by G
Add it now
No users
but fiona.moel has added it.
Search For Flights, Hotels, and Rental Cars at Hotwire
Hotwire
Find last-minute flights up to 40% off with Hotwire Hot-Rates.®
Flights Hotels Cars
From
To
Enter city
Depart
Return
Passengers
mm/dd
Show me

Figure 6. Gadgets Reuse from Previous Events.

4. The system automatically recommends her the gadgets (Figure 3). Gadgets are sorted according to individual sub categories in the left panel and displayed on the right side in an order regarding the gadget rating, user installation number and Fiona's gadget selection history.
5. Fiona starts to select the precise gadgets of interest by clicking on the "Add it now" button. After selection of gadgets, Fiona clicks on "Go to Calendar Dashboard".
6. The selected gadgets are mashed up on the Google Calendar (shown as circular green icons on top of the specified event, Figure 4), and can be accessed and used directly from the calendar.
7. Since Fiona has agreed to share this event, the "Attend IIWAS conference" event is published to the events blog for other users' reference (Figure 5).
8. Vincent, who plans to attend the same conference, happens to find Fiona's "Attend IIWAS conference" activity from events blog. He creates the event by copying Fiona's event details with a simple click. He is able to reuse Fiona's selected gadgets or choose his own gadgets of interest (Figure 6).

5. DESIGN DETAILS

5.1 System Framework

Based on the design principles, this section presents a brief overview of the principal components of the system and their functionalities. Figure 7 shows the architecture of the system, which is fundamentally composed of four layers: Presentation, Logic, Data access and Data. Mashup creation is done in the

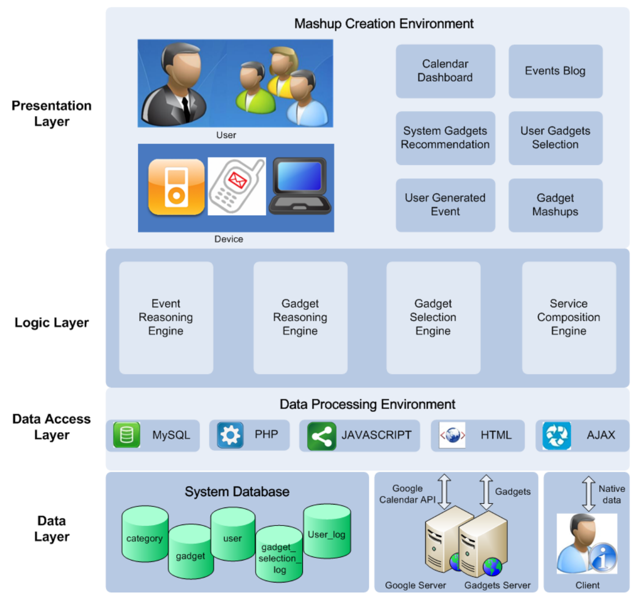


Figure 7. System framework.

presentation layer. A user has two approaches to access the event creation environment, either from direct event input or copying events created by other users*. When he/she uses his/her device to enter event details or copy event details from other user, the input data (user identity, user entered data or old event data*) is passed to the data access layer, where the data is processed, and further sent to logic layer, where reasoning is done. Based on this, resources are looked up from the Data layer, which is a layer for aggregation of resources (Data).

In Data layer, reside the gadgets imported to the local system database from the iGoogle database. The logic layer extracts from the data access layer, the sorted gadgets according to several rules: 1) based on the event category 2) based on user's gadget selection history 3) based on gadget rating (star) and user number 4) based on gadgets selected in inherited event*.

These gadgets are pooled, and sent to the presentation layer, as service recommendation to the user. Of the recommended services, the user selects the gadgets of interest. The selection of gadgets sends a trigger to the logic layer to record the selection for that user, and update it to the system database (gadget selection log and user log). The finalization of service selection triggers the event and gadget reasoning engine to communicate with the Google server, using Google calendar API, to upload the selected gadgets to user's Google calendar. The Google server sends the uploaded gadgets to the presentation layer for display to the user. At the same time, the current event and selected gadgets are shared in the public events blog. The aggregation of selected gadgets will allow end users to create a mashups application. The mashup is done in a visually intuitive manner, without skills being required for programming.

