SOA: MASHUPS and composite applications

Cristina Leitão
ESGS – Polytechnic Institute of Santarém
2000 Santarém - Portugal
cristina.leitao@esg.ipsantarem.pt

Abstract

An explanation of the current methods used in Web service composition within a Service Oriented Architecture (SOA) is presented. We found that composite business applications can be formed by Web service orchestration using the BPEL programming language, which permits the execution of business process logic. Mashups, simple composite applications that are usually used for ad-hoc situations, do not need to be orchestrated, and could be composed using lightweight Web services. Such Web services are aggregated using current tools that are simple to use. Our study was based on a theoretical literature review, and is useful in understanding ways that different business users can build applications within the SOA, according to their needs and their technical competences.

Keywords: Composite Application, Mashup, Service Oriented Architecture, Web Service composition, Decision Support System.

1. Introduction

Presently the use of business logic is expanding in the enterprises' information systems. The enterprises are having their portfolio of applications substituted by a portfolio of services, whereas the tendency today is to change the enterprises' application architecture into a Service Oriented Architecture (SOA) (Cunha, 2007). An SOA is an application architecture where functions are defined as loosely coupled services with well-defined interfaces that can be called in specified sequences to form business processes (Sherman, 2004). By loose coupling, the services maintain a relationship that minimizes dependencies, and only need to be aware of each other's existence. Resources on a network in a SOA environment are made available as independent services that can be accessed without knowledge of their underlying implementation (Channabasavaiah et al., 2003). The services, that support the requirements of business processes and users, are independent of each other and can be called to perform their tasks in a standard way, without having knowledge of the applications calling them. In the same way, the applications do not need to have knowledge of how the services that they call perform their tasks.

SOA relies on a service-orientation as its fundamental design principal. It is an architecture, and it is independent of any particular set of technologies. This type of architecture allows an organization to maintain a focus on solving business problems and not just application problems. Solutions to business problems can be addressed and implemented in a consistent and methodical way.

SOA was initially intended to address integration problems in the enterprise (Sherman, 2004). In SOA, services are used as building blocks that are composed and coordinated to form business processes. The reusability and flexibility of services leads to integration problems from a top-down perspective rather than the traditional bottom up perspective (Sherman, 2004). Rather than having to first deploy complex integration infrastructure, SOA addresses one problem at a time. Services are consequently created to address specific integration problems and can later be reused to address new problems. Over time, collections of existing reusable services can be used to fully address an organization's integration problems (Sherman, 2004).

Web Services collectively serve as the foundation for implementing SOA (Gortmaker et al., 2004). A Web Service (WS) is a technology that enables the provisioning of functionality, on an application level, or on a business level, by means of a standardized interface in a way that they are easily invoked via internet-protocols (Gortmaker et al., 2004). WS provide a standard means of interoperating between different software applications on different platforms. WS is just another way of specifying remote procedure calls (RPC) (Gortmaker et al., 2004). WS originally were limited to enterprise application integration (EAI), but soon the potential to use WS for Business to Business application integration was realized (Gortmaker et al., 2004). Web services are increasingly being used to expose applications over the Internet. These Web services are being integrated within and across enterprises to create higher function services (Ezenwoye and Sadjadi, 2006). Eventually, the need to integrate WS into business processes arose, and Web Service Orchestration (WSO) was developed to fulfill this necessity.

This paper presents the different ways that the Web services may be aggregated in order to form applications that fulfill business needs. In section two, Mashups, the new concept in the Web 2.0, are described. In section three, the study will describe the ways to form composite applications by Web service orchestration. In section four, mashups in the enterprise, a new concept of application composition is explained. Finally, section five presents the conclusions.

2. Mashups in the Web 2.0

A mashup is a new concept used in the Web 2.0. It is a Website, or a Web-based application, composed of two or more components from different sources, but presented to the user as a single application (Edwards, 2006). Mashups draw upon existing Web applications or data sources and combine these resources to create a new application (Patten, 2005). The main philosophy is that services and data are shared so developers can extend functionality as opposed to spending time duplicating what is already available (Patten, 2005).

