Towards TomTom like Systems for the Web

A Novel Architecture for Browser-based Mashups

Emilian Pascalau
Hasso Plattner Institute, University of Potsdam
Prof.-Dr.-Helmert-Str. 2-3
Potsdam, Germany
emilian.pascalau@hpi.uni-potsdam.de

ABSTRACT

Business Intelligence (BI) for the new economy requires people to take active part and do by themselves BI development tasks from within their browsers. With the great progress of Web 2.0 into the mainstream the perspective of the BI development has widened and a new set of characteristics drive it: new business models, changing customer relationship, software on demand, instant use, and deep architectural impact. Mashups that emerge from the cloud computing are available in various vendors’ BI environments and allow users (although restricted by the features these systems support) to develop their BI tasks.

This paper introduces a new architecture for browser based mashups that inherits ideas from TomTom systems. This architecture is capable to address issues such as business intelligence on demand, instant use and to offer the same degree of generality as the browser.

Categories and Subject Descriptors

D.2.11 [Software]: Software Engineering Software Architectures; H.4.3 [Information Systems Applications]: Communication Applications—Information browsers; H.1.2 [Models and Principles]: User/Machine Systems—Human factors

General Terms

Design, Human Factors

Keywords

Business Intelligence, Mashups, Browser, Architecture, TomTom

1. INTRODUCTION

Future Internet as denoted by Zittrain [16] is a generative environment that fosters innovation, through the advance of technologies and a shift in people’s perception of the Web and a paradigm shift in how they use it. The Web 2.0 [11] is no longer a bleeding edge but rather a leading edge now [14]. Mashups are one of the paradigms that emerged from the Web 2.0 movement and according to [2] a mashup is a web application that combines content from two or more applications to create a new application. Situational applications are enterprise web applications built on-the-fly to solve a specific business problem.

Mashups have started to play an important role also in the Business Intelligence context [8] as the BI professionals face resource constraints and the soft economy keeps budgets tight. Self-service is at the heart of BI for the new economy. For the BI people, self-service means speedy answers, single view of everything, single version of truth, self-service information exploration [8]. People are required to take active part and to do by themselves BI development tasks from within their browsers.

SAP executive vice president of Small and Midsize Enterprises, Peter Lorenz, in his talk “The SAP Vision on Cloud Computing”1 emphasized that users, easy consumption, software on demand, instant use but powerful mean a deep architectural impact on how software has to be developed in the future.

Unlike traditional technologies such as ERP and CRM where users mostly process information in the form of reports or execute transactions, Web 2.0 technologies are interactive and require users to generate new information and content or to edit the work of other participants [4].

Next generation software [11] will have people not processes [5] at its heart and will have to tackle software on demand, business intelligence on demand, instant use but powerful, innovation and creativity. This paper introduces a new architecture for browser-based mashups that strives to tackle previously enumerated aspects. Therefore for the architecture presented in this paper, the user is in the same time user and component of the system itself. The architecture inherits characteristics from TomTom systems.

2. RELATED WORK

The term mashup is borrowed from pop music, where it denotes remixing songs or parts of songs to deliver new derivative works. Similar to this, Web mashups remix content from the Web and deliver new results, often insight among disparate and originally independent sets of information and function, and thus create innovation through assembly.

1http://www.tele-task.de/archive/lecture/overview/5014/
From a technological point of view mashups are hybrid web applications, usually found under the association of SOA plus REST principles [12]. Typically, mashed-up applications are created by means of lightweight components - known as widgets or gadgets. These components have to be previously created, in order to be used. Their execution is platform depended and the platforms are server side. The widgets/gadgets approach is thought to be a client side technology but because gadgets are executed on the server side the author considers them to be hybrid client-server approaches.

Although there is work that concerns architecture for mashup platforms (see for instance [7]), that work does not address pure mashups as the approach introduced in the following sections. According to [3] “pure” mashups are client based mashups; however as the paper [3] continues to state those are much more difficult to achieve in the browser.

