EzWeb/FAST: Reporting on a Successful Mashup-based Solution for Developing and Deploying Composite Applications in the Upcoming Web of Services

David Lizcano
School of Computing
Universidad Politécnica de Madrid
Campus de Montegancedo s/n
28660 Madrid (Spain)
(+34) 913367396
dlizcano@fi.upm.es

Javier Soriano
School of Computing
Universidad Politécnica de Madrid
Campus de Montegancedo s/n
28660 Madrid (Spain)
(+34) 913367394
jsoriano@fi.upm.es

Marcos Reyes
Telefónica Investigación y Desarrollo
Emilio Vargas 6, 28043 Madrid (Spain)
(+34) 913374000
mru@tid.es

Juan J. Hierro
Telefónica Investigación y Desarrollo
Emilio Vargas 6, 28043 Madrid (Spain)
(+34) 913374000
jhierro@tid.es

ABSTRACT

Service oriented architectures (SOAs) based on Web Services have attracted a great interest and IT investments during the last years, principally in the context of business-to-business integration within corporate intranets. However, they are nowadays evolving to break through enterprise boundaries, in a revolutionary attempt to make the approach pervasive, leading to what we call *a user-centric SOA*, i.e. a SOA conceived as a Web of Services made up of compositional resources that empowers end-users to ubiquitously exploit these resources by collaboratively remixing them. In this paper we explore the architectural basis, technologies, frameworks and tools considered necessary to face this novel vision of SOA. We also present the rationale behind EzWeb/FAST: an undergoing EU funded project whose first outcomes could serve as a preliminary proof of concept.

Categories and Subject Descriptors

C.2.4. [**Distributed Systems**]: Distributed applications; D.2.11 [**Software Architectures**]: patterns; H.3.3. [**Information Search and Retrieval**]: Retrieval models, Search process, Information filtering, Relevance feedback; H.5.2. [**User Interfaces**]: GUI and Screen design

General Terms

Management, Design, Standardization, Experimentation.

Keywords

SOA, Web Services, Mash-up, Web 2.0, user-centric SOA, Web of Services, Composite Applications

1. INTRODUCTION

A service-oriented architecture (SOA) based on semantic Web services has been considered the key IT for achieving a machine-to-machine integration within company boundaries over the last few years [8]. Therefore, traditional SOA has attracted a great deal of interest in the composite application paradigm. Indeed it is the only current technology stack capable of dealing with composite application developments [1]. However, the latest big phenomena like Web 2.0, and its application to enterprises, the so-called Enterprise 2.0, have revealed the current need to offer a user-centered face in IT to improve business productivity and innovation [12]. And this user-centered approach has never been considered before in traditional business-to-business (B2B) SOAs or in the composite application paradigm.

With this new approach, that is, a user-centric SOA, it would be feasible to achieve a real Web of services, i.e. a Web of ubiquitous compositional resources/services that offer uniform access to end users, giving browsers, mobile devices, and server applications alike accessibility to resources (i.e. providing a "multi-channel" and ubiquitous face to end users). This great revolution in user-service interaction could finally enable a user-friendly, semantically-guided and context-aware framework for end users to develop real composite applications on their own, making back-end resources and services very appealing to a wide range of users and to different usage areas.

This new approach is completely incompatible with traditional SOA, which was conceived for the B2B domain instead of for user-centered composite applications. It is not at all easy to enrich real SOAs with this new face for users. This paper elaborates on the methods, tools and heuristics that SOA must embrace to deal with user-centered composite applications, using Enterprise 2.0 principles (and specially enterprise mashups) as a source of inspiration [6].

With this in mind, we elaborate in this paper on the synergies between the Enterprise 2.0 and the WS-SOA concepts with regard to the development of user-centered composite applications. Enterprise 2.0's focus on the principles of including human beings and multi-device and mobile ubiquitous adaptation, etc., and the exploitation of users' collective intelligence should be considered a key enrichment of existing composite applications. It is therefore expected to act as an enabler of an improved user-service interaction. Our approach is being supported by two hot research projects: FAST and EzWeb. These projects are referenced and examined at length in this paper. The rationale behind FAST, i.e. a complex gadget development environment, and EzWeb, i.e. a reference architecture and implementation of an open Enterprise 2.0 Mashup Platform, are both presented and exploited in a use case as a proof of concept. These two elements together empower users to co-produce and share *instant composite applications and their components*. The remainder of the paper is structured as follows. First we revisit the notion of traditional WS-SOA-based composite applications, and analyze their major shortcomings with regard to the ideal user-service interaction in a user-centered Web of services (Section 2). Also, we elaborate on a use case that illustrates the current user-service interaction needs in composite application development, where current IT like traditional Web services or novel mashup ideas based on disparate and independent gadgets, have more than once been found to be wanting. The above shortcomings of current approaches and technologies, and the ideal solutions for these problems can be easily identified in this use case (Section 3). We then go on to illustrate the guiding principles to achieve our aim in the Section 4 and some general key ideas to materialize this principles in Section 5. We then present a novel architecture framework, built on the FAST development approach and the EzWeb exploitation platform. FAST creates the building blocks and EzWeb interconnects these building blocks to compose instant applications (Section 6). Section 7 describes existing simple prototypes of this framework as a proof of concept, dealing with the use case presented in Section 3. Section 8 presents other related work. Finally, the last section concludes this paper and presents a brief outline of future work.

2. WS-SOA-BASED COMPOSITE APPLICATIONS AND THEIR SHORTCOMINGS ON THE ROAD TOWARDS A POWERFUL USER-SERVICES INTERACTION IN A WEB OF SERVICES

The massive deployment of user-centric services over the Internet demands services that must be accessible for all users (not only enterprise stakeholders). Therefore, services should flexibly and dynamically support common daily processes (both business processes carried out by companies and processes conducted by individuals or groups in their daily life) at any time [17]. Users will see the tools supporting their daily work replaced by composite applications based on Web Services, but traditional Web Services are not well enough tailored to users and their daily processes. Obviously, SOA, as it was originally conceived, represents an architecture focused fundamentally on a B2B context. It is weak for B2C problems, since it does not offer the best prospects for dealing with user-service interaction [9]. We can tackle its shortcomings from three different perspectives:

1) ***SOA's aim:*** SOAs merely aim at facilitating seamless machine-to-machine collaboration. SOA deployments are very abstract and invisible to users. Its customers of choice are medium-sized or larger corporations instead of normal end users along the long tail of Internet. Therefore, with SOA, normal Internet users with little IT expertise have not been able to easily retrieve and use services because services mostly reside within company boundaries and are only accessed for professional use in a corporate context.

