EzWeb/FAST: Reporting on a Successful Mashup-based Solution for Developing and Deploying Composite Applications in the Upcoming "Ubiquitous SOA"

David Lizcano and Javier Soriano
Universidad Politécnica de Madrid, Spain
{dlizcano, jsoriano}@fi.upm.es

Marcos Reyes and Juan J. Hierro
Telefónica I+D, Madrid, Spain
{mru,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 the ubiquitous 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.*

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 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 4). Section 5 describes existing simple prototypes of this framework as a proof of concept, dealing with the use case presented in section 3.. Finally, the last section concludes this paper and presents a brief outline of future work.

2. Shortcomings of WS-SOA on the road towards a Web of services

Many definitions of composite applications have been published since the late 1990s [14]. The widely accepted OASIS Web Services Composite Application Framework (WS-CAF) standard defines composite applications as "...a perspective of software engineering that defines an application built by combining multiple existing functions into a new application" [7]. According to this definition, the major supporting technology for the composite applications paradigm is nowadays a Web services-based SOA. However, it is wrong to assume that composite applications are by definition part of a SOA, since one can build composite applications using any technology or architecture. In fact, there is a relatively new initiative, called Service Component Architecture (SCA) [15], advocated by major software vendors that claim it is more natively suited for the delivery of applications that conform to service-oriented architecture principles in a composite application paradigm.

The central concepts of the composite application framework are enterprise and enterprise-ready sources (e.g., existing resources or even enterprise Web services) of information and functionality (as opposed to current general mashups, which usually rely on web-based, and often free and ungoverned sources). These business sources provide access to capabilities via well-defined interfaces to be exercised following a component contract with constraints and policies. This enables a loose coupling of sources (thereby minimizing mutual dependencies) and complies with some of the probably best-known software engineering principles: information-hiding and modularization. This main idea is often supported by Web services that are provided by entities, the service providers, and are to be used by others, the service consumers.

Components may be composed on the basis of other, existing services, thereby adhering to the principle of reuse. They are autonomous (control the logic they encapsulate only), uniformly described and publicly retrievable via certain discovery mechanisms. Nowadays, IT and Internet evolution has stated that these services should be more easily deployed by providers, as well as accessed and managed by end users than traditional packaged software [20]. Therefore, **services must be accessible for all users** (not only enterprise stakeholders), because these services should replace a previous wide range of enterprise software [5], [17]. Therefore, services should 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, and as flexibly and dynamically as possible.

Business users will see their job support tools replaced by composite applications based on Web services, that are not well enough tailored to users and their routine processes. If these users were at liberty to create their own flexible management tools to support their routine operations, this would encourage business innovation. If they were able to create their own tools from friendly interfaces, they would put substantial innate knowledge of the business into their construction, and this would encourage business innovation and development [3].

Obviously, composite applications based on SOA, as it was originally conceived, represent an architecture focused fundamentally on a B2B context and are not good for B2C problems, since they do not offer the best prospects for dealing with user-service interaction [16]. We can tackle their shortcomings from three different perspectives:

1) Traditional composite application's aim: Conventional composite applications merely aim at facilitating seamless machine-to-machine collaboration. Web services 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 [2] of the Internet.

2) Current composite application's technology: Apart from its aims, this framework relies on a set of complex standards that are not user friendly. 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 via the Web.

3) Component's government: Finally, traditional components, i.e. enterprise services, are subject to clearly defined regulatory frameworks since they mostly exist in the corporate context. The design, provision, maintenance, and coupling of components 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

In this section, we are going to imagine an anonymous end user of any enterprise and her daily user-service interaction. One of tasks she performs as part of her job is to find out and interact with different services, resources and information, either by sharing knowledge with other parties (enterprises, knowledge workers, Web intermediaries and so on) or looking for the information, services and resources herself. In the latter case, she usually spends a lot of time scrutinizing UDDI's Web Services or surfing the blogs, forums and boards of the different service websites that she or her company are acquainted with [16].

This tedious task gets even worse when our end user realizes that, depending on the service she wants to use —which can be provided locally at her company or by another service provider—, she has to fill in application forms with several kinds of data formats to deal with completely different websites where she can either invoke Web services in the application online or get the information to manage the service. She also has to take into account the different invocation processes for properly using the services and enter the information in the correct format.

This end user is fed up with keeping her bookmarks updated with the latest Web services, portals and information used. Also when she knows the required service is new, going through services catalogues like UDDI looking for particular terms and functionality is a waste of time, especially when she doesn't know the provider and service's particularities and she has to find her way [11]. **This end user is looking for some kind of facility that she can use to easily manage different services, resources, information and, ultimately, back-end legacy and its related knowledge by composing an instant application from the services and back-end provided by each service provider, which would be displayed via a single user-centered face, regardless of the data/functionality source or particularities.** She even wants to manage the resources and services personalized according to her tastes and also by the context, discarding the knowledge she is not interested in and presenting the best user interface for her current access device.

Some important service providers, like Amazon, already offer several Web services from their back-end wrapped with a resource layer (similar to a POX-RPC abstraction layer). Initiatives like these offer a uniform way to manage back-end systems easily, but a little chunk of either application or JavaScript code in an html page or a little gadget still needs to be developed to invoke these Amazon Web services. The problem is that our end user knows about business logic in her domain, but is a nontechnical person and has no programming skills.

As a first approximation, some of these service providers could provide end users with isolated gadgets to facilitate the back-end management process. This is a rather significant enhancement with respect to the initial situation and represents the current state of the art of enterprise mashups and Web 2.0 ideas [13]. However, there is room for further improvement: the end user would be grateful for not having her desktop flooded with lots of gadgets that do the same thing, each of which gives her separate access to some isolated service or functionality.