5.2 Process Flow in the System

The process flow in the system is shown by means of a flowchart in Figure 8. When a user connects to our system and starts creating an event, the first process is authorization for Google Calendar Service. The OAuth authorization method [16] is invoked during this process. After authentication, a single-use OAuth token is

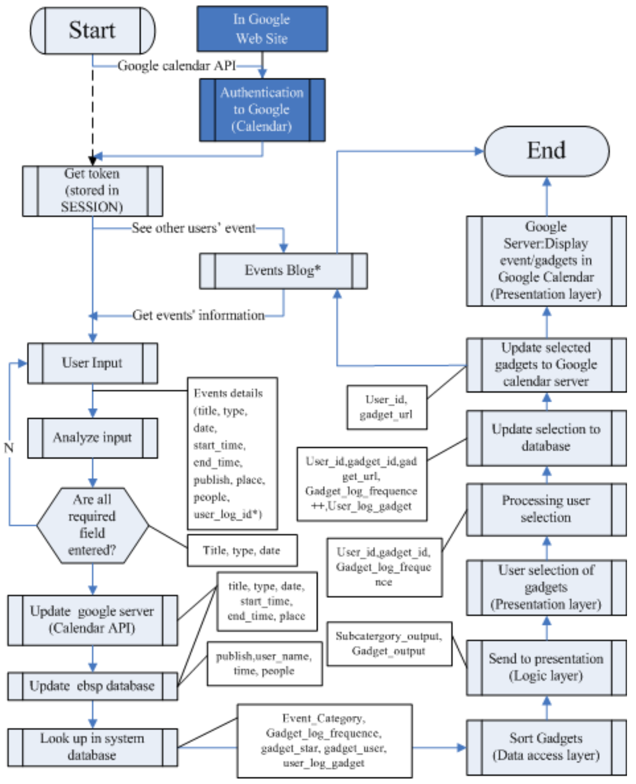

Start
In Google Web Site
Google calendar API
Authentication to Google (Calendar)
Get token (stored in SESSION)
See other users' event
Events Blog*
Get events' information
User Input
Analyze input
N
Are all required field entered?
Events details (title, type, date, start_time, end_time, publish, place, people, user_log_id*)
Title, type, date
Update google server (Calendar API)
title, type, date, start_time, end_time, place
Update ebsp database
publish,user_name, time, people
Look up in system database
Event_Category, Gadget_log_frequence, gadget_star, gadget_user, user_log_gadget
Sort Gadgets (Data access layer)
Send to presentation (Logic layer)
Subcatergory_output, Gadget_output
User selection of gadgets (Presentation layer)
User_id,gadget_id, Gadget_log_frequence
Processing user selection
User_id,gadget_id,gadget_url, Gadget_log_frequence ++,User_log_gadget
Update selection to database
User_id, gadget_url
Update selected gadgets to Google calendar server
Google Server:Display event/gadgets in Google Calendar (Presentation layer)
End

Figure 8. Flowchart of the system process.

made available to our system. This token made for calendar application when the user gets logged in and is later used or the communication as user authorization between our system and the calendar API.

Now the system is in ready to use state. The user enters or copies* the details of the event, and input parameters (title, type, date, start_time, end_time, place, publish, people and user_id) are passed to the data access layer. If all required fields for the event are entered, and the event information for this event are correct, the event is regarded as a simple event and will be uploaded to Google calendar through the calendar API and updated the system database as well as Events blog, when the user clicks on 'Find Gadgets for your event!".

At the same time, based on the given event context (user input), this layer looks up the corresponding event category, user's history of gadgets (gadget_log_frequence), gadget rating (gadget_star), gadget installation number (gadget_user) and gadgets used in the copied event* (user_log_gadget). For a new user, initially there will be no record of gadget selection log and user log, but the system will update usage once the person starts selecting gadgets. For a user having previous experience using the system, the data access layer dynamically computes the subcategories that are to be output (subcategory_output) along with appropriate gadgets (gadget_output) (in logic layer). The discovery of appropriate gadgets ("service discovery") is done though a hierarchical directory-based-search mechanism, described in section 5.3.

