A unique interface for web and telecom services: From feeds aggregator to services aggregator

Nassim Laga[1, 2], Emmanuel Bertin[1], and Noel Crespi[2],

[1] *{{nassim.laga, emmanuel.bertin}@orange-ftgroup.com}*
[2] {{*noel.crespi@it-sudparis.eu}*

[1]*Orange Labs Orange Labs - France Telecom R&D, 42, rue des Coutures, 14000 Caen France,*
[2]*Institut TELECOM SudParis, Mobile Networks and Multimedia Services Department, 9 Rue Charles Fourier, 91011, Evry Cedex, France,*

Abstract

NGN promise more innovative services that will blend telecom services and web information. However, there is no real implementation of such converged services today. In this paper, we highlight the benefits of adopting web 2.0 technologies for telecom services. As the services are today mainly driven by the user's needs, we proposed the concept of unique customizable service interface. To prove this idea, we have specified, built and deployed a dashboard that blends web information and telecom services at the presentation layer, on a single web page.

Introduction

With the standardisation of the IMS architecture [1][2][3], the service development methods of telecom operators become more and more similar to the web development methods. Indeed, service development environment of telecom operators is based on reusability of the basic enablers (e.g. presence, messaging) which is very similar to a service oriented architecture (SOA) approach [4] that has proved its usefulness in the decade. Moreover, several operators have even opened their network through OSA/ParlayX [5] web services to facilitate the development of new telecom services. However, the promised innovative services take a long time to appear because of difficulties to manage the real time applications in the web environment.

The web community has otherwise gained in experience of real time applications (e.g. googleTalk, and webMessenger) that uses web 2.0 [6] technologies such as AJAX [7][8]. Moreover, innovative applications have appeared on the public web, such as web aggregators, mashups, and social networking. These applications are characterized by the aggregation of services, sharing, participation and personalization. Web aggregators (like Netvibes [9] and iGoogle [10]) are applications that give access to many web feeds (data format used to provide users with dynamic content such as news and weather) of many providers from a single web page; these feeds are displayed as independent blocs in the page (e.g. weather feeds, sport feeds, political news feeds…).

The goal of this paper is to show how telecom operators can benefit from web technologies to provide a unified point of access to user's services (web information and telecom services). We consider the unified interface concept as a first step toward those innovative services. Therefore, we have developed a customizable dashboard that blends both telecom real-time services and web information services on a single web page. This enables the user to access, to monitor, and to use all its preferred services simultaneously. Our solution is implemented at the presentation layer. This is a definitely a new approach for the convergence of the telecom domain and the web domain. The real-time issues are handled using web 2.0 [6] technologies.

The rest of this paper is organized as follow. We review the related work in section (1). Section (2) illustrates through motivating and concrete examples the added value of services aggregators for both the operator and the end user. Section (3) details our contributions and the functional description of the proposed framework. A high level overview of the architectural design is described in section (4). We go over the implementation issues and chosen solutions in section (5) and then conclude in section (6) with future directions of our research work.

1 Related work

Customizable web feed aggregators such as Netvibes [9] and iGoogle [10] provides the user with (1) the ability to access many feeds from a single web page,

(2) the ability to integrate third party feeds, and (3) personalization capabilities.

1.1 The ability to access many feeds from a single web page

This is definitely the main characteristic of web aggregators. Each module, named widget or portlet [11], is defined with an URI that refers to its presentation layer. The aggregator requests the URIs of all modules chosen by a given user, and then displays the widgets as independent services in the same web page. Each interaction between the presentation layer of a module and its business logic is handled using AJAX technology. AJAX technologies enable the framework to update a module according to its business logic, hosted on the server, without refreshing the whole page. This mechanism enables the framework to keep modules independent each from others.

1.2 The integration of third party feeds

AJAX technologies enable aggregators to integrate modules of different providers. Service providers and independent developers can then develop their own widgets. There are many ways to develop a widget. Actually, this depends on the target aggregator (whether this widget will be incorporated in iGoogle, Netvibes, or another aggregator). However, the universal widget API UWA [12] has gained a large community (Netvibes, iGoogle, Mac, Windows vista, Yahoo! Widgets, and Opera). Moreover, W3C has initiated a standardization effort of widgets development API [13].

1.3 Personalization capabilities

Aggregators are powerful tools that promote the personalization of user feeds. Indeed, each user can personalize its aggregator by organizing his modules into groups by adding, moving, and deleting widgets tabs. Moreover, he can even add, move, and delete modules.