2) ***SOA's technology***: Apart from SOA's aims, this architecture relies on a set of complex standards that are not user friendly [1]. Because, technically speaking, SOA is extremely complex, there needs to be one or more expert players within the value chain to build and provide solutions for their customers. In contrast to this one-to-many value chain model of numerous SOA use cases (where one expert serves many clients), new value chains should begin to be mostly loosely coupled (many-to-many) networks of self-managed self-sufficient users who can offer and consume resources Web.

3) ***SOA's government***: Finally, SOAs are subject to clearly defined regulatory frameworks since they mostly exist in the corporate context. The design, provision, maintenance, and coupling of services must be compliant with legal frameworks. Therefore, they do not allow for the flexibility that the described new user-services interaction model appears to need.

3. USE CASE: A REAL PROBLEM OF USER-SERVICE INTERACTION IN PEOPLE'S DAILY LIFE

Bill is a production manager in a footwear firm. As part of his job, Bill makes frequent business trips to attend shoe industry fairs and to visit his company clients and suppliers. Partners are scattered around the globe. Bill's next trip takes him to China where he will give a talk about his company and deal a raw material purchasing contract.

Planning such a business trip usually took him long time browsing several Web portals and searching for accommodation and transportation, mainly including flights. During the trip, he also has to take with him lots of notes, maps and documents that guided him and where he has recommendations, addresses and his agenda.

Bill and his workmates have created and evolved a business trip supporting mash-up. They have been able to manage this task without programming skills. This mash-up is a Web workspace where they have integrated the following elements:

- A trip search window that operates on the travel companies his firm works with. Bill chooses the suggested train trips, flights, busses and so on, after introducing his requirements such as dates and destination. This element allows Bill to purchase the boarding cards to China without worrying about doing the payment by himself since it automatically invoices the trip to Bill's company.

- A hotel finder element is also included in this workspace. This window offers accommodation possibilities in destination and dates that Bill has specified. These recommendations follow his preferences, such as price or desired hotel facilities.

- Other window deals with searching regional transport at destination. It offers information about busses, trains, trams, taxies, etc.

- The most important element in this mash-up is an agenda where the chosen trip elements, such as hotels, flights, trains, etc, are graphically shown all together. This information appears as Bill configures his trip in the other windows. This window integrates also Bill's personal agenda information, including his arrangements and meeting during and outside the trip. This tool helps Bill to match all the trip elements and smartly plan it.

- Finally, the mash-up includes a map where all trip's locations and journeys are graphically shown. This map supports Bill during his trip, helping and notifying him about footways, bus stops, hotels, restaurants and even tourist information.

Using this mash-up, which Bill received from a workmate, he can manage and configure his whole trip in an easy and integrated manner, including support when he is on the road. Obviously, Bill's preferences about trip specifics are not the same as his workmate's, i.e. flight menu, seat in the flight, price ranges, etc. However, the workspace is its elements are adapted to fit this new context (new user), and allows him to personalize the work bench in a guided, flexible and easy manner from his desktop PC.

Once this setup has been done, Bill starts planning his trip. First, he introduces the meetings and arrangements dates in his personal agenda. In the workspace, the trip search window shows flights around these dates, and the hotel selector shows available hotels in the destination city matching with the options already chosen. However, a tiny news aggregator based on the destination city, announces that there is a great sportive event on that city during his visit. That news explains the expensive prices for the accommodation suggested. Then Bill decides to search for accommodation in some other city using the map, and he finds more affordable hotels in a nearby city. Before booking the hotel he looks for transportation between both cities in the regional transport window, getting a high speed train that, according to the agenda's timing representation, allows him to reach his meetings on time. Nevertheless, he notices that the train fares are only given in yens, Chinese currency. Bill realizes that a currency converter would be very useful during the trip. This issue is new to their workspace since his workmates have only travelled inside the EU. Using a simple drag and drop action, Bill adds such a converter to the mash-up, which translates to Euros any other windows price whatever its currency.

To his surprise, the support given by this workspace goes far beyond the single arrangement of his trip. This mash-up helps him showing itineraries using his own PDA, checking timetables or even ask for a taxi or rent a suit if there is an unforeseen event during his trip. Moreover a new element appears in the application. It is a restaurant recommender that gathers opinions and ratings from Spanish travelers.

During the visit the PDA is continuously guiding him with the map about his real situation, when he must reach the places, which means of transport to use and its price. One day in the visit it happens that the train that takes him from one city to another is quarter an hour delayed. The mash-up alerts Bill since he will not arrive on time to his meeting, and suggest him to take a taxi at the final train station, as Bill used to go by walk. When the train finally arrives, the delay is even longer and the recommendation changes because the underground has just started its service. Bill is also indicated the path to the nearest underground station.

This ubiquity approach helps Bill to easily reach the company he is visiting, and accessing through his PDA companies internal services. He therefore asks for a meeting room, accesses the internal network and borrows a projector in his visit. During the meeting Bill gives his talk and automatically receives digital updated information about the attendees, and sends his own business information back. This fosters the exploitation of his social network and makes the most of the possibilities of his company, allowing it to thrive day by day.

4. DESIGN PRINCIPLES ENABLING THE WEB OF SERVICES

The evolution of Web 2.0 sites and applications is a testimony to the progress achieved to improve user relevance and service usability. However, current service front-ends are far from meeting end-user expectations [18]. Applications are still based on monolithic, inflexible, and unfriendly UIs. This is a serious obstacle for achieving the benefits of the Web of Services. In order to build the next generation service front-end for this ecosystem of services we propose three guiding principles. These principles are further detailed in the following.

