Formalization of the User Centric SOA Approach*
Implementation and End User Satisfaction Evaluation

Meriem Benhaddi[1], Karim Baïna[2] and El Hassan Abdelwahed[1]
[1]*Faculty of Sciences Semlalia, Cadi Ayyad University, BP 2390-Marrakesh, Morocco*
[2]*ENSIAS, Mohammed V University, BP 713 Agdal-Rabat, Morocco*

Keywords: SOA, Mashup, Integration Patterns, End User Development, End User Satisfaction, Usability, Intuitiveness, Cloud Computing.

Abstract: User-centric SOA is a new paradigm allowing unskilled end users to compose services to create new one. Mashups represent new agile and quick ways to compose and integrate structured and unstructured resources, from different types existing on the web. Mashups emerged as a new way to democratize the SOA and realize the user-centric SOA; However, Mashups are emerging applications, and thus consist of immature, non intuitive and non formalized area. In this paper, we formalize the user-centric SOA development by proposing a new cloud-based architecture for user-centric SOA platforms, and by introducing a new rich integration language based on the advanced Enterprise Integration Patterns (EIPS). We also propose a new intuitive and self-explanatory semantic methodology and interaction model for end users services integration. Through these contributions, we give the promise of realizing the user-centric SOA.

1 INTRODUCTION

1.1 Problems and Limitations of SOA

The concepts behind the Service Oriented Architecture - which consist of modulating applications as interoperable services - has proved that it is the best way to urbanize the enterprise information system by promoting the reuse of services to build more complexes ones, the interoperability between different heterogeneous system, and the standardized languages and protocols (WSDL, SOAP, BPEL). Nevertheless, enterprises that applied SOA didn't get the great promised added value, which has prevented the installation of the global SOA, and has lowered the percentage of companies planning the SOA (Gartner, 2005).

In this section, we introduce the concept of "End User", to signify the non-computer user, who has very little computer knowledge. We will give a further definition of this concept in the next section.

* This Work is partially financed by the research project EvA (vulgarisation of Enterprise Architecture) \no 002/ENSIAS/2011 of Mohammed V Souissi University

The limitations of SOA could be summarized as:

- Exclusion of the end user from the hierarchy of the SOA actors: users kept away and out of the loop. In fact, the SOA technologies (WSDL, SOAP, SCA, BPEL, etc) are hard to master and require advanced knowledge (Nestler et al., 2009; Zhao et al., 2009).
- Rigidity, heaviness and incompatibility of SOA implementations with the real constraints of end users:
  - Lack of accessibility: UDDI registries are dedicated to expert; therefore, end users have to browse different web sites in order to use services. (Hierro et al., 2008) states that SOA was originally designed as an architecture focused fundamentally on the B2B context, and does not offer support for B2C interactions.
  - Lack of flexibility and scalability: SOA technologies cannot support the services composition on the fly: After composition design, implementation, testing and deployment, it becomes very difficult to change the composition logic according to the changing needs of users, as it involves a long life cycle (Liu et al., 2007).

Benhaddi M., Baïna K. and Abdelwahed E..
Formalization of the User Centric SOA Approach - Implementation and End User Satisfaction Evaluation.
DOI: 10.5220/0004446304810488
In *Proceedings of the 15th International Conference on Enterprise Information Systems* (ICEIS-2013), pages 481-488
ISBN: 978-989-8565-60-0

o Lack of mobility: SOA implementation and integration technologies are very heavy for devices with limited capabilities. WSDL and SOAP are instances of complicated XML documents, which makes the WS* services very demanding in terms of computing power, bandwidth and storage (Guinard and Trifa, 2009).

1.2 End Users: Who are they? What Do they Need?

A software end user is a person who interacts with information systems solely as a final information consumer. It's a user with minimal technical knowledge, and who uses the software in the context of daily life or daily work for personal (business or leisure) purposes, without having any intentions to produce other systems (Cypher, 1993, Allison and Kelly, 1992).