Figure 1 is a Netvibes screenshot that shows the listed characteristics of feed aggregators.

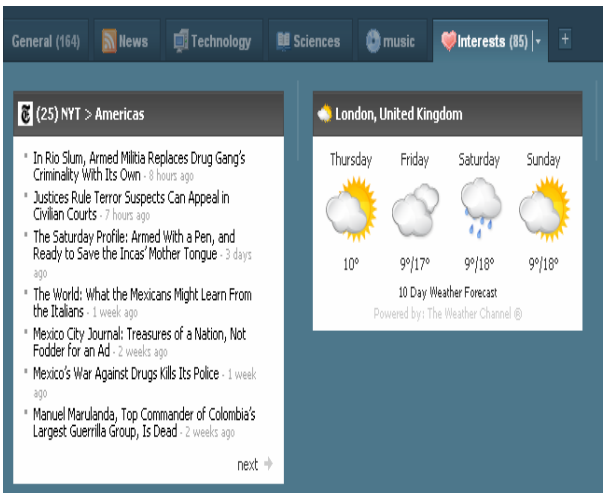

Figure 1: Netvibes screenshot

However, existing feed aggregators are limited to information services (e.g. task management, notepad, map or RSS feeds) and they do not provide telecom services such as telephony or web-conferencing. Indeed, one of the difficulties is how to send real time events from a server to the web page and especially to the presentation layer of the corresponding module.

In the next section, through examples, we highlight both end user benefits and operator benefits of aggregating services in a single web page. We underline the concept of "service aggregation" instead of "feed aggregation" as our main contribution is the usage of the dashboard concept, not only to aggregate feeds but also to aggregate services such as email, phoning, IM, and all enterprise applications.

2 Services aggregators added value

Aggregating services into a single web page provides many advantages to both end users and operators.

2.1 End users advantages

- **Rapid interaction between independent services:** Indeed, as users can access instantly to theirs services, they can perform communication between two or many services using simple and usual actions such as "copy and paste". One motivating example amongst others is when the user receives a call; he can copy the phone number of the caller and paste it on the directory service, search for the caller information such as name, his function title, work address and personal address, and then decide whether to respond or not. Moreover, suppose now that the interlocutors decide to have a meeting at the caller office, the user can copy the office address of the caller and paste it on the "*Map*" to check how to reach it.

- **Third party services integration:** The user is not limited to the services of a single operator; he can also use services of third-party providers. Indeed, in the example above, the "*Map*" service can be provided by a company A and the phone service by a company B. Moreover, advanced users can create their own services and integrate them into their personal web page; they can even share the developed module within a user community.

- **Single authentication:** As users access to their services from a single web page, the operator can perform a single authentication for all provided services. This will avoid users to re-authenticate for each service. (E.g. if the phone service and the directory service are