4.1 Enabling users to design and share their operating workspace and applications

Users should be able to design and implement their own interfaces in a flexible and friendly manner. New generation front-ends should provide new tools aiding end-user UI creation and self-adaptation, while supporting a dynamic computing infrastructure [19]. Community-based collaboration tools should also be supported to satisfy the demands for secure social interaction and improve knowledge and resource sharing. More to the point, the following aspects should be covered:

1. Empower users to select resources of their interest, annotate, and configure their own personalized operating environment
2. Promote user pro-activeness for creating new resources, to improve versions of resources, and share further expertise about these resources, their use, and their interrelationships
3. Provide social facilities to share services, results, knowledge and resources with other users
4. Foster social interaction by providing visual mechanisms to manage user communities, user identity, privacy and security constraints, and the information to be shared, annotated, or send to specific groups and individuals

The new generation of user-service front-ends should be based on helping users with the flexible composition of applications. Application and service front-ends will no longer be conceived as monolithic blocks, but as a set of interoperable service front-end components –called gadgets–, which are available from a catalogue. A catalogue of components or gadgets will be available for creating application or service front-ends by using and combining these building blocks to construct new components. Users should be able to create their own applications by combining different catalogue components without any help from IT experts or a thorough knowledge of the underlying infrastructure.

4.2 Businesses need to adapt to the new reality

Today’s competitiveness-driven business markets and the severe time-to-market restrictions on applications, specifically for enterprise IT systems, have increased the business needs to evolve applications to suit this new reality. They can be evolved through the following key guidelines:

1. Businesses need to embrace the Software as a Service (SaaS) model as an effective software-delivery mechanism [5]. This approach helps to reach a marketplace of services that can be

composed to create unanticipated business solutions adapted to real needs.

2. Next generation Web Services ecosystems must respond to unforeseen business requirements that emerge or evolve spontaneously. This should be supported by new software development methodologies that ease the integration, adaptation and evolution of service-based applications.

3. Company boundaries must be eroded, evolving towards the Internet of Services vision. This approach can be split into two perspectives:

a) Next generation business systems should adopt a user-centric approach to take into account users. Users of these services are no longer just the company's employees; clients should also access the same business resources. This could even affect the entrepreneurial service-based workflows [2].

b) Collaboration between companies, irrespective of their size, must be fostered, thus increasing productivity and accelerating innovation [19]. The creation of collaborative services by integrating components from disaggregated companies will afford new business opportunities and improves the global service provided to end-users.

4.3 Context-adapted user-service interaction

The proliferation of multiple Internet-enabled devices allows end users to access the Web anytime and anywhere. As a result, there is a wide range of situations in which a user might need to access these services, and they must therefore be provided the right user interface for the right situation. These situations can be defined as the context in which access occurs.

Next generation service front-ends should take into account the following context aspects:

1. The delivery context, that is, a set of attributes that characterizes the environment in which a service is going to be delivered. Delivery context is a crucial aspect with regard to service front-ends, as it provides a clear indication of what the capabilities of the target device, web browser and network are. Such aspects play an important role in the end user experience. The information about these capabilities should be exploited by service front-ends to provide a harmonized experience adapted to the peculiarities of every delivery context

2. Users and their circumstances: This aspect includes properties such as user identity, profile and roles.

3. The surrounding environment, including the spatial location, speed, light, level of noise, nearby objects/things, etc.

4. The situation / time which has to do with variables such as date and time, weather, season, at home or at work, ...

5. MATERIALIZING THE GUIDING PRINCIPLES

In this section we offer a general approach to support the previous guiding principles, recommending key technologies, issues and ideas to materialize them.

5.1 Enabling users to create their applications

The end-user empowerment idea gives an active role to end-users letting them author, enhance, share and customize their own applications and operating environment (workspace), thus going beyond traditional, monolithic user-service interaction. Figure 1 depicts our technical proposal to materialize such principle. The main component of the underlying platform is a resource catalogue containing gadgets (which will implement be the front-end for one or more services) and possibly other application or content delivery services. This resource catalogue plays a similar role than backend service and process registries.

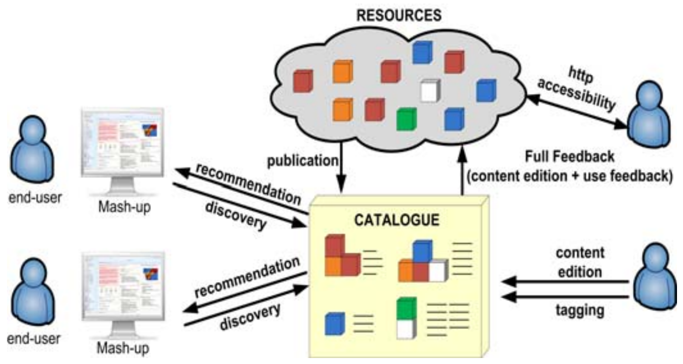


Figure 1. Resource Catalogue.

Our proposed catalogue will be responsible for maintaining the information about each front-end resource (gadget) and associated metadata (template descriptors, deployment information, author, icons, versions, formal-semantics-based annotations, user-assigned tags and punctuation…). In addition the following software modules will be devoted to user-catalogue interaction:

- An upload module that will allow service front-ends creators to add new resources to the catalogue, making them available to the community. To upload a new resource it will be necessary to provide a gadget descriptor (*template*) with all the metadata and deployment information.
- A search and recommendation module that will make it possible to locate and discover those gadgets that satisfy the necessities of end-users at a given moment. This search application will be intelligent, recommending the most suitable resources in the given Context. For example, the set of pieces found in a catalogue should be different when the end-user is at work than when is at home or on vacation.
- A drag-and-drop-based browsing and selection module that will allow end-users to browse and select the resources of their interest, putting them in their operating environment, thus enabling further interactions.

Apart from a resource catalogue and its supporting modules, it is also necessary a runtime environment devoted to the execution of the user's operating environment. Such operating environment (or workspace) will act as a container, enabling the interaction between the end-user and service front-ends. We envisage an operating environment runtime with the following functionalities:

- Flexible layout composition enabling the free disposition of gadgets on the viewport.

- Multiple service front-end views (implemented as tabs, for example) allowing the user to group and compose different gadgets for different problems or situations.
- Publish and subscribe facilities enabling the interconnection between the different gadgets.