Mashups are gaining popularity and are now present throughout the internet. Table 1 shows some examples of mashups that can be found in the internet.

Weather Bonk: using services from Google, NASA, Yahoo!, Microsoft, WeatherBug, and more to offer a comprehensive weather site.

Global Incident Map: Stay informed about the latest terrorism threats as they happen with this site, which utilizes the Google Maps API.
Babelplex Google AJAX Bilingual Search: The Google AJAX Search API is used to allow a bilingual search of the Web and Wikipedia.
Flickr Sudoku: A mashup of a popular online sudoku player and Flickr.
www.chicagocrime.org The site combines reported crime date obtained from the Chicago Police Department with Google Maps. Filters crimes based on: crime type, e.g. assault, battery, burglary, theft; location type, e.g. alley, bank, petrol station, restaurant; date; police district; ZIP code; Ward; or route - defined by the user.
USA Military Intelligence (Havenstein, 2007).

Table 1 – Mashups in the Web

The ProgrammableWeb maintains information about mashups on sites (ProgramableWeb, 2010). In March 2007, 1,796 mashups were already registered in this site, where almost 50% were mapping mashups. Mashups could be classified in the following genres: mapping mashups; video and photo mashups; search and shopping mashups; and news mashups (Merrill, 2006).

The mashup Architecture consists of three components (Merrill, 2006) (Patten, 2005). The first component is the API/content providers, who are the providers of the content being mashed. A mashup may include resources from more than one provider. Another component of the mashup architecture is the mashup site, which is where the mashup is hosted. This is where the mashup logic resides, but it is not necessarily where it is executed. The client's Web browser is where the application is rendered graphically, and where user interaction takes place.

Web Services to build mashups are available throughout the internet. One well known service provider is Google, which offers Google Calendar, Google Maps, Google Search, and so forth. Other well known examples are Amazon E-commerce Service, Blogger, del.icio.us, and Flickr.

2.1. Web 2.0.

According to (Merrill, 2006), mashups are the second generation of Web applications informally known as Web 2.0. that developers' face. In other words, Web 2.0 is what the Web is the next step in the evolution of the Web (Merrill, 2006).

The concept of "Web 2.0" began with a conference brainstorming session between O'Reilly, a well known publisher, and MediaLive International, a well known producer of technology events. In this brainstorm session, it was noted that far from having "crashed", the Web was more important than ever, with exciting new applications and sites popping up with surprising regularity (Lewis, 2006).

Web 2.0 defines what Web sites should look like, methods of interaction, styles of development, and sources of content. Mashups are based on the following three concepts of the Web 2.0 defined by O'Reilly: topical information sharing (Merrill, 2006); open, collaborative approach to content generation (Merrill, 2006); and lightweight programming models (O'Reilly, 2006). By building Mashups, developers share services, and collaborate with each other in content generation.

Lightweight Programming Models in the Web 2.0 were recommended due to the fact that the Web succeeded precisely because it overthrew much of hypertext theory, substituting a simple pragmatism for ideal design (O'Reilly, 2006). Complex Web services stack designed to create highly reliable programming environments for distributed applications have yet to achieve wide deployment (O'Reilly, 2006). O'Reilly defined three Lightweight Programming Models Concepts which are the following: (1) Support lightweight programming models that allow for loosely coupled systems. (2) Think syndication, not coordination. Simple Web services, like RSS and REST-based Web services, are about syndicating data outwards, not controlling what happens when it gets to the other end of the connection (O'Reilly, 2006). (3) Design for "hackability" and "remixability". This third concept turns the barriers to re-use content extremely low (O'Reilly, 2006).