An end user, playing the role of knowledge worker, mostly needs a few (complex) gadgets that provide her with all the relevant business logic she requires, assuring standardized and easy interaction with the desired services and back-end legacy. These (complex) gadgets will be the basic building blocks in our new approach to composite applications development, and their interconnection will be able to support real functionalities and solve real daily problems. Moreover, resources and application gadgets of her interest could be more easily published and discovered by leveraging their semantic description to exploit collaboration and the emergence of collective intelligence in her company [9].

4. Composite application Framework, combining FAST and EzWeb in an Enterprise 2.0 Compositional Platform Architecture

Now that we have looked at the shortcomings of the traditional composite application framework, 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 principles, 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 Fig. 1.

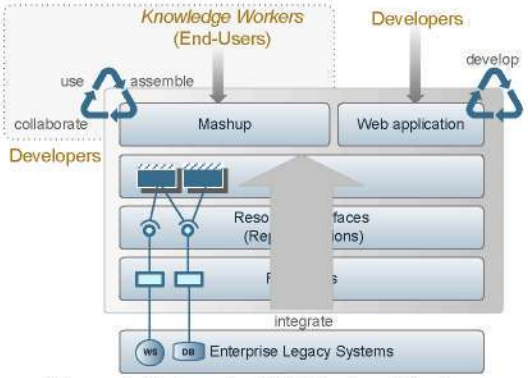

Figure 1. Enterprise 2.0 Mashup Stack.

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 2). 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.
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 3, the composition takes place both in the resource layer (piping) and in the gadget (wiring) layer according to the enterprise mashup stack (see Fig. 1).

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.

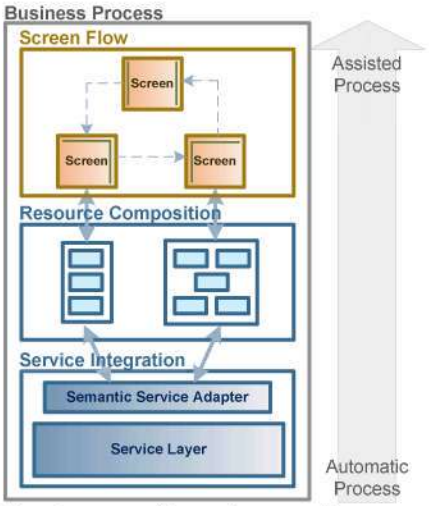

Figure 2. Visual composition of screen-flow resources and interoperability with back-end Web services

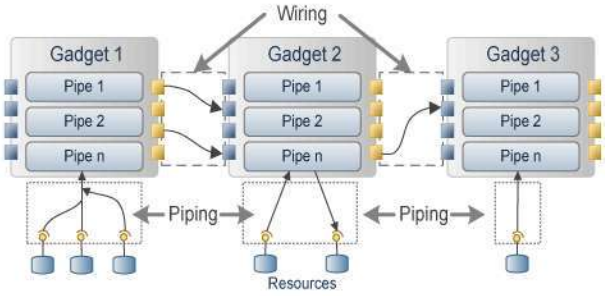

Figure 3. Resource and gadget compositions building composite applications

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 following basic principles:

- 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.

5. Proof of concept: application of a composite application framework

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 4. 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 Telefonica's back-end (see Fig. 4: three web services associated with different projects called "avanza", "singular" and "tractor" 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 5 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 job 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 at Telefónica as a fully operational environment. The mashup connects four gadgets: the list of core Web services deployed in Telefónica's back-end, including functionalities to manage customer requests (this is in-tray created by FAST), a customer agenda, a Google map and a network status map. 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 customer details and have a customer/task selection option, the network map will represent the selected customer's network status and the Google map gadget will display the selected customer's address on a map when a given task is selected from the list.

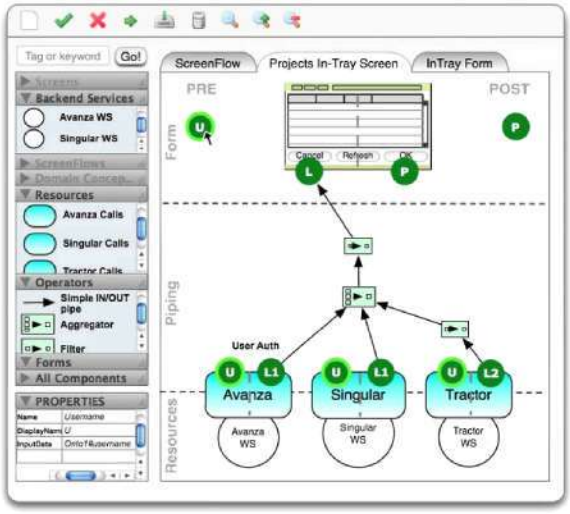

Figure 4. Creation of a complex in-tray gadget using the FAST tool

Figure5. Creation of an enterprise mashup on the basis of the proposed framework

7. Conclusion and future trends

The appearance of user-centric approaches to composite application developments, 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 composite application paradigm. The emergence of such platforms will solve key problems in three different scenarios. Large enterprises may capitalize on faster application development (for what are known as instant applications). There will be a more agile system landscape. .

On the other hand, Enterprise 2.0 collaboration platforms enable SMEs to find, customize, combine, catalogue, share and finally use applications that exactly meet their individual demands by leveraging the SaaS model, Finally, individuals benefit from a strongly increased capability of personalization and participation. The proposed 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.

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].

8. Acknowledgment

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.

9. References

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

[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

[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/a/oreilly/tim/news/2005/09/30/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): IEEE Computer Society. Retrieved 2007-07-11, from http://www.alexandria.unisg.ch/publications/37038.

[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. Retrieved September 18 2007, from http://blogs.zdnet.com/SAAS/?p=13