The computed subcategories and gadgets are then sent to the presentation layer, this process referred to as "gadget recommendation". The user can select gadgets of interest. Google calendar API is used to tunnel the selected gadgets to the calendar via a piping mechanism (executed by service composition engine in logic layer but proceed in data access layer). The gadgets are then sent by the Google server from the data layer to presentation layer of the calendar for display. The uploading of gadgets into calendar is achieved in real time, the mashup phenomenon is done in the presentation layer, and the events blog is updated, by a simple click of the button "Go to Calendar Dashboard".

Meanwhile, in our system, when the user selects one or more gadgets, user's selection is recorded, and for that user (user_id), the selected gadgets (gadget_id) are identified. The frequency of use (gadget_log_frequence) is increased by once per selection and updated to the gadget selection log database and also recorded in user log.

5.3 Matching Between Event and Gadgets

To look up the exact gadgets of interest in the system database (service discovery), a hierarchical, directory-based search has been implemented. Before describing the search mechanism, the framework of an event hierarchy is presented.

5.3.1 *Event Hierarchy*

Since our system aims at bringing the concept of event to explore the user intention for service integration and management, the first challenge is how to define event in an efficient way to retrieve and organize relevant services, i.e. the functional description of the event. In current event-based system, the related event elements are nothing less than event theme, occurrence place, occurrence time and involved people, which can be expressed as *what*, *where*, *when* and *who*. In our approach, we follow the same definition of event elements. Each event element comprises a hierarchy of related information organized in a dependent manner. Each element (attribute) of the event answers a question inside the event. Each question is related to the user's goal, which is further associated with the functional description of the event to retrieve relevant services (see Table 1). *What* defines the user's main objective, which is associated with the event type/category; *Where* is associated with location and presence service; *When* is functionally related to the time based service and notification service; and finally, *Who* defines whether the event is a personal or a social event, which is associated with personalized service, communication and social service. Note that the event attributes are regarded as the first layer of the event hierarchy.

Table 1. Functional description of Event

Event attributes	Questions	Functional descriptions
What	What's the type of the event you deal with?	Event type/category
When	When is the event happen?	Time Notification
Where	Where do you carry out this event?	Location Presense
Who	Who else participate in this event?	Social Communication

The implementation of the event hierarchy is shown in Figure 9. The functionalities for "*who*" are based on Google calendar's functionalities like inviting people, sharing calendars (events and gadgets), and also system-based functionalities like social/communication. The functionalities for "*where*" are based on Google calendar's functionalities like 'Google Map', and also

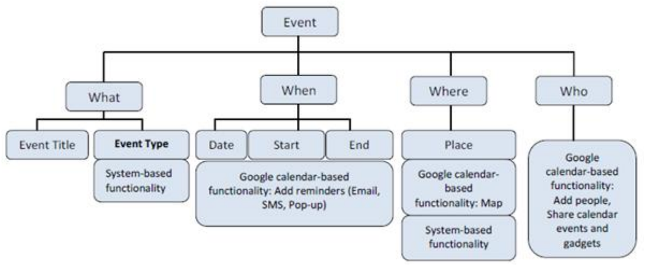

Event
What
When
Where
Who
Event Title
Event Type
System-based functionality
Date
Start
End
Google calendar-based functionality: Add reminders (Email, SMS, Pop-up)
Place
Google calendar-based functionality: Map
System-based functionality
Google calendar-based functionality: Add people, Share calendar events and gadgets

Figure 9. The implementation of event hierarchy.

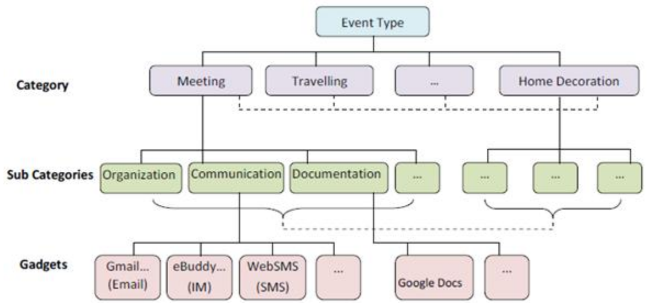


Figure 10. An example of event category hierarchy.