End users have many requirements that should be respected by system designers and developers in order to deliver systems satisfying end users. Based on the work of (McCall et al., 1977) and (ISO/IEC 9126-12001), we have grouped into four criteria the end users requirements, which are listed as follows:

- Functional richness is features requested to execute different tasks. A limited set of offered features could be a problem at this level.
- Usability & intuitiveness concern user interfaces, interaction and dialogue mode. Lack of usability results in lack of visibility, feedback, consistency, non-destructive operations, discoverability, scalability, reliability (Norman and Nielsen, 2010).
- Efficiency, reliability, maintainability and portability (ERMP) are difficulties that do not refer directly to system features. Problems could be lack of documentation, performance, security, supportability.
- Personnalizability, customizability is the capability of end user to tailor themselves their systems. Not providing end users with this capability results in useless systems that lack many important features.

Based on this section, we define the user-centric SOA as the expectation of end users, their future hope, and the promise for better information systems. A user-centric SOA offers:

- Empowerment of the end user: Easy and flexible composition on the fly of services by all end users.
- Openness of the Information System to the public: the democratization of SOA and the installation of the global SOA or the Internet of Services (Schroth and Janner, 2007).
- More independence of SOA: the adoption of a variety of interoperable technologies in order to meet the great variety of the web.
- Lightweight SOA technologies: the support of SOA technologies by all mobile devices.

2 STATE OF THE ART

2.1 Mashup Frameworks Limitations

Mashup is a new paradigm of the Web 2.0 (O'Reilly, 2005) – the new generation of the web - that enables the user generation of services by allowing end users to personalize and customize their applications (Hoyer et al., 2009; López et al., 2008; Bradley, 2007). Today, there are a big number of Mashup frameworks on the web, which allow end users to mix visually different heterogeneous resources and thus create new applications called mashups. Mashup frameworks have helped to bridge the gap between end users and software development, but they are still some technical gaps (Benhaddi et al., 2010): Mashup frameworks use lightweight resources (RSS, ATOM, REST services, etc) (Roy, 2010; Nestler, 2008), they do not allow the creation of business process mashups (Nestler, 2008), they do not provide stable applications (Anjomshoaa et al., 2010) and finally they are still outside the scope of end users (Nestler et al., 2009).

These critics show that the Mashup is at an early stage and needs more research. In fact, there is a lack of a powerful integration language, and intuitive design process. Hence, in order to achieve the user-centric SOA, there is a need to introduce new elements consisting of patterns and models to enhance the development of Mashup applications.

The next section introduces the Enterprise Integration Patterns, and shows their contribution to any integration solution.

2.2 Study: Mashup Frameworks and the User-centric SOA

The Enterprise Integration Patterns (EIPs) collected by (Hohpe and Woolf, 2003) describe a number of design patterns for enterprise application integration and message oriented middleware. The EIPs are implemented by a set of sophisticated mediation bus, such as Camel, Mule and Apache, in order to achieve very complex integration scenarios. Enterprise Integration Patterns propose the best and

common solutions to integration problems. Therefore, when EIPs are used, they enhance the quality of the integrated applications. EIPs consist of six groups of patterns: messaging channels, message construction, message routing, message transformation, messaging endpoints and system management. Based on the book of (Hohpe and Woolf, 2003), we categorize these patterns groups according to the four end user satisfaction criteria that we defined and presented in section I.2.

The Enterprise Integration Patterns, when used together, help achieving a high level of system quality by ensuring the end user satisfaction criteria. The use of EIPs is therefore considered as a proof of the system quality. Hence, we had the idea of studying different Mashup frameworks - Yahoo! Pipes (Yahoo! Pipes, 2012), Jackbe Presto Wires (Jackbe Presto Wire, 2012) and IBM Mashup Center (IBM Mashup Center, 2012) - based on the EIPs. Our study showed that the three Mashup frameworks implement a limited set of the integration patterns, which are very basic and simple; the three Mashup frameworks fail to implement advanced and sophisticated integration patterns.

According to this study, we deduced that the three Mashup frameworks fail to totally ensure the criteria of "Functional richness", "Efficiency", "Reliability" and "Maintainability".

We also studied these Mashup frameworks from the usability & intuitiveness perspective. The study showed also that Mashup frameworks lack ease of use and intuitivity for inexperienced end users. Unfortunately we could not introduce this study in this paper because of the restricted number of pages.

Table 1 present the final result of our studies that all showed that the Mashup frameworks are not user-centric SOA solutions.