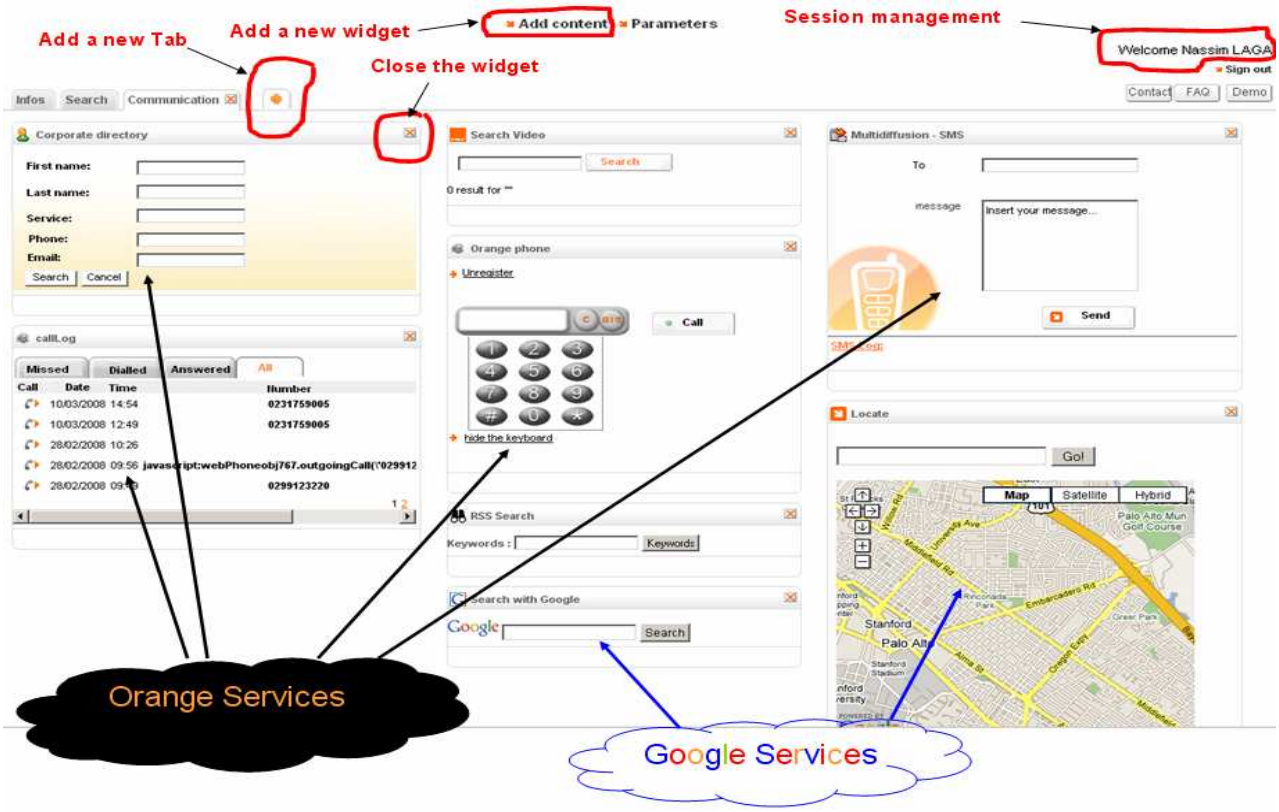

Figure 2: Web and telecom unified dashboard

provided by the same operator, the user then performs a single authentication for both services).

- **End user personalization:** This is of course an important added value of aggregators of services. Users can personalize theirs web page. They can organize their services into groups through tabs. They can also add, move, and delete services at the run time. All configurations (tabs configuration and modules configurations) are saved and retrieved at each disconnection and connection.

- **Service centralization:** Instead of managing many applications and the corresponding URIs, the user aggregates them into a single web page (e.g. Map, Phone, Email…). For example, users can organize their web page into a search-engines tab, emails tab, telephony tab, and RSS feeds tab. The search-engines tab contains services that performs search on the web such as the Google search engine and the Yahoo search engine; therefore, users should no longer launch a new instance of their web browser and open the needed web site for each specific need, instead they aggregate theirs preferred search engines into a single web page.

2.2 Operator advantages

- **Better know its users:** Telecom operators will be aware of the modules chosen by a given user. This enables them to detect the main interests of this given user and to use this knowledge for advertising purpose.

- **Reusability:** Making a widget as simple as possible is the slogan of the widgets developers; indeed a widget is supposed to perform a basic function, analogous to IMS enablers [16]. This extends the reusability principles to a wider domain and opens a new research area that consists in the composition of web services with IMS enablers at the presentation layer. Indeed, telecom operators can develop widgets that expose the basics enablers such as presence and IM. Third party developers can then reuse and combine these widgets with other applications using simple Ajax requests and JavaScript.

- **Single authentication:** Single accessing interface enables the operator to manage a single authentication for all their services. This facilitates telecom operators in the management of theirs services.

3 Contributions and functional description

We have developed a unified dashboard for Internet and telecom services. This dashboard consists in a web page that aggregates (as described above) both web services and telecom services. Web services consist for example in web search, map, and RSS feeds. And telecom services include for instance telephony, messaging services, and videoconferencing. Our prototype is illustrated in figure 2 and it is available on [14] and [15].

Users can access to their services through the dashboard, by using an Internet connection and a web browser whatever the used device (mobile phone, PDA, laptop, and desktop PC). Moreover, the dashboard enables the user to download services of different providers. This dashboard is therefore a strong driver for service convergence.

As illustrated on figure 2, users can dynamically add, move or delete widgets, and organize the dashboard into tabs. Then, instead of using many different applications, the user aggregates its preferred services into a single unified interface. The user retrieves its dashboard configuration after his authentication, as the framework saves the user preferences at each session. These mechanisms provide the users with the ability to personalize their own page.
Our solution does not impose any API to service developers. The only condition to integrate a widget on the aggregator is that the service must render a well-formed HTML document (XHTML). This provides more flexibility on service development while facilitating page parsing which is necessary in widget incorporation process. However, to interact with the dashboard in order for example to set up a configuration form, or to react on module events (onload, onrefresh, and onclose) the developer should use a specific API.