Nevertheless approaches such as mashArt [6] and rule based execution of mashups [12] address mashups from a higher perspective.

The approach introduced in this paper differs from the current mashup landscape in several ways: (1) tackles fully fledged browser-based mashups; (2) the user is an active part of the system; (3) the architecture inherits characteristics from TomTom systems; (4) utilizes a cognitive context aware engine; (5) tackles software and business intelligence on demand. The author is not aware of any other architectural approaches similar with the one introduced in this paper. Basic ideas towards this approach have been already discussed by the author in [12].

3. USE CASE

For scientists in the filed of IT the DbWorld Mailing List is the well known place where they can search for an IT conference. A series of information are provided here, but most important are subject, deadline and web page of the published event. From a technical perspective DBWorld does not provide an API to allow programmatically access and interrogation of the service. With respect to current mashups approaches this service is useless. Google Calendar, one of the known Google Apps services, is the use case’s calendar. For any calendar the basic required information is the title of an event, the date and description. Opposed to the DbWorld services, Google provides for this service beside the regular web page form also an API to access the contents. However a uniformly way of dealing with all the services is desired. In addition the user should not be required to face technical issues (i.e. APIs).

In this context achieving the goal of having the conferences stored in Google Calendar means a manual approach. Hence the user is required to maintain two open tabs in the browser. Even though there might be several events that comply with a search term the user must deal with them one by one as DBWorld does not provide built-in search functionality. The user has to move between the open tabs several times in order to store only one event in the calendar.

To store a list of conferences in a calendar the user is part of collaboration between these two services and has to interact with the services at least to initiate the search for conferences. Thus the mashup engine supporting such collaborations has to handle both user’s behavior as well as system’s behavior.

4. TOMTOM SYSTEM

The TomTom2 system is probably one of the most known and more evolved [1] Global Positioning Systems (GPS)3. Compared to the Web the GPS resolves only the intelligent navigation problem. Before the technological advance, people where navigating by reading the position of the stars, land based signs and maps. Navigators were required to have knowledge of astrology, geography, topography, cartography to be able to compute their current position, routes and distances. In addition a specific process, reasoning techniques as well as a computational model are required to compute position, distances and routes. There is a set of concepts that has to be known and understood by the human user in order to be able to resolve the issue of navigation. Concepts such as location, map, route, obstacles on the route are indispensable entities.

However with the GPS appearance the navigation is not a problem anymore. Nowadays most of the cars have as a standard feature an intelligent GPS device. First of all these intelligent devices are highly user oriented and they have been designed in such way that most of the users are able to use them in a very simple manner. Users are not required anymore to compute position; maps which nowadays are digital are provided today also by the GPS unit. These maps are context based, hence according to the position the maps can display a higher or a lower degree of details (e.g. outside of a town, or inside of a city where details are most welcomed). Graphic representations are easy to understand and the user is actually assisted in arriving to the destination based on a route.

Figure 1 depicts the complete system required to get a person for example from Paris to Berlin. The problem itself of getting to Berlin is known only to the user. The available options are: either define into the GPS the route to follow or define only the starting and the end points and the GPS unit proposes a route. The route proposal is conditioned by the existence of necessary knowledge - here maps and reasoning and computational algorithms - on the GPS unit. Position is computed based on the continuous stream of events that the GPS unit is receiving from the available satellites. Both the user as well as the GPS unit follows the route. Current position is always continuously communicated to the user. One very important as in order to achieve the

\[2\] http://www.tomtom.com/

\[3\] http://en.wikipedia.org/wiki/GPS
goal of arriving to Berlin, is that the user must drive the car. Therefore the user itself is part of the system, and the system can not resolve the problem by its own unless the user gets to be an active component of the system.

5. PROPOSED ARCHITECTURE

Mashups are still an emerging BI practice [8] as users are only aware of the limited contexts (i.e. Google Mahsups) where they have been deployed initially. BI mashups are required to deal with rich contextual information, semantic data [8]. The architecture introduced in this section complies with the requirements discussed in Section 1: software and business intelligence on demand, easy consumption, instant use but powerful. The most important aspect of this architecture is that the user is seen as a component part of the system.