5.2 A Universal Framework for Ubiquitous and Context-Aware Service Front-Ends

The seamless context-aware user-service interaction ideas will drive the construction of novel service front-ends capable of using contextual information to influence their behaviour, thus supporting human users in a more effective and personalized way. Particularly, Context-Awareness will provide the following benefits to service front-ends:

- User Interface harmonization for every device or mode of interaction. Contextual information will enable automatic content and application delivery tailored to each target environment.
- User-Service interaction enhancement:
    - Contextual information can be exploited to present a different operating environment depending on each situation. Each operating environment may include different functionalities or interaction modes depending on the situation (at home, at work, on vacation, on the move, on a business trip …).
    - Context can be used by resource catalogue search and retrieval modules to recommend the best gadgets to be used under a given situation, matching rich resource descriptions with the characteristics of the target environment.

To materialize our vision, new tools, formalisms and engineering methods need to be devised in order to simplify the development of context-aware service front-ends:

- A device and modality independent Declarative Authoring Language (and associated standard context-based adaptation policies). This is a fundamental piece for those service front-ends intended to work on multiples devices or modes of interaction.
- A Context Framework composed, at least, by the following elements:
    - A Universal and Extensible Context Model enabling the development of Context Vocabularies.
    - A Context Mediation Layer, including binding formalisms, hiding applications from the complexities of dealing with multiple and heterogeneous Context data sources.
    - A Universal Context API, providing a simple, uniform programming abstraction for the development of context-aware services.

Once this Context Framework is in place an application can infer which actions should be triggered, what data is relevant, and what topics are interesting for the user within a specific context. In the following sections we describe with a finer level of detail the elements that compose our envisaged Context Framework.

5.3 A framework for user's knowledge capture and exploitation in service front-ends

In order to allow end users to create their own workspaces and adapt them to the new business reality appears a new need: exploit users' domain knowledge and collective intelligence for enhancing service front-ends. This idea can be materialized by:

- Encouraging users to create and compose their own service front-ends as a way to solve not anticipated problems that require thorough domain knowledge.
- Stimulating user's participation in the provision of semantic descriptions and annotations for service front-ends. This point is really challenging as it will imply the integration between folksonomies (light semantics) and ontologies (formal semantics).
- Monitor users' behaviour and input to service front-ends.

The accomplishment of the previous ideas requires research on new formalisms and technologies such as:

- Advanced user-centric integrated development environments (User-Centric IDEs) making it possible in a couple of clicks, without deep IT knowledge, to create new and not anticipated service front-ends.
- A catalogue browsing module enhanced with collaborative, user-generated semantics (folksonomies). These folksonomies will be constructed incrementally as users assign tags to gadgets found in the catalogue. A rating mechanism can also be used to promote those service front-ends with a better quality level. This user-generated metadata should be integrated with the formal metadata (coming from an ontology, for example) about service front-ends and will serve to improve the search and recommendation processes.
- Collaborative filtering techniques aimed at analyzing user behaviour in order to:
    - Create and extrapolate implicit user profiles taking into account which links are more frequently clicked or the time spent on each one.
    - Capture underlying user's knowledge to be later used, for example in form filling assistance
    - To extract repetitive interaction patterns driving the automation of new, not anticipated processes

All the novel approaches described above should be the starting point for a new innovation culture based on continuous improvement via reusing and sharing of knowledge. In fact, users will connect and use resources in their workspace according to their skills, situation, social status or others. Wheel-reinventing in organizations or between individuals regarding service front-ends will no longer happen. Competition will rise as there will be an ecosystem of providers and consumers of resources willing to provide the best solution for a problem.

6. COMPOSITE APPLICATION FRAMEWORK, COMBINING FAST AND EZWEB IN AN ENTERPRISE 2.0 COMPOSITIONAL PLATFORM ARCHITECTURE

Now that we have looked at the design principles of the future Web of Services and its general materialization, this section will present the architecture of a specific potential framework based on an enterprise mashup workflow-oriented enabler that empowers its users to co-produce and share *instant applications*, i.e. applications based on composition rather than programming and their building blocks**.** This framework has been built adhering to the design guidelines, technologies, tools and methods of two initiatives, FAST and EzWeb, as an integral solution bridging users and final services.

First of all, we should briefly review the reference architectural stack underpinning the FAST and EzWeb projects, shown in Figure 2.

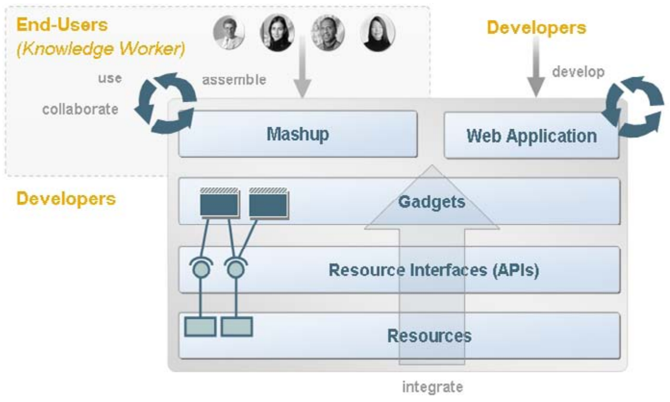


Figure 2. Enterprise 2.0 Mashup Stack.

The bottom layer contains the actual Web **resources**, be they content, data or application functionality. They represent the core building blocks of Enterprise 2.0 Mashups [19] and are the mark of the resource-centric paradigm. According to the lightweight Representational State Transfer (REST) architecture style [4], each Web-based resource can be addressed by a Universal Resource Identifier (URI) giving browsers, mobile devices, and server applications alike accessibility to those resources (i.e. multi-channeling). This programming style makes the resources very appealing to a wide range of developers and for different uses.