4 Architectural description

In this section we give a high level architecture description of the proposed framework. Figure 3 displays a component overview of the framework.

Components of our framework are categorized into two parts: the server side component and the client side component. The server side components manage the persistent data such as user credentials, user preferences, the catalogue of services, and the configuration parameters of the services.
The task of the client side components consists essentially in the web page organization according to the user preferences, downloading the modules and modifying their HTML content to make it compatible with the aggregator (more details on these modifications are reported in the implementation issues section).

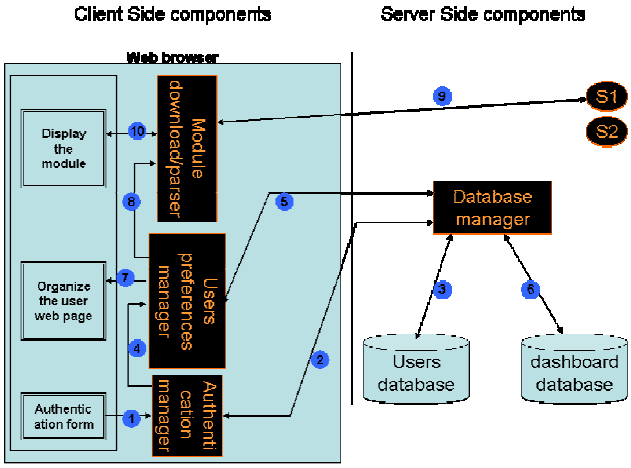

Figure 3: Internet and telecom unified dashboard Architecture

First of all, when the user fills the authentication form, a request is sent to the server side that checks in the database whether the user is recognized or not. If the user is well authenticated, the "authentication manager" component launches the "user preferences manager" component that loads the user web page according to the user preferences (number of tabs, name of tabs, the chosen modules, the positions of the modules, the modules of each tab, and the configuration parameters of each module). The "user preferences manager" can then display the tabs and launch the "module download/parser" component with the configuration parameters. This component downloads the modules and parses them in order to integrate them into the web page. The most important action performed by the parser is the modification of all widget links into AJAX requests in order to keep module browsing inside the module and not to change the whole page.

5 Implementation issues

Our implementation of the dashboard is based on web technologies such as JavaScript, AJAX and PHP. We have indeed implemented a comprehensive framework for services aggregation. As presented above, this framework consists in a client side (executed on a web browser) and in a server side (executed on a web server).

The client side of the framework is implemented with JavaScript. AJAX is the basis of our framework as it enables us to request a server side of a module and modifying its presentation layer while keeping the browser in the same page. Moreover, this mechanism keeps the modules independent each from others as actions on module A will not modify a module B.

However, for the sake of security, today most used web browser (IE6, IE7, and Firefox) don't allow making AJAX requests between two different domains. However, this security measure presents a handicap in our context. Indeed, we need to download third party services. To circumvent this constraint, we use a proxy; implemented in the same domain as the framework using PHP, the proxy enable the client side part to request third party servers. Indeed, the "module download/parser" component does no longer make the requests directly to the services but makes them to the proxy with the URL of the needed module as a parameter. The proxy then downloads the service and renders it to the client side part. The browser does no longer blocks the AJAX requests as the proxy and the framework are in the same domain. However, for the sake of security, we need to prevent all modules to make AJAX requests outside their domains. This control can be done in "module download/parser" component.

Figure 4 summarize the usefulness of the proxy.

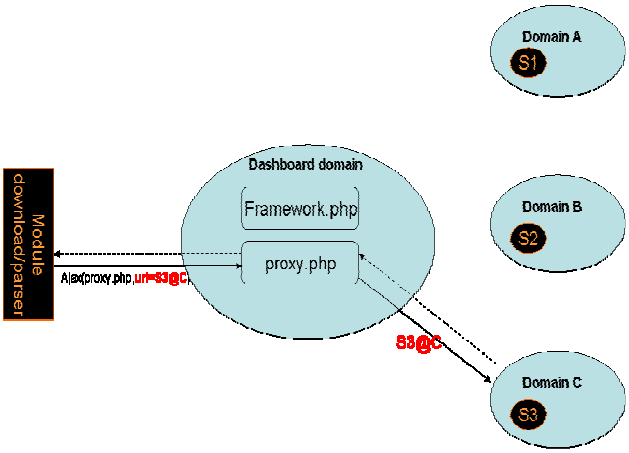

Figure 4: circumventing security constraint of web browsers

The server side is implemented with PHP**.** The main parts are:

- As mentioned above, a proxy through which we load all requested services.
- A component that manages users' database.
- A component that manages the users' preferences database.