Web 2.0-based lightweight services, are now increasingly being used using the REST architectural approach (Sheth et al., 2006). Whereas the SOAP Web services are provided using Web services stack, the REST Web services are provided by simply providing XML data over HTTP, in a lightweight approach. The REST protocol uses the classical HTTP GET-POST approach to invoke services, Uniform Resource Identifiers (URI) to represent resources, and messages encoded in XML for communication (O'Reilly, 2006). Although SOAP has advantages in terms of greater tooling support and the ability to support quality of service guarantees (for example, security and transactions), the ease of use and lightweight style of interaction over the Web using XML have made REST services very popular (Sheth et al., 2006).

Amazon's Web services offer the choice of one of the protocols SOAP or REST. While some high value Business to Business connections use the SOAP stack, Amazon reports that 95% of the usage is of the lightweight REST service (O'Reilly, 2006). According to (O'Reilly, 2006), "Google Maps" simple AJAX interface was quickly decrypted by hackers, who then proceeded to remix the data into new services. Google Maps set the world on fire because of its simplicity (O'Reilly, 2006). Although some mashups are still using SOAP, the trend is to follow O'Reilly's lightweight programming models and abandon the Web service stack by substituting it by REST services. Besides REST or SOAP, other technologies may be used to build mashups (Merrill, 2006) (Patten, 2005), which are illustrated in Table 2.

XML
RSS/Atom XML-based content syndication standards provide an easy way to push data to users that subscribe to the feed.
AJAX This Web application model brings together various technologies (JavaScript and XML) to enable the asynchronous loading and displaying of data.
Web protocols (SOAP, REST*)*
Screen scraping The old-fashioned approach to pulling data from other sites via simply retrieving or scraping data from a Web page.
Semantic Web and RDF Semantic Web creates Web infrastructure that augments data with metadata to give it meaning, thus making it suitable for automation. Resource Description Framework (RDF) provides methodologies to establish syntactic structures that describe data.

.

Table 2 – Mashup Technologies (Patten, 2005) (Merrill, 2006).

The most common way to create mashups is by using REST and AJAX technology, but since mashups have gained so much popularity in the Web 2.0, many vendors are commercializing software to facilitate the conception of mashups. The tendency is to have tools that make mashups easier to compose. Table 3 shows some of the tools that are available in the market to build mashups.

Above All Software Back to create mashups applied only to music.
Adobe With improved XML support, Flex 2.0 real-time, data-driven Flash applications.
Applibase DataMashups.com- For SMBs and workgroups to build hosted enterprise portals combining Internet and internal data.
BEA WebLogic Portal 9.2 - Enterprise portal server.
Dapper Short for "data mapper." -Mashups without programming.
Kapow Technologies Mashup Server
IBM - QEDWiki enables business users and developers to build ad hoc mashup applications quickly and collaboratively.
Laszlo Systems - Open source ECMA

script platform that enables deployment of Internet applications to DHTML or Flash.
/**n software** RSSBus - tools and services to create RSS Feeds with databases, spreadsheets, etc. into HTML pages, Word documents, and more as targets.
NexaWeb The Enterprise Web 2.0 Suite builds a Universal Client Framework.
Oracle As part of Application Server 10g release 3.

Table 3 – Mashup Platform Vendors

Source: D. Linthicum , “Practical Enterprise Mashups”

3. Composite Applications by orchestration in the enterprise

The enterprises’ SOA has been implemented by Web Service composition resulting in composite applications. A Web Service composition combines services following a certain composition pattern to achieve a business goal, solve a scientific problem, or provide new service functions in figure 1 (Curbera et al., 2003). Service compositions may themselves become services, making composition a recursive operation (Curbera et al., 2003). Services are the basic building blocks out of which new applications are created, and service composition becomes the main concern of the application development process (Curbera et al., 2003).