The resources themselves are sourced via a well-defined public interface, the so-called **Application Programming Interface (API)**. APIs encapsulate the actual implementation as separate from the specification and allow Web-based resources to be loosely coupled. In this sense, the underlying resources are used as core building blocks to compose individual applications on top of existing resources. According to the REST architectural style, the four CRUD-Operations (Create, Read, Update and Delete) are represented by the HTTP verbs Put, Get, Post and Delete. The Atom Publishing Protocol (APP) represents a first application-level protocol for publishing and editing Web resources following the REST architectural style. The protocol is based on the HTTP transfer of Atom-formatted representations. This is documented in the Atom Syndication XML Format. A new Google-driven initiative, called GData, uses the APP extension mechanism and also provides queries and authentication functionalities. It allows full-text search queries to be sent to the underlying Web-based resource. The returned syndication XML format (Atom or RSS) is based on the Opensearch.org response elements, an Amazon-driven specification. Besides these new lightweight standards, existing application functionalities described with WSDL also represent an enterprise mashup API. A big issue surrounding APIs is identity. Most of the major API vendors have their own authentication APIs. Even though they are all similar, each one is different in the end. The OpenID.Net initiative is looking for a solution to deal with this challenge.

Being resources based and sourced via public APIs, **gadgets** (also known as **widgets**) provide application domain functions or information-specific functions. They are responsible for providing graphics, simple and efficient user interaction mechanisms which put a face to the resources and abstract from the technical description (functional and non-functional) of the Web-based resources. In general, widgets can be both visual (in that they render visual content, such as a chart) or non-visual (in that they provide some form of discrete function or access to a Web-based resource). In contrast to full-blown software applications, widgets represent a tool or component providing a small and specific application domain function. Through configuration and personalization, the underlying Web-based resources can be used according to individual requirements. Therewith, they tend to be designed with a focus on consumption and customization to ensure they are extremely flexible and reusable. But, although the respected W3C published a draft widgets specification, there is no widespread widget model. Software vendors (like Microsoft or Google) defined their own widget model, NetVibes has the compelling Universal Widget Architecture (UWA), and OpenAjax has no component model per se but vital strategies for fitting/putting Web components together in the same mashup.

By assembling and composing a collection of gadgets stored in a catalogue or repository, knowledge workers are able to define the behavior of the actual application according to their individual needs, creating a composite application as a **mashup**. By visually and intuitively aggregating and linking content from different resources, knowledge workers are empowered to create a workspace of their own that best solves their heterogeneous business problems. No skills in programming concepts are required. Many software vendors have started the implementation of so-called mashup makers; visual mashup environments, i.e. IBM QED Wiki, Microsoft Popfly, Serena Mashup Composer, Kapow, JackBe Presto Enterprise Mashup Solution or NexaWeb Enterprise 2.0.

Based on this reference stack, we are considering software development from a **top-down perspective** as opposed to the conventional bottom-up approach. Users will play a leading role in this new approach and the applications will **automatically** adapt to their data and functionality requirements (see Figure 3). This new top-down scheme can be summarized as follows:

1. The end user or consumer identifies a need or series of needs in the form of data to be displayed and functionality to be offered. These users will create their own solutions based on the ideas of mashup and freewheeling wire framing of

complex resources and APIs in a do-it-yourself (DIY) business process. Users will have already composed this complex of resources from REST resources via a piping and wiring composition of simple resources or by remixing/fixing existing resources. REST resources are a front-end to enterprise legacy, traditional web services, data in enterprise boundaries, etc., resulting from a "*RESTify*" process carried out by enterprises themselves.

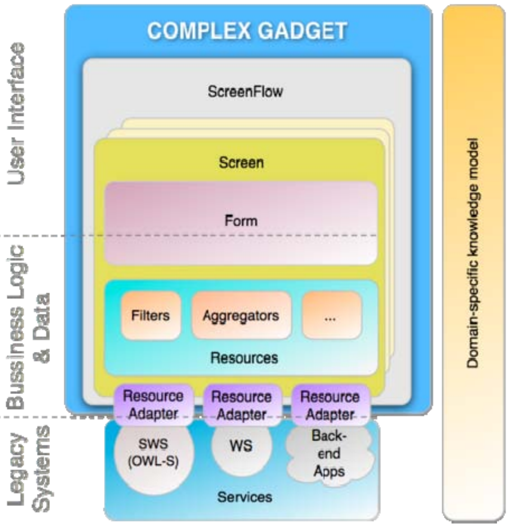


Figure 3. Visual composition of screen-flow resources and interoperability with back-end Web services.

2. Users just have to search the gadget registry to find a gadget (or part of one) that meets their needs or more than one gadget that they can put together to create or compose a new one of their own. The result of this stage is a gadget conceptualization including all the above aspects (user needs satisfied, user capabilities required, interaction models applied, internal logic flow, etc.). This way, a semantic enrichment based on rules, facts and pre/post conditions improves the whole B2C channel of services.

3. Users manage their new gadgets in their own dashboard, supported by an Enterprise Mashup platform (i.e. EzWeb). This platform allows gadgets to intercommunicate with each other and with their own platform, creating a hybrid composite web application. This instant application supports users' daily work thanks to an environment of interconnected resources, offered by a gadget ambience. In addition, users can publish their improvements to the gadget registry for future reuse, adaptation or specialization. Thanks to this, knowledge and innovation management is an implicit part of the process, fostering user collaboration and collective intelligence exploitation.

4. Alternatively, users could contribute clarifications, innovations, bugs, enhancements, comments or simply new usages of their mashup components without actually recomposing, remixing or creating new resources. Increasing the visibility of these business inputs and assuring that they rapidly reach the users of that collective intelligence is vital for boosting business innovation [10]. Therefore, each resource that appears in a mashup should be associated with standard Web 2.0 communication channels (such as blogging, edition of associated entries in the underlying wiki-based catalogue, etc.). This way, users would be able to implement inputs simply and flexibly without having to create/tailor and publish the solution in order to be able to contribute their expertise and share it with the enterprise. This puts existing knowledge to better use.

   Note that, taking into account the concepts and technologies governed by this idea, the contributions will focus on the end user without specific background knowledge in semantics, user interfaces, and back-end integration. However, the user-centric approach supports the rapid development and maintenance of applications and information systems in a clear attempt to reach The Web 2.0 Long Tail.

The central driver of this framework designed to address The Long Tail of user needs is the lightweight resource composition style. In this style building blocks from different contexts are reused to build individual enterprise applications. As illustrated in the Figure 4, the composition takes place both in the resource layer (piping) and in the gadget (wiring) layer according to the enterprise mashup stack (see Figure 2).