The next section gives a brief description of our proposed new user-centric SOA solution, by introducing new patterns and methodologies helping to formalize the user-centric SOA development.

Table 1: Mashup frameworks and user-centric SOA criteria.

User-Centric SOA criteria/Mashup Frameworks	Yahoo! Pipes	Jackbe Presto Wires	IBM Mashup Center
Functional Richness	2	2	2
Personnalizability	3	3	3
Usability & Intuitiveness	2	2	2
Efficiency, Reliability, Maintainability and Portability	3	2	2

3=High, 2=Medium, 1=Low

3 USER-CENTRIC SOA PROPOSAL

3.1 A Cloud-based Architecture

We presented the technical architecture of the user-centric SOA in (Benhaddi et al., 2012). This Architecture includes several services required by Mashup platforms. These different services can be homemade (developed internally), or accessible through the Cloud Computing. Indeed, the Cloud Computing can be considered as a new way to retrieve and use IT-enabled services by customers. According to (Buyya et al., 2008), the Cloud Computing is an emerging paradigm that is based on compute and storage virtualization to deliver reliable services to customers. Customers can access data and applications anywhere in the world on demand.

This way, Mashup platforms can rely on the Cloud Computing services to ensure the operation of each layer of the technical architecture. For example, Enterprise Service Buses could be used for their routing and translation capabilities, BPEL engines could be used for their orchestration capability and the CRUD services offer different services such as identity management, persistent storage, resources access, routing and translation.

As stated before in this paper, end users have four requirements: functional richness, usability && intuitiveness, personalizability and infrastructure requirements such as reliability, efficiency, maintainability and portability. As Mashup platforms were created to let end users personalize their applications, we consider that the third requirement is ensured. The fourth requirement is out of the scope of this paper. We focus our work on the first two requirements. The next section is dedicated to the study of the first requirement - functional richness – and provides a solution based on the Enterprise Integration Patterns (EIPs).

3.2 First Requirement: Functional Richness

As it was showed in section 2.2, the Enterprise Integration Patterns enhance the system quality in terms of the functional richness. Therefore, our proposal is based on the Enterprise Integration Patterns.

In order to achieve his task, the end user needs a platform that encapsulates the following elements:

- Objects/resources to integrate: services spread across the web that, together, offer a new service with a new added value.

- Fields on interface allowing the entry of intermediate data.
- Communication channels that allow binding and forwarding the results between different objects.
- Messages of different types which will be carried by channels and sent by one object to another. A message can be of different types: a message representing a document, a message representing an order, etc.
- Routing components whose role is to route the results of an object to another.
- Translation components that transform the results of an object before sending them to another object.

We have identified the different basic elements that will form our future language that we named SOA4EU (SOA for End User). Table 2 lists these elements.

We have realized the formalization of SOA4EU language using Backus-Naur Form (BNF). Because of the pages number restriction, we do not present it in this paper.

The next section focuses on the second requirement – usability & intuitiveness – and presents a methodology helping end users to easily compose services.

3.3 Second Requirement: Usability and Intuitiveness

3.3.1 Goals Composition vs Services Composition

When creating new applications, end users try to achieve a new goal by composing existing sub-goals. Each sub-goal is represented by a service. In this way, when composing services, end users try to resolve a problem whose solution does not exist yet on the web. In fact, the answer exists in the form of many subparts – services – dispersed on the web. Therefore, the inexperienced end user faces many challenges when trying to compose services in response to a new goal:

- Determine the types of resources: what to do?
- Find resources that meet the end user criteria (quality, price, etc.).
- Determine necessary actions for the use of interfaces (selection problems): what and how to use interfaces?
- Determine how to arrange and coordinate resources (integration): how to coordinate the elements?
- Determine the final interface of the integrated resources.

Table 2: Constructs of SOA4EU language.