5.1 The Conferences’ Calendar and TomTom Side by Side

TomTom it is an interactive system, and in order for an user to achieve a desired goal (e.g get from Paris to Berlin), both the user and the TomTom are required as components in the system that will fulfill user’s goal. Only a car equipped with a GPS will not get to Berlin by itself unless a driver drives it. The web browser provides the same generative and interactive environment that requires users to take active part. Web pages have meaning for the human users, and unless AJAX is involved nothing will change in a web page unless a human user interacts with that web page.

Navigation context requires a route and a map to get from A to B. When it comes to BI on the web the route and map are represented by the mashup. While in the TomTom case the user deals with a restricted context, but when it comes to the web users must face actually an infinite number of contexts. Different contexts are put together in order to achieve a desired goal. The conferences’ calendar use case’s (discussed in Section 3) contexts comprise concepts such as: subject, deadline, web page, calendar event, date, description. Albeit the web offers such a large landscape of contextual information, the way users interact with the web never changes. This is another resemblance with the TomTom devices. As stated in Section 4 the required reasoning and computational mechanism required to calculate position are predefined in the GPS device. Similarly a mashup engine for the browsers requires such an approach. But the reasoning and computational mechanism for the engine must cope with the high degree of generality imposed by the web.

The need for TomTom like functionality and characteristics has been identified also in other fields, i.e. business process management [15]. A mashup behaves like a web page and so it can be further combined to create other mashups. A proxy server to mediate interaction between services that reside on arbitrary domains is not required anymore as the browser itself works as a proxy server.

Basically there are several layers involved when dealing with a web page: JavaScript / AJAX, DOM, CSS, and a representation layer (HTML, Atom, RSS, XML). The mashup engine has access to all these layers.

Figure 3 portrays the mashup engine itself. Basically it comprises a set of sensors, a set of actuators and a Central Processing Unit (CPU). Similarly to TomTom any interaction, conversation, collaboration between user, services, browser is done via events and actions. Sensors perceive events and actuators perform actions. DOMSensor deals with Document Object Model (DOM) events. Consequently DOMActuator performs DOM related actions.

The CPU comprises (see Figure 4) two perceptual processors one that deals with external perceptions and the other one that deals with the internal perceptions. The memory has two subcomponents: WorkingMemory and LongTermMemory. LongTermMemory is the one that stores knowledge over time. WorkingMemory on the other hand deals with the on-the-fly processing instructions. WorkingMemory mediates interaction between components. The CPU architecture has influences from the Model Human Processor (MHP) block diagram introduced in [10]. The Cognitive Processor performs all the reasoning and computational processes in a unified way based on the available contextual information. As stated in [10] the mind is the
control unit that guides the behaving organism in its complex interactions with the dynamic real world. Both the behaving organism as well as the environment behave through time with a series of interactions between them.

The entire framework is illustrated in Figure 5. Three major components are involved: the mashup editor, the mashup engine and the repository. The framework does not constrict the writing environment to a particular language or to a particular tool. However the editor needs to implement a Uniform Communication Mechanism. Hence an option could be the Oryx editor which is business process oriented. Nevertheless since the engine works at the concept level, another editor could address also users with a background in psychology. Different types of repositories can be used to store the mashups. A specific manager mediates the interaction.

An initial use case (see Section 3) that uses the approach introduced here will be available at http://www.calendconf.eu.

6. CONCLUSIONS

This paper introduced a novel architecture for browser based mashups. There are several new things that this architecture brings into the light: (1) it is tailored for fully fledged browser based mashups; (2) the user is an active part of the system; (3) it utilizes a cognitive context aware engine; (4) tackles software and business intelligence on demand, instant use but powerful.

7. REFERENCES


http://www.oryx-project.org