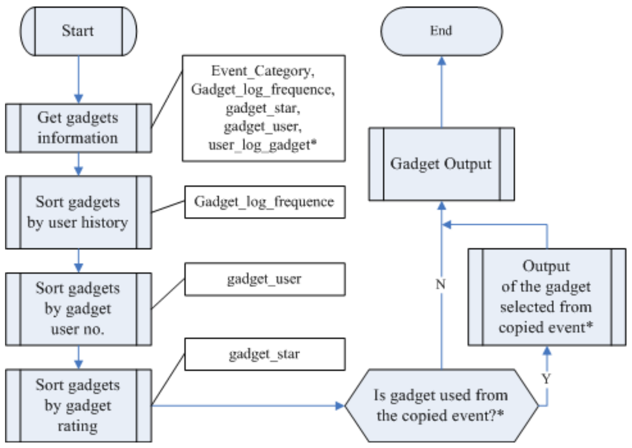

Start
End
Get gadgets information
Event_Category, Gadget_log_frequence, gadget_star, gadget_user, user_log_gadget*
Sort gadgets by user history
Gadget_log_frequence
Sort gadgets by gadget user no.
gadget_user
Sort gadgets by gadget rating
gadget_star
Gadget Output
Output of the gadget selected from copied event*
N
Y
Is gadget used from the copied event?*

Figure 11. Flowchart of gadget recommendation process.

on system-based location/presence functionalities (like driving directions, route planning, weather information etc). The functionalities for "*when*" are based on Google calendar's functionality of adding reminders/notifications for the event (email, SMS, pop-up). The functionality for "*what*" and specifically "event category" is the most significant in our system. In providing relevant services (gadgets) in response to the input, the event will be composed not only of the four attributes (*what*, *when*, *where* and *who*) but one more attribute will be added to it, i.e. "*how*", meaning how can the event be carried out, provided that relevant information are given by the recommended gadgets.

5.3.2 *Event Category*

Among those attributes the event category is the most important factors in choosing related services, which is defined further firstly by the related functional requirements, and then the specific services. The contributions of this event category hierarchy are two-fold. On the one hand, it defines the useful functionality inside each event activity/type for purposes of filtering out less useful or useless services, i.e. increase the accuracy of retrieved services; on the other hand, it provides the relationships among different events, thereby enabling reusability of the functionality for different events.

An example of event category is shown in Figure 10. The category is prepared by trying to encompass diverse forms of activities and events in daily lives of people, spanning different hobbies, interests and professions, which are available in the iGoogle gadgets database. In some cases, one category maybe reused in the other, for e.g. “Meeting” can require “Travelling”, hence “Travelling” maybe reused in the former. The event type entered by the user is matched with one of these categories. On finding the appropriate category, a second level directory of “sub-categories” is opened, which are functionalities required for the corresponding upper layer category (e.g. organization, communication, documentation functionalities may be required for the event category meeting).

5.3.3 Gadget Discovery and Recommendation

The final gadgets are discovered via a hierarchical directory-based search mechanism, based on the input for event hierarchy. After the gadget discovery, the gadget recommendation (i.e. the order of the gadgets displayed in the webpage) is reasoned according to three rules: 1) based on user’s gadget selection history 2) based on gadget rating and gadget user number 3) based on the gadgets selected in the inherited event*. The flowchart of this process is shown in Figure 11.

5.4 System Database

In total, there are five local databases in the system: event category database (for categories like meeting, shopping, travelling etc), gadgets database (the actual gadgets, user installation number and gadget rating), user database (user id), gadget selection log (user id and gadget selection history) and user log (user created event and selected gadgets information) database. The entity-relationship (ER) diagram for our system database is shown in Figure 12, using the *Crow’s feet notation*. Here, category, gadget, gadget_selection_log, user and user_log are the entities, and the lines show the relationship between the entities.

5.5 Automatic System Update

Originally the iGoogle gadgets stored in our system database are static. The drawbacks are obvious: The gadget appears unavailable without being detected, gadget rating and user number are not updated, while new gadgets are not discovered. Hence, we need to build a dynamic system to update the gadget information timely. Here, a library called “cURL” is used to get the web page information from Google server. We search gadget by semantic key words defined in each subcategory. Here, WordNet [17] is used to find the synonyms of each keyword, and the set of keywords together are used to perform gadget search. After processing the page information, the gadget details (name, script, url, height, width, star, user number) are retrieved and recorded in our system database in the corresponding category. This process is performed periodically to update all the gadget information in the database as well as add new gadgets.