Construct	Description
Task	is the goal of the end user performing the integration. Each task can have a frequency of execution.
Tag	key words used to describe a task
Mashup	A Mashup application represents the realization of a task and includes a set of integration taking place between several resources.
Process	Is the composition process of the Mashp application resources and consists of parallel or sequential integration flows.
Step	Is a step in the integration process and consists of a link between two or several components.
Component	Is the integration process node: resource, input of the end user, router or translator.
EndUser	Represents the interaction with end users during the integration process.
Resource	Represents the applications to integrate by the Mashup. A resource is described by its type, address and exchange format.
Expose Resource	Represents an exposed resource with input and output variables. The same resource can be exposed many times within the integration process.
Channel	Allows communication between two components and supports the single atomic integration step.
Message	is the entity transferring in a channel between two components.
Router	Is a node forwarding messages between resources, end user fields or translators.
Translator	Is the messages translation node.
System Manager	Each Mashup application can have one or several managers to improve reliability and maintainability.
Transaction	End users may want to synchronize actions of components to realize a transaction.

The system has the role of helping end user to answer these different questions, by suggesting resources, providing guidelines for the coordination of resources and providing feedback and documentation for each selected action.

Faced with these design problems, the end user will use the knowledge he possesses that describe his goal and which consists of the objective or set of operations that the goal task must accomplish, the final result of the goal task (output of the process),

the frequency, the degree of importance and the duration of the goal task execution.

This end-user knowledge represents the semantic which, alone, should be involved in the interaction between the end user and the user-centric SOA platform. Indeed, the service-to-service interaction, which is based on the syntax, is not valid at the interface level. The interface provides gadgets that represent a sub-goal, which is an abstraction of services; therefore, the interaction and communication way at the interface level should also be an abstraction of the communication way between services (Figure 1).

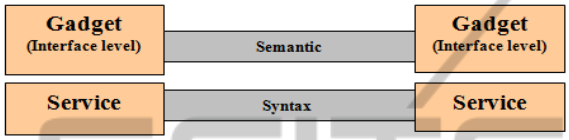

Figure 1: Interaction way on the service level and the interface level.

The user centric SOA platform has to allow to end users to link the various resources in a very intuitive and self-explanatory way, requiring no knowledge of how to map an output of a resource to an input of another. To achieve this, the user centric SOA platform has to provide the end user with a set of goal prototypes or goals patterns, which have the role of guiding the end user through the goals composition process. The next section presents our goals patterns-based suggestion system.

3.3.2 Goals Patterns-based Suggestion System

3.3.2.1 What are the Goals patterns?

In the world of software development, design patterns are solutions or best practices in response to common problems in software design. For example, the "Model-View-Controller" pattern help organizing an application by splitting it into a data model, an interface or a presentation and a controller (control logic, event management and synchronization).

Goals patterns represent common and repetitive use cases, and can also be called end users experience patterns. They provide answers to questions like "How to automate the execution of two consecutive tasks - eg. Turn on the light on the entrance of the house and turn on the heating - in response to a triggered event? - ex. presence of a person detected by the sensor.

The following are examples of goals patterns:

- Booking airline ticket, hotel room and car for a destination.
- Purchase order for a product whose quantity reached a limit value.
- Turning on the room light and the coffeemaker when the alarm clock goes off.

While software design patterns are derived from the experience of the software developers, goals patterns are created, improved and enriched by end users themselves.

3.3.2.2 Suggestion System

The usefulness of the goals patterns is the suggestion system. In fact, end users will be guided in the process of services composition through the database of goals patterns that contains the possible links between the various gadgets. As gadgets represent sub-goals, the database links represent also relations between sub-goals. The system will utilize this goals patterns database to suggest to the end user links and components in order to build new applications.

The suggestion system should be based on the semantic information, as it is explained in section 3.3.1. In fact, the different links between components should be represented by semantic information as input/output matching.

The database of goals patterns being built through the experience of end users, the system will score the various components, depending on the frequency of use, and thus offer to the end user the best one - which has the highest score.

Our suggestion model is similar to e-mail interfaces - ex. Gmail. When writing a new message, and when the first recipient address is entered by the user, other addresses are proposed and suggested at the basis of the previous messages sent by this user.

The goals patterns database elements that constitute also the components of the services composition interface are managed by the following description:

- An end-user profile is described by the age, the types of goals (work, leisure or both) the end user is interested in, the areas of interest, the physical environment.
- A profile is a set of goals.
- A goal is described by its type, its physical environment of execution, its objective, its frequency and its degree of importance.
- The realization of a goal involves several composition steps. A step represents a link from a component to one or several components (one-to-one or one-to-many).