As a result of Web service composition, we have composite applications. A composite application, which contains multiple services used in combination (OASIS, 2007), consists of functionality drawn from several different sources within an SOA (Wikipedia, 2009). A Service-Oriented Business Application (SOBA) is a composite application composed of services that implements a business process (Shmelzer, 2006). The components of such applications may be individual Web services, selected functions from within other applications, or entire systems whose outputs have been packaged as Web services (Wikipedia, 2009).

The traditional WS technology is based on a combination of XML/SOAP over HTTP. The whole WS protocol stack consists of a collection of protocols that are needed to implement a SOA. This collection of protocols is designated as the Web services stack, which is shown in Figure 1 (Gortmaker et al., 2004). As shown in Figure 1, the WS stack is composed of various layers. The interface of a WS is defined by means of WSDL, Web Service Definition Language. Relevant WS can be discovered in a WS-directory through UDDI (Universal Description, Discovery and Integration), and can be invoked by means of messages that are defined in SOAP (Simple Object Access protocol) (Gortmaker et al., 2004).

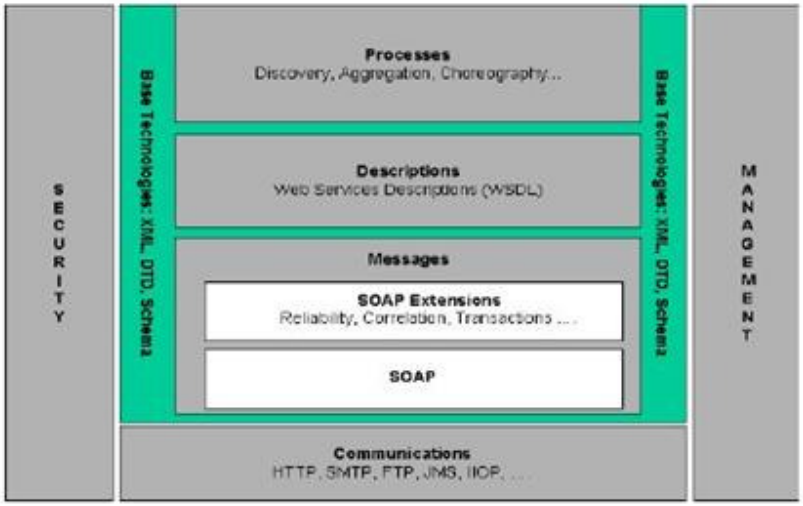

Figure 1 – The Web service stack

Source: J. Gortmaker, and all, " The Advantages of Web Service Orchestration in Perspective"

3.1. Web Service Orchestration

The composition of services may be done in unconstrained ways of Web service aggregation or in constrained ways of Web service aggregation which implies some sort of coordination between the constituent services (process-flow) (Gortmaker et al., 2004). The composition of Web services in constrained ways may be accomplished by WS choreography and WS orchestration. WS choreography are rules that govern behavioral characteristics relating to how a group of Web services interact (Earl, 2004). These rules include the sequence in which Web services can be invoked, conditions that apply to this sequence being carried out, and usage pattern that further defines allowed interaction scenarios (Earl, 2004).

Although choreography is commonly used to aggregate Web services, Web service orchestration has been widely accepted as the means of aggregating Web-services, and consequently executing business logic. In this type of WS aggregation, an orchestration engine is used to encapsulate and execute business process logic (Earl, 2004). By involving a number of different applications or data sources, a process can introduce complex rules and workflows composed of new business rules, exception handling, and transaction management features. The orchestration managed messages are, therefore, more sophisticated than standard SOAP messages. WS orchestration supports louse coupling, since the applications do not need to be aware of each other's existence. The applications are only aware of the process instance as its sole external of contact for all communication outside of its application boundary. The applications simply forward requests to and receive requests from the orchestration component representing the process (Earl, 2004). As can be seen in Figure 2, the key components participating in orchestration are the orchestration engine, each application's logic, and each application's data.