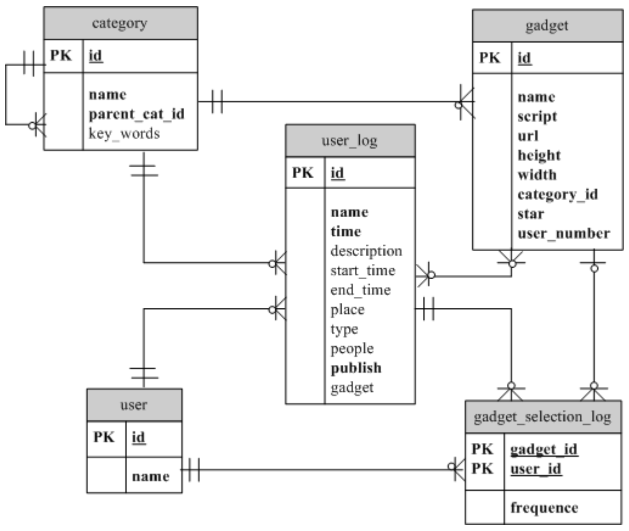


Figure 12. ER Diagram for the system database.

6. USABILITY TEST

This section introduces the usability test of EBSP system. We intend to evaluate the EBSP system by letting end-users use our system, and collect their feedback online. To this regard, a questionnaire is designed to gather user’s perceptions about our system, and service mashups as a whole. Before letting the users experience with our system, we provided a brief video tutorial about commercial mashup tools (iGoogle, Yahoo! Pipes, Popfly and Intel Mashmaker) to make them familiar with mashup technology. Then we provided a brief video tutorial on the EBSP system, with various examples. Users are asked to complete a given scenario, based on the tutorial, and rate their experience with the system.

The questions used a five-point Likert scale, from Strongly disagree to Strongly agree, such that Strongly disagree =1, disagree=2, Undecided=3, Agree=4 and Strongly agree=5. A total of 423 requests are sent out online, of which we received 233 partial responses. Only 131 were complete and valid for our study.

6.1 Usability Metrics

Measuring the usability of a system is not always obvious. Several parameters and methods have been applied in the current work to define usability of the system, like performance expectancy, effort expectancy, time on task, scenario completion percentage, error percentage, and overall feedback.

6.2 User Acceptance Model

User acceptance models are also used to model usability of a system, specifically by identifying factors or constructs that play an important role in user’s intention to use a system. The system is used as a representative example in proposing a user-acceptance model for consumer mashups, to precisely identify what factors lead to their adoption and to what extent. The proposed model is constructed based on the UTAUT model, which is a consolidated model derived from eight different models, and reported to outperform the performance of each one in explaining user acceptance for an IT system or product [2].

Results show that users perceive the system to have high values for perceived usefulness and ease-of-use. The results of the hypotheses set in the study show support to the UTAUT model. The fact that only 36.6% of the variance in behavioral intention has been explained suggests the need to further refine the model by incorporating unmeasured variables. Nonetheless, this study can provide useful directions for user acceptance of mashups. Through the results of the study, we recommend mashup developers who target the less programming savvy user group to pay particular focus in bringing value to organizing better existing day-to-day activities of average end-users rather than focusing on complex feature extensibility in the platform. More information about the usability test results and user acceptance model can be seen from [18].

7. CONCLUSION

In this paper, we have presented a user friendly consumer mashups framework to manage multiple resources in a single platform though life events. The contributions of event hierarchy, context association and functionality integration have been illustrated respectively. We have further implemented a prototype and evaluated our solution by designing a user acceptance model. The usability test has showed that our system outperforms the classical approaches in both usefulness and ease-of-use attributes. The presented model can serve as a reference for mashup designers to design platforms having better user acceptance.

8. ACKNOWLEDGMENTS

Our sincere thanks to Mr. Hui Wang and Mr. Honguang Zhang for their efforts on system construction. This work is carried out in the frame of the Eureka-ITEA 2 "Do-it-Yourself Smart Experiences" project. It is supported in part by the French Ministry of Industry.

9. REFERENCES

[1] Liu, X., Hui, Y., Sun, W., and Liang, H. 2007. Towards Service Composition Based on Mashup. In *Proceedings of the IEEE Congress on Services 2007* (Salt Lake City, Utah, USA, July 09-13, 2007). SERVICES ' 07. IEEE, 332-339.