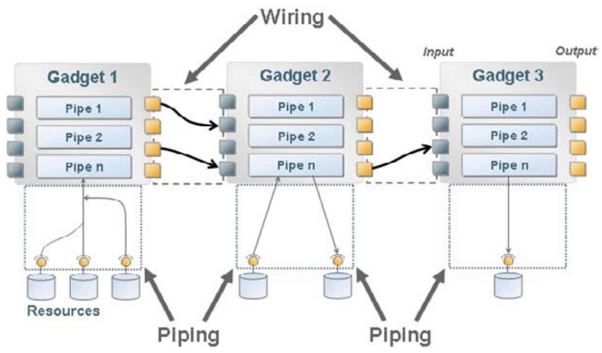


Figure 4. Resource and gadget compositions building composite applications.

In reference to the UNIX shell pipeline concept, the piping composition integrates a number of heterogeneous Web-based resources defining composed processing data chains/graphs concatenating successive resources. The output of each process feeds directly as input to the next one. Aggregation, transformation, filter and sort functions adapt, mix and manipulate the content, data and application functionality of the Web-based resources. Intuitive visual environments for the piping composition represent Yahoo Pipes or IBM Diama. The piping composition itself addresses users versed in classical development or data manipulation languages. In the gadget layer, the actual end user is able to wire existing gadgets together with behavior and data relationships by visually interconnecting their input and output parameters. For example, a form gadget can be placed on a page, allowing a user to enter data. This data entry can be connected to the input of a gadget that provides a Web-based resource invocation, and the output of the resource response can be connected to a gadget that renders a visual display. Users then can interconnect existing resources with each other to create increasingly complex web services and their APIs, taking up the idea of resources composition based on storyboard-driven creation. End users can then exploit this complex of creations to achieve enormous improvements in their job performance and innovate by creating/remixing their own business tools.

A key challenge of this whole new approach is to create a new visual programming environment that will facilitate the development of composite applications from complex front-end gadgets, involving the execution of relatively complex business processes that rely on traditional back-end semantic Web services. This is the main objective of the FAST project. The adopted approach should be user-centric rather than program-centric. Instead of first building programs that orchestrate available semantic Web services and then trying to figure out how to implement interaction with the users at given points of the process execution flow, programmers will start from the front-end gadgets that the user will see and interact with and then visually establish the connection to back-end Web services by tracing back process execution flows, if necessary. Programmers take an approach similar to the visualization of UML sequence diagrams to visually establish this connection. In this approach, programmers will visually manage and connect front-end gadgets, screen-flow resources and back-end services and overcome the limitations of current business process engine approaches (based on the traditional SOA vision).

Note that the programming tool to be used in this solution should be compatible with existing and future mashup exploitation platforms. Its goal is not to develop the gadget mashup platform. It is a tool that should enable the development of mashupable gadgets that rely on screen-flow resources and semantic Web services stored in a catalogue. In this paper we are going to emphasize the exploitation of created gadgets in the well-known EzWeb mashup platform, which is fully tailored to support complex gadgets intercommunication, multi-device ubiquitous adaptation and ambience creation.

Clearly, this particular solution is a subset of a global user-centric composite application framework, specified around the creation of software based on screen-flows and work-flows that end users could establish via a catalogue of existing semantic Web services.

In summary, this solution aims to define a whole new approach to front-end and back-end integration by developing a new visual programming environment. This new environment will facilitate the development of complex front-end gadgets, involving the execution of relatively complex business processes that rely on back-end semantic Web services and applying the previous guiding principles and its materialization (Section 4 and 5):

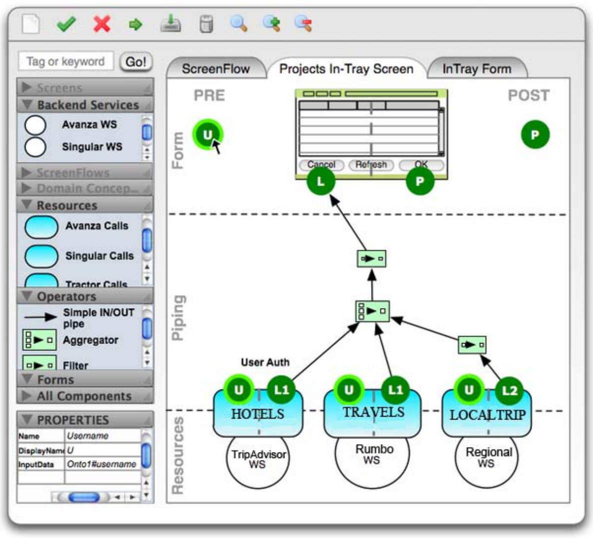
- As opposed to front-end-oriented mashup platforms, which are concerned with facilitating retrieval, mashing and utilization of lightweight gadgets, this platform would go a step further and deal with the creation rather than the utilization of such lightweight gadgets. This can significantly improve programmers' operational efficiency.
- Instead of first building programs that orchestrate available semantic Web services with BPEL and then trying to figure out how to implement interaction with the users at given points of the process execution flow, users will start from the front-end gadgets they will see and interact with, and then visually establish the connection to the back-end Web services tracing back the process execution flows, if necessary. The framework will visually establish this connection adopting an approach similar to the visualization of UML sequence diagrams.
- The goal is to build a system that reads the URIs of a number of semantic Web services and is able to automatically interpret and visualize possible messaging patterns between them for the developer.

Instead of implementing a choreography from scratch, developers and end users have a visual and efficient interface by means of which to orchestrate services according to their needs and to create a gadget on top that clearly conveys its functionality to human users.