- A component can be another application participating in the composition as a sub-goal, a user input or an operator (translator or router).
- In order to suggest to the end user the appropriate actions, the database must store the various possible relationships between components. Thus, each composition step possesses a relation.
- Each link between two components (composition step) is described by a semantic data that corresponds to the output of the message transmitter and the input of the message receiver.
- The semantic data of a component can be information, event, interface or nothing.
- The participating applications or sub-goals can be synchronized in order to realize a transaction.

The object model of the goals patterns database is represented by Figure 2.

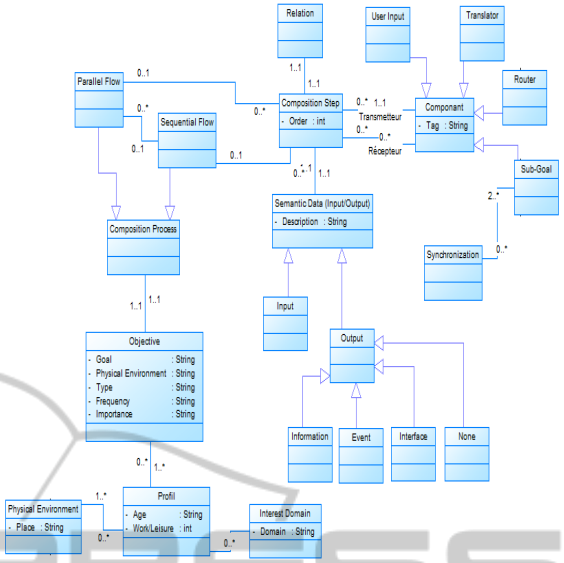

Figure 2: The object model of the goals patterns database.

4 IMPLEMENTATION AND ILLUSTRATIVE EXAMPLE

4.1 Overview of SOA4EU Framework

Our new framework is characterized by being:

- Dynamic: the static services composition used by existing Mashup frameworks does not meet end users needs. In fact, in some situations, end users do not have a clear understanding of how to design the entire composite application; displaying results step by step help end users determine the future actions.
- Semantic: the interaction between end users and our framework use the semantic layer offering an intuitive way to link services and match inputs and outputs.
- Suggestive: our framework helps end users faced with many challenges while designing their future applications. Suggestions ease and accelerate the system learning.

All the framework integration (resources invocation, transformation and routing) are done using the Apache CAMEL integration framework (Ibsen et al., 2011) which was realized based on the Enterprise Integration patterns (Hohpe and Woolf, 2003).

4.2 Evaluation

4.2.1 Demonstration Scenario

To illustrate our new proposal, we choose an example from the public health field. Our end user, Mark, got diabetes with kidney complications. Mark lives in a small city, so he wants to plan a medical consultation with a lower cost by comparing costs in three different neighbouring cities. Then Mark would like to search for kidney doctor addresses in the city with the lower cost, and display the addresses on a map. In order to watch his diet, Mark would like also to have a list of diabetes products that are sold in the supermarkets of the city with the lower medical consultation cost, so he could both visit a doctor and buy the diet products.

In the goals patterns database, there is a set of gadgets that Mark could use and that the platform could suggest to him. The gadgets are represented in four sub-directories depending on their output type (information, event, interface, none). Mark could use semantic tag while searching for a specific gadget. In order to link and adapt gadgets, Mark will also use transformer and router operators: content filter, content enricher, aggregator, content-based router, etc.

4.2.2 End Users Satisfaction Evaluation

The platform-implementation of our model was evaluated by analyzing end users reactions and feedback. We invited twenty individuals to use and test our framework, we then measured their satisfaction against the criteria of integration scenario richness and usability & intuitiveness.

4.2.2.1 Integration Scenarios Richness

The chosen scenario implements various integration

patterns: channel patterns (Point-to-Point, Publish-Subscribe), message construction patterns (command message, document message, event message), message transformation patterns (content filter, message enricher) and message routing patterns (content-based router, message filter, message aggregator).

We asked end users to realize this use case by providing them with a description of the objective to be achieved. We then collected their feedback and reactions about the usefulness and appropriateness of this use case compared to their daily activities. In other words, we asked end users if they sense any interest or gain by using our framework.