The "module download/parser" component receives the XHTML response of the requested service through the proxy and makes the necessary modifications on the module for the sake of keeping modules independent each from others on the web page. The necessary modifications consist essentially in the modification of all relative URLs (relative to the page domain) to absolute URLs (include the whole path). These URLs are then modified so that the new one will target the proxy with the real URL as a parameter (e.g. *http://example.com* will be transformed to *http://frameworkdomain/proxy.php?url=http://example.com*). To keep browsing always inside a module, we need to change all links and URLs to AJAX requests. such modification avoid reloading the whole page when a user click on a module links. Further, we need to search about the API key words inside the module and replace them with the necessary functions (e.g. if the developer of the module need to handle the onclose event, he should use the aggregator API; he should add the following statement: ON_UNLOAD = handler).

Another implementation issue is the trade-off between simplicity of service development and scalability of the framework. Indeed, defining a specific API for service development (as this is the case for Google or Netvibes, e.g. with the Universal Widget API UWA) facilitates service integration in the dashboard but is a constraint for third party developers who need to master that API. On the other hand, not defining an API raises security and scalability issues. To address this problem, we impose to developers to render a well formed XHTML document. This simple condition is a trade-off between flexibility in the development and scalability of the integration. Indeed, while the condition of rendering an XHTML document is simple, it provides the dashboard the ability to load and parse the document efficiently (parsing an XML file is much faster than parsing an ordinary file using regular expressions).

6 Conclusion

Aggregating different services in a single web page is a typical web2.0 way to access and to use telecom services. The user is able to compose his own dashboard of services from those that he subscribed to, instead of using many different applications. Users can, in this way, access and use all his telecom and web services (phone, calendar, visit Card…) through a single web page.

Experimentation has been open since November 2007 for the Orange Labs staff. And the first user feed back was very positive. Indeed, In January 2008, after two month of experimentation, 74% of users use the dashboard as a start page to aggregate mass-market services as well as corporate services.

As many operators plan to open their infrastructure through reusable components, we consider the dashboard concept as a first step toward communication services integration; it brings new research fields like composition at the presentation layer and user participation in the creation of new services. Our future research work will investigate this thematic in more details. We will propose mechanisms to enhance the usability of this unified service dashboard, for instance with the drag&drop

of information between various service widgets. We will also study how the user can participate in the creation of new services.

References

[1] 3GPP: http://www.3gpp.org/ftp/Specs/html-info/23228.htm
[2] 3GPP: http://www.3gpp.org/ftp/Specs/html-info/22228.htm
[3] Miikka Poikselka, Aki Niemi, Hisham Khartabil, Georg Mayer, "The IMS: IP Multimedia Concepts and Services," ISBN: 0470019069.
[4] E. Thomas, Service-Oriented Architecture: A Field Guide to Integrating XML and Web Services. Prentice Hall, Upper Saddle River, NJ, 2004.
[5] http://www.parlay.org/en/specifications/
[6] O'Reilly T. "What is Web 2.0. Design patterns and business models for the next generation of software." O'Reilly Media, 2005.
[7] Linda Dailey Paulson, "Building Rich Web Applications with Ajax," Computer, vol. 38, no. 10, pp. 14-17, Oct., 2005
[8] Zepeda, J.S. Chapa, S.V. "From Desktop Applications Towards Ajax Web Applications," Electrical and Electronics Engineering, pp 193-196, 5-7 Sept. 2007
[9] http://www.netvibes.com
[10] http://www.google.com/ig
[11] C Kaar, "An introduction to Widgets with particular emphasis on Mobile Widgets," Computing, Oct. 2007.
[12] http://dev.netvibes.com/doc/uwa_specification
[13] W3C http://dev.w3.org/2006/waf/widgets-reqs/
[14] http://www.bubbletop.com/
[15] http://www.espace-utilisateur.orange-business.com
[16] http://parlay.org/en/specifications/pxws.asp