7. PROOF OF CONCEPT: APPLICATION OF A COMPOSITE APPLICATION FRAMEWORK TO OUR USE CASE

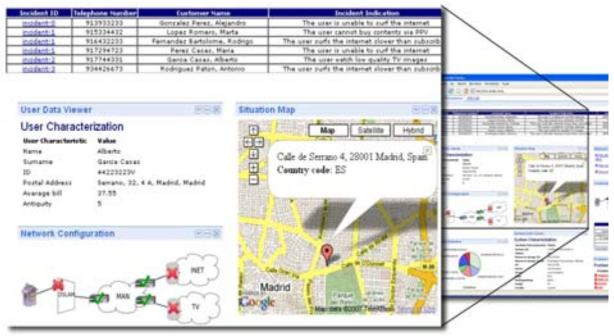
As a proof of concept, this section shows the application of our framework (based on the FAST tool and EzWeb mashup platform) to the domain problem presented in the section 3. This scenario is only one example of the many solutions that could be developed based on the proposed novel user-centric SOA-oriented framework. This section explains a composite application deployed on an existing prototype of the EzWeb platform, where a service-oriented environment is created by visually attaching different complex gadgets to each other and to the enterprise back-end. This specific enterprise mashup environment is useful for an end user responsible for the task of managing services, data, functionalities and resources from back-end legacy systems. Each complex gadget has been created using FAST, putting a face on SOA and leveraging a user-centered top-down approach to Web services as shown in Fig 5. This screenshot shows how a domain expert can create a multi-screen gadget, whose main functionality is to list several corporate services (managed thanks to a REST abstraction layer) on an in-tray screen. The main screen is therefore an in-tray (as a html visual artifact) that is linked to a piping (concatenation and filtering) built on three resources of travel agencies back-end (see Figure 5: three web services associated with different agencies for hotel management, travel management and local travels management respectively that are wrapped with REST resources). By clicking on one of these lines, every end user can deal with a back-end service, through forms and screens that act as a screen-flow wizard to manage that resource's specifics. These visual artifacts are deployed and presented in accordance with the special features of the end user device. In addition, FAST manages facts as internal pre/post conditions (thereby interconnecting building blocks to build complex gadgets) or as external stubs (events and slots) to orchestrate several complex gadgets in an EzWeb mashup.

The zoomed screenshot in Figure 6 depicts a simple scenario when the created in-tray gadget is put together with other gadgets and APIs to create a complex workspace as a composite application that provides trip support for the EzWeb mashup platform end users. More precisely, this dashboard has been extracted from a Telefónica core OSS, which is part of a more general mashup now deployed by Telefónica as a fully operational environment. The mashup connects four gadgets: the list of core Web services deployed in several agencies' back-end, including functionalities to manage user requests (this is in-tray created by FAST), an end user agenda, a Google map and a travel manager. A fully functional environment is created by visually attaching these FAST gadgets to each other and to the enterprise back-end in a wireframing-oriented integration: the agenda gadget will display end user tasks, travels and trip details, the location map will represent the selected element's situation and the gadget for travel management will display the selected travel's information, operations, etc.



Name	Username
DisplayName	U
InputData	Onto1#username

Figure 5. Creation of a complex in-tray gadget using the FAST tool.



Incident ID	Telephone Number	Customer Name	Incident Indication
incident-0	913933233	Gonzalez Perez, Alejandro	The user is unable to surf the internet
incident-1	915334432	Lopez Romero, Marta	The user cannot buy contents via PPV
incident-1	916432233	Fernandez Bartolome, Rodrigo	The user surfs the internet slower than subscrib
incident-1	917294723	Perez Casas, Maria	The user is unable to surf the internet
incident-2	917744331	Garcia Casas, Alberto	The user watch low quality TV images
incident-2	934426673	Rodriguez Paton, Antonio	The user surfs the internet slower than subscrib

User Data Viewer

User Characterization

User Characteristic	Value
Name	Alberto
Surname	Garcia Casas
ID	44223223V
Postal Address	Serrano, 32, 4 A, Madrid, Madrid
Avarage bill	37.55
Antiquity	5

Situation Map

Calle de Serrano 4, 28001 Madrid, Spain
Country code: ES

Figure 6. Creation of an enterprise mashup on the basis of the proposed framework.

It is the end users themselves who develop this "service" to meet their own requirements. They do this on the fly with the help of frameworks like the one proposed in the last section, based on a user-centric approach. Additionally, these systems help users to define and to customize their operational environment, thus improving their use of the entire infrastructure of a real enterprise system in a collaborative and knowledge-driven fashion.

8. RELATED WORK

The Companies are beginning to focus on people as the SOA entry point and therefore to composite applications. Thus they need a means to bridge the gap between people and services. It is then that they come up against traditional composite applications' shortcomings. Consequently, a number of user-centric composite application frameworks are beginning to proliferate. Worthy of note is IBM's solution, named *SOA for people*. It focuses on a portal framework acting as a SOA front-end to maximize people's productivity and collaboration. IBM claims that "with portal and collaboration software, an SOA environment can simplify the way people interact". The increasing interest in this approach is indicative of the current importance of user-centric SOA in the business world.

However, IBM's approach focuses on employing particular Web 2.0-based technologies to deliver a front-end to SOA, instead of reconsidering the whole SOA vision and its application to a composite application domain from an end user viewpoint. In this paper all creation, composition and consumption of new services and applications following a user-centric approach based on enterprise composite applications are completely revisited from an Enterprise 2.0 perspective. Additionally, IBM's approach only considers mashups composed of simple gadgets. Our approach goes a step further and also considers the development of complex gadgets based on the *storyboard* notion. (Storyboarding is an approach used when filming movies, where the use cases and user needs are identified visually in the form of a flow of abstract windows, pages or screens.) Complex gadgets are now composed of gadgets of more than one page/screen and help to deal with real-life workflows and business processes. This is quite an original approach, and is being developed as part of the FAST initiative, a STREP project partially funded under the European Commission's 7th Framework Programme (INFSO-ICT-216048), as part of NESSI, the Networked European Software and Services Initiative. Other similar initiatives are beginning to proliferate in this research field. Of these, the NEXOF-RA project deserves a special mention. Authors collaborate and participate actively in this project, which aims to build the Reference Architecture for the NESSI Open Service Framework by leveraging research in the area of service-based systems, and to consolidate and trigger innovation in service-oriented economies. NESSI is the European Technology Platform dedicated to Software and Services. Its name stands for the **Networked European Software and Services Initiative**.