The majority of end users feedback was in favour of the usefulness and adequacy of our framework.

4.2.2.2 Usability && Intuitiveness

The usability & intuitiveness criteria is composed of several sub-criteria (Norman and Nielsen, 2010) listed in the table below. We asked end users to measure their satisfaction against these criteria; the results are as follows.

Table 3: Evaluation of the usability & intuitiveness.

Criteria	Description	Satisfaction (%)
Visibility	Accessibility, system learning	80
Feedback	Possibility of cancelling the effect of previous action	40
Consistency	Respect of interfaces design standards	80
Non-destructive operations	Undo operations	40
Discoverabi-lity	All operations can be discovered by systematic exploitation of menus	80
Scalability	The system should work on all screen sizes.	---
Reliability	Operations should work, events should not happen randomly.	80

As shown in the table above, most end users were satisfied with the main criteria of usability && intuitiveness. End users have particularly pointed the usefulness of the suggestions, the use of semantic and the dynamic execution of our framework operations. Indeed, the suggestions can push and accelerate the system learning, visibility and discoverability, by guiding and accompanying the end users in their choices and actions. The semantic and dynamic interfaces facilitate also the system learning and discoverability by hiding any complexity and by giving immediately the result of each performed action.

The feedback and non-destructive operations criteria are a weakness element in our system that we are improving.

We were not able to test our system regarding the scalability criteria. In fact, the hardware we used was a computer with characteristics (processor, memory, screen size) equal or higher than that of a laptop. Materials such as tablet or mobile phone (smartphone) were not used.

5 CONCLUSIONS AND FUTURE WORK

In this paper, we presented the limitations of the Service Oriented Architecture leading to the emergence of the user-centric SOA concept. As a technology allowing the end user SOA implementation, the Mashup remains immature and needs new patterns, failing thus to be user-centric SOA solution. Our contribution aims at the formalization of the end user service creation, consisting of the proposal of a new Cloud-based architecture, a new integration language based on the advanced Enterprise Integrations Patterns and a new intuitive and self-explanatory service creation methodology. The tests that we conducted showed end users satisfaction with integration richness and usability & intuitiveness. Our future work consists of enhancing and completing our user-centric SOA framework.

REFERENCES

Allison, H and R. Kelly, R, 1992. 'The Influence of Individual Differences on Skill in End-User Computing'. *Journal of Management Information Systems I Summer 1992*, Vol. 9, No. 1, pp. 93-111. (1992).

Anjomshoaa, A., Tjoa, A. M. and Hubmer, A., 2010. 'Combining and integrating advanced IT-concepts with semantic web technology, Mashup architecture case study'. Paper presented at *The 2nd Asian Conference on Intelligent Information and Database Systems*, ACIIDS 2010, 24–26 March 2010, pp.13–22, Hue City, Vietnam, Part I, LNAI 5990. (2010).

Benhaddi, M., Baïna, K. and Abdelwahed, E., 2010. 'Towards an approach for a user centric SOA'. Paper presented at *The third International Conference on Web & Information Technologies, Marrakech*,

Morocco, April 2010. ISBN: 978-9954-9083-0-3. Pages: 91-104.

Benhaddi, M., Baïna, K. and Abdelwahed, E., 2012. 'A user centric Mashuped SOA'. *Int. Journal of Web Science. Volume 1*, Issue 3. DOI: 10.1504/IJWS.2012.045812

Bradley, A. (2007) Reference Architecture for Enterprise Mashups, Gartner Research.

Buyya, R., Yeo, C. and Venugopal, S., 2008. 'Market-Oriented Cloud Computing: Vision, Hype, and Reality for Delivering IT Services as Computing Utilities'. Paper presented at *The 10th IEEE International Conference on High Performance Computing and Communications (HPCC-08)*, pages 25{27, Los Alamitos, CA, USA, 2008. IEEE

Cypher, A., 1993. Watch What I Do: Programming by Demonstration. The MIT Press, Cambridge.

Gartner. 2005. Gartner Newsroom http:// www.gartner.com/it/page.jsp?id=790717. (2008). (Accessed:10/06/2012).