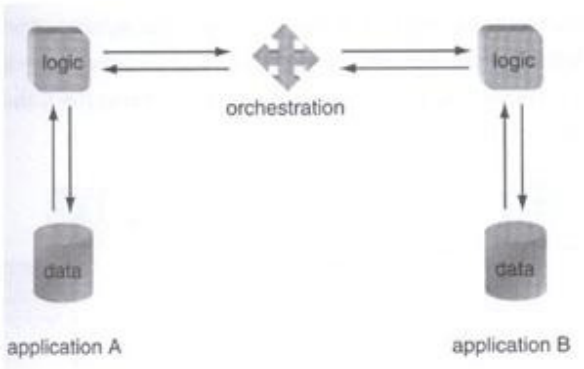

Figure 2 - The key components participating in process orchestration.

Source: T. Earl, Service Oriented Architecture, A field Guide to Integrating XML and Web Services

Figure 3 illustrates that the orchestration engine executes a wide variety of processing options, all depending on the business process it represents.

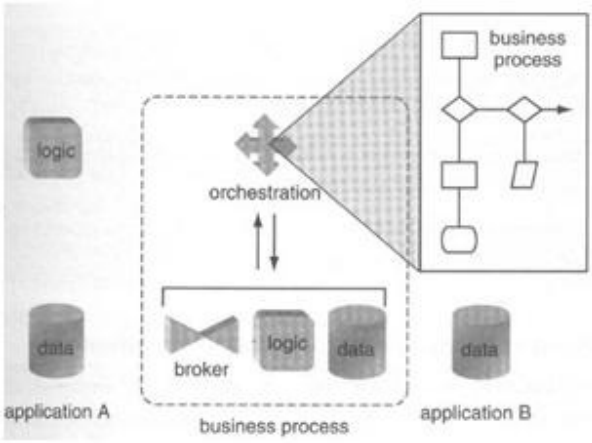

Figure 3 -. The orchestration engine executing the business processes

Source: T. Earl, Service Oriented Architecture, A field Guide to Integrating XML and Web Services

3,2. Business process execution language (BPEL)

The programming language used for WS-orchestration is Business Process Execution Language for Web Services (BPEL4WS) or BPEL (Gortmaker et al., 2004). BPEL was developed by Microsoft, IBM, and BEA, and it unifies XLANG and WSFL. BPEL has been accepted as a standard by the Organization for the Advancement of Structured Information Standards (OASIS), which is a non-profit, international consortium that drives the development, convergence and adoption of e-business standards. BPEL focuses on

supporting composition and coordination of services into business processes and new compound services (Gortmaker et al., 2004). It separates process logic from the rest of the application, and it promotes more collaborative development. Process developers, business analysts, and other skilled professionals engage where they fit best at various stages of design, development, deployment, and management (Gortmaker et al., 2004). The creation of BPEL was driven by the need to facilitate flexibility, visibility, and ease of management at the process layer of the WS stack. BPEL focuses on supporting composition and coordination of services into business processes and new compound services. As shown in Figure 4, BPEL is used at the Process Layer of the Web services stack.

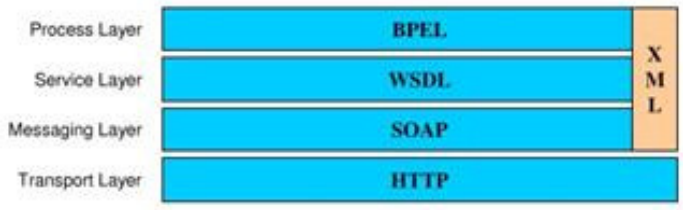

Figure 4. BPEL- the process layer.

Source: O. Ezenwoye, and Sadjadi, S. “Composing Aggregate Web Services in BPEL”.