[2] Venkatesh, V., Morris, M. G., Davis, B., and Davis, F. 2003. User Acceptance of Information Technology: Toward a unified view. MIS Quarterly, 27, 425-478.

[3] Grammel, L., and Storey, M. A. 2008. An End User Perspective on Mashup Makers. University of Victoria Technical Report DCS-324-IR.

[4] Zhang, J., Karim, M., Akula, K., and Ariga, R. 2008. Design and Development of a University-Oriented Personalizable Web 2.0 Mashup Portal. In *Proceedings of IEEE International Conference on Web Services* (Beijing, China, September 23-26, 2008). ICWS '08. IEEE, 417-424.

[5] Chang, C. H., Yang, S. F., Liou, C. M., and Kayed, M. 2008. Gadget creation for personal information integration on web portals. In *proceedings of IEEE International Conference on Information Reuse and Integration* (Las Vegas, Nevada, USA, July 13-15, 2008). IRI '08. IEEE, 469-472.

[6] Laga, N., Bertin, E., and Crespi, N. 2009. Building a User Friendly Service Dashboard: Automatic and Non-intrusive Chaining between Widgets. In *proceedings of IEEE Congress on Services - Part I* (Los Angeles, CA, USA, July 06-10, 2009). SERVICES '09. IEEE, 484-491.

[7] Elmeleegy, H., Ivan, A., Akkiraju, R., and Goodwin, R. 2008. Mashup Advisor: A Recommendation Tool for Mashup Development. In *Proceedings of IEEE International Conference on Web Services* (Beijing, China, September 23-26, 2008). ICWS '08. IEEE, 337-344.

[8] Blake, M. B., and Nowlan, M. F. 2008. Predicting Service Mashup Candidates Using Enhanced Syntactical Message Management. In *proceedings of IEEE International Conference on Services Computing* (Honolulu, Hawaii, USA, July 8-11, 2008). SCC '08. IEEE, 229-236.

[9] Hoyer, V., Stanoesvka-Slabeva, K., Janner. T., and Schroth, C. 2008. Enterprise Mashups: Design Principles towards the Long Tail of User Needs. In *proceedings of IEEE International Conference on Services Computing* (Honolulu, Hawaii, USA, July 8-11, 2008). SCC '08. IEEE, 601-602.

[10] Janner, T., Siebeck, R., Schroth, C., and Hoyer, V. 2009. Patterns for Enterprise Mashups in B2B Collaborations to Foster Lightweight Composition and End User Development. In *Proceedings of IEEE International Conference on Web Services* (Los Angeles, CA, USA, July 06-10, 2009). ICWS '09. IEEE, 976-983.

[11] Laga, N., Bertin, E., and Crespi, N. 2010. Business Process Personalization Through Web Widgets. In *Proceedings of IEEE International Conference on Web Services* (Miami, Florida, USA, July 05-10, 2010). ICWS '10. IEEE, 551-558.

[12] Zang, N., and Rosson, M.B. 2008. What's in a mashup? And why? Studying the perceptions of web-active end users. In *proceedings of IEEE Symposium on Visual Languages and Human-Centric Computing* (Herrsching am Ammersee, September 15-19, 2008). VL/HCC '08. IEEE, 31-38.

[13] Zang, N. 2008. Mashups for the web-active user. In *proceedings of IEEE Symposium on Visual Languages and Human-Centric Computing* (Herrsching am Ammersee, September 15-19, 2008). VL/HCC '08. IEEE, 276-277.

[14] Zang, N. 2009. Mashups on the Web: End User Programming Opportunities and Challenges. In *the proceedings of First Workshop on Evaluation and Usability of Programming Languages and Tools* (Orlando, FL, USA, October 25-29, 2009). PLATEAU '09.

[15] Zhao, Z., Laga, N., and Crespi, N. 2010. The Incoming Trends of End-user Driven Service Creation. Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering (LNICST), Volume 21. Springer-Verlag, 98-108.

[16] OAuth 1.0: http://code.google.com/apis/accounts/docs/OAuth.html

[17] WordNet: http://wordnet.princeton.edu/

[18] Bhattarai, S., Zhao, Z., and Crespi, N. Consumer Mashups: End-User Perspectives and Acceptance Model. In the proceedings of information integration and web-based applications & services (Paris, France, November 8-10, 2010). IIWAS '10. ACM, 930-933.