ServFace is another STREP project funded under the European Commission's 7th Framework Programme related partially to the ideas presented in this paper. This initiative aims to add an integrated UI description and development approach to SOA concepts by introducing the notion of a correspondent user interface for services. This is a completely bottom-up approach: the idea is to enrich Web services and resources with UI descriptions (i.e. in UML) to build generic faces to this back-end. Therefore, it takes a completely opposite approach to this paper's top-down line of attack. ServFace aims to create composite applications from user-created UIs that then are related to an existent enterprise back-end. Taking our approach, UIs are richer, more flexible and closer to end users.

9. CONCLUSION AND FUTURE TRENDS

In this paper, we elaborated on the traditional WS-SOA-based approach to composite applications as a traditional enabler for integrating distant and potentially heterogeneous web resources, data and functionalities. As this traditional approach has neither lived up to its promise of facilitating a global network of loosely interconnected services nor provided an appropriate front-end to deliver composite application domains to people, new approaches are required to rise to this challenge of enabling user-centered composite application development. As a key component of the Web 2.0 era, novel technologies and design principles are now about to emerge that will allow human users to use, customize, combine, interconnect and finally display Web-based content or functionality as new resources on the Web. In this paper we have elaborated on the synergies between the Web 2.0 and the composite application worlds that can be exploited to rise to this important challenge and have proposed a generic model of a global user-centric SOA and a novel architecture for enterprise mashup composite applications.

The appearance of user-centric approaches to next generation service front-ends, such as the one proposed in this paper, will be a major step forward, providing solutions to currently hard-to-solve problems in the traditional SOA paradigm. The emergence of such service architectures will solve key problems in three different scenarios. Large enterprises may capitalize on faster application development (for what are known as instant applications), a more agile system landscape and the empowerment of their employees to design their own applications that best satisfy their unique requirements, and to share this knowledge with other employees better than in traditional Web service architectures.

On the other hand, the proposed architecture enables SMEs to find, customize, combine, catalogue, share and finally use applications that exactly meet their individual demands by leveraging the SaaS model, viewed as Utopian from a traditional SOA perspective. Supported by the new Internet of Services approach, they can select and combine resources hosted by third parties rather than buying a pre-determined, inflexible and potentially heavyweight solution.

Finally, individuals benefit from a strongly increased capability of personalization and participation. This approach will provide end-users with intuitive, unsophisticated IT ways to discover, remix and use those Web-based services that they consider interesting and useful. It will also allow them to participate, swap information with other users and service providers and to actively contribute in a way that encourages extensive use of the resources offered. This speeds up the service innovation pace. Focusing on the"long tail" advanced by Chris Anderson rather than a limited number of sophisticated experts, a user-centric SOA will involve the bulk of private users or small businesses and allow for"customer self-service".

Future work will concentrate on evolving FAST and EzWeb, the open source composite application framework used as proof of concept in this paper. We expect them to become a major hub for the publishing, brokerage, customization and, finally, the consumption of Web-based resources on a global, cross-organizational scale [18].

10. ACKNOWLEDGMENTS

This work is supported in part by the European Commission under the first call of its Seventh Framework Program (FAST STREP Project, grant INFSO-ICT-216048) and by the European Social Fund and UPM under their Researcher Training program.

11. REFERENCES

[1] Alonso, G., Casati, F., Kuno, H. & Machiraju, V.(2004). Web Services Concepts, Architectures and Applications.

[2] Anderson, C.(2006). The Long Tail, Why the Future of Business is Selling Less of More. Hyperion. July 2006.

[3] Davenport, T. H.(2005). Thinking for a Living: How to Get Better Performance and Results from Knowledge Workers. Harvard Business School Press, Boston, MA, USA. 2005.

[4] Fielding, R. T.(2000). Architectural styles and the design of network-based software architectures, Ph.D. thesis, University of California, Irvine, 2000

[5] Gartner Inc. (2006). Hype Cycle for Software as a Service, Gartner Research, 10 August 2006.

[6] Högg, R., Meckel, M., Stanoevska-Slabeva, K. & Martignoni, R.(2006). Overview of business models for Web 2.0 communities, Proceedings of GeNeMe 2006, p. 23-37, Dresden, 2006.

[7] IBM Developer Works (2006). Composite applications – Business Mash-ups, http://www.ibm.com/developerworks

[8] MacKenzie, M. (2006) OASIS - Reference Model for Service Oriented Architecture 1.0, http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=soa-rm

[9] McAfee, A.(2005). Will Web Services Really Transform Collaboration, MIT Sloan Management Review, Vol.46, No.2, 2005

[10] McAfee, A.(2006). Enterprise 2.0: The Dawn of Emergent Collaboration. MIT Sloan Management Review, Vol.47, No.3 (pp. 21-28).Spring 2006.

[11] North, D.C.(1990). Institutions, Institutional change and economic performance, Cambridge University Press, Cambridge, 1990

[12] O'Reilly, T. & Musser, J.(2006). Web 2.0 Principles and Best Practices. O'Reilly radar, November 2006.

[13] O'Reilly, T.(2005). What is Web 2.0: Design Patterns and Business Models for the Next Generation of Software. www.oreillynet.com/pub/what-is-web-20.html

[14] OASIS (2003).Web Services Composite Application Framework (WS-CAF) TC, http://www.oasis-open.org/committees/tc_home.php? wg_abbrev=ws-caf

[15] OASIS Open CSA (2007). Service Component Architecture (SCA), http://www.oasis-opencsa.org/sca

[16] Roman, D. (2005). Web Service Modeling Ontology, Applied Ontology, Vol.1,No.1 (pp. 77 - 106), 2005.

[17] Schroth, C. & Christ, O. (2007). Brave New Web: Emerging Design Principles and Technologies as Enablers of a Global SOA. In Proceedings of the 2007 IEEE International Conference on Services Computing (SCC 2007) (pp. 8)

[18] Schroth, C. & Janner, T. (2007). Web 2.0 and SOA: Converging Concepts Enabling the Internet of Services. IEEE IT Professional Vol.9, No.3(pp.36-41) , June 2007.

[19] Smith, R.(2006). Enterprise Mashups: An Industry Case Study. Keynote at the New York PHP Conference & Expo, Manhattan, New York, USA, 14-16 June 2006

[20] Winewright, P. (2005) Why Microsoft can't best Google, Software as a Service ZDNet editorial, August 2005.