Guinard, D. and Trifa, V., 2009. 'Towards the Web of Things: Web Mashups for Embedded Devices'. Paper presented at *The 18th Int World Wide Web Conference*, April, 2009, Madrid, Espagne.

Hohpe, G. and Woolf, B., 2003. Enterprise Integration Patterns: Designing, Building, and Deploying Messaging Solutions, Addison-Wesley Professional.

Hoyer, V., Janner, T., Schroth, C., Delchev, I. and Urmetzer, F., 2009. 'FAST Platform: A Concept for user-centric, enterprise class Mashups'. Paper presented at *The 5th Conference of Professional Knowledge Management*, Poster Session, Solothurn, Switzerland, 25-3-2009, pp.5-8.

IBM Mashup Center. [Online] http://www-01.ibm.com/software/info/mashup-center/ (Accessed : 04 Mars 2012).

Ibsen, C., Anstey. J. and Zbarcea. H.(2011). Camel In Action. Manning Publications.

ISO/IEC 9126-1. (2001) Software engineering – Product quality - Part 1: Quality model. ISO.

Jackbe Presto Wire. [Online]. www.jackbe.com/ (Accessed : 04 Mars 2012).

J. Hierro, J., Janner, T., Lizcano, D., Reyes, M., Schroth, C. et Soriano, J., 2008. 'Enhancing User-Service Interaction Through a Global User-Centric Approach to SOA'. Paper presented at *The Fourth International Conference on Networking and Services IEEE Computer Society*, ICNS '08. Washington, DC, USA (2008).

Liu, X., Hui, Y., Sun, W. and Liang, H., 2007. 'Towards service composition based on Mashup'. Paper presented at *The IEEE Congress on Services*, 9–13 Juillet 2007, pp.332–339, Salt Lake City, Utah, USA.

López, J., Pan, A., Bellas, F., and Montoto, P., 2008. 'Towards a Reference Architecture for Enterprise Mashups'. Paper presented at *The Jornadas de Ingeniería del Software y Bases de Datos*, 7-10 October 2008. Gijón, Spain.

McCall, J. A., Richards, P. K., and Walters, G. F., 1977. Factors in Software Quality, RADC TR-77-369, 1977, Vols I, II, III, US Rome Air Development Center Reports. Italie. (1977).

Nestler, T., 2008. 'Towards a Mashup-driven end-user programming of SOA-based applications'. Paper presented at *The 10th International Conference on Information Integration and Web-based Applications & Services*, iiWAS 2008, 24–26 November 2008, pp.551–554, Linz, Austria.

Nestler, T., Dannecker, L. and Pursche, A., 2009. 'User-centric composition of service front-ends at the presentation layer'. Paper presented at *The 2009 International Conference on Service-oriented Computing*, ICSOC/ServiceWave, 24–27 November 2009. Stockholm, Sweden.

Norman, D. and Nielsen, J., 2010. 'Gestural Interfaces: A Step Backward In Usability'. Interactions' magazine, Volume 17 Issue 5, September + October 2010 ACM New York, NY, USA.

O'Reilly, T., 2005. 'What is Web 2.0 – design patterns and business models for the next generation of software', O'Reilly [Online] 30 September. http://www.oreillynet.com/pub/a/oreilly/tim/news/2005/09/30/what-is-web-20.html. (Accessed: 10 November 2011).

Roy, M., 2010. 'Towards end-user enabled web service consumption for Mashups. International conference on software engineering'. Paper presented at *The 32nd ACM/IEEE International Conference on Software Engineering*, ICSE 2010, Vol. 2, pp.413–416, Cape Town, South Africa.

Schroth, C. and Janner, T., 2007. 'Web 2.0 and SOA: converging concepts enabling the internet of services'. *Journal of IT Professional*, Vol. 9, No. 3, pp.36–41. (2007).

Yahoo! Pipes [Online]. http://pipes.yahoo.com/pipes/. (Accessed: 04 Mars 2012).

Zhao, Z., Laga, N. and Crespi, N., 2009. 'The Incoming Trends of End-user driven Service Creation'. Paper presented at Digital Business: *the first International ICST Conference*, DigiBiz, London, UK, June 17-19, 2009 Springer (Ed.) (2010) 98-108.