BPEL has a standardized syntax that is defined in XML, and most users use proprietary graphical notations supported by the various BPEL editors. BPEL’s basic activities include the invocation of a remote service, the deterministic or non-deterministic receipt of messages, the assignments of XML data structures to variables, exception handling and creation, wait operations, etc (Chen et al., 2006). It defines flow elements that support conditional executions and the definition of various loops as they are known from other structured programming languages.

BPEL is used in combination with a number of other standards which are the following: Web Services Description Language (WSDL); Simple Object Access Protocol (SOAP); eXtended Markup Language (XML); Universal Directory and Discovery Interface (UDDI); and WS-Security (Chen et al., 2006). BPEL relies on WSDL to ensure that the invocation of a service is correctly typed, and also to deduce the synchronization behaviour of this invocation (Chen et al., 2006). A process defined in BPEL may expose a WSDL interface itself, which enables the invocation of that process from remote hosts through standard Web service mechanisms. The fact that such a Web service is a service orchestration remains entirely transparent to clients (Chen et al., 2006). A Web service is invoked by a BPEL interpreter using SOAP, which defines the exchange of messages that are encoded in the XML. The structures of these messages, as well as the syntax of BPEL itself are defined using XML Schemas. BPEL uses XPath both to extract data from XML messages and assign them to internal variables and also to compose the parameters for invocations from such variables (Chen et al., 2006). BPEL is often used together with implementations of the UDDI, which is capable of registering and locating services across autonomous domains (Chen et al., 2006). Because orchestrations span autonomous domains, BPEL is often used

jointly with WS-Security for encryption of SOAP messages and proof of origin(Chen et al., 2006).

The BPEL specification attempts to meet the following requirements for the orchestration of business processes: (Ezenwoye and Sadjadi, 2006).

1) The ability to adequately represent the business logic of the process;
2) The ability to provide asynchronous as well as synchronous invocations of Web services;
3) The ability to support long-running transaction;
4) The ability to manage failures, exceptions and recovery.

Since some real examples of orchestration with BPEL were found in B-On (Ezenwoye and Sadjadi, 2006) (Fu et al., 2004) (Chen et al., 2006) (Zimmermann et al., 2005) (Tai et al., 2004), mainly in Telecommunications companies , BPEL has been proven to work well in the orchestration of Web services. Some Well-Known vendors of BPEL are BEA Systems, IBM, Microsoft, and Oracle.

3.3. Business process modelling notation (BPMN)

The Business Process Modeling Notation (BPMN) is a standardized graphical notation for drawing business processes in a workflow (Wikipedia, 2009). BPMN may also be used to Model BPEL Processes. The primary goal of BPMN is to provide a standard notation that is readily understandable by all business stakeholders (Wikipedia, 2009). Such business stakeholders are the business analysts who create and refine the processes, the technical developers responsible for implementing the processes, and the business managers who monitor and manage the processes (Wikipedia, 2009). BPMN is intended to serve as common language to bridge the communication gap that frequently occurs between business process design and implementation (Wikipedia, 2009). BPMN is supported with appropriate graphical object properties that will enable the generation of executable BPEL (Earl, 2004). Through the use of this notation business stakeholder are able to create BPEL processes without having to know how to program in BPEL, making the creation of BPEL processes accessible to most users.

4. Mashups in the Enterprise's SOA

Since mashups are a new concept in the enterprise, their place in the SOA is not yet well defined. Authors' opinions differ in respect to the definition of mashups in the SOA and the use of mashups in the enterprise.

In "Semantics to Energize the Full Service Spectrum", a Communications of the ACM, by A. Sheth, K. Verma and K. Gomadam, the following affirmations are made "One of the most popular applications of lightweight Web services is called a mashup, which is basically a Web site that aggregates content from different providers. A mashup uses lightweight services to query the providers to get content in XML format." (Sheth et al., 2006).

In "Mashups Start to Make More Sense for Business", an article published in InformationWeek, M Hayes quoted "Mashups are one of those fuzzy Web 2.0 concepts with questionable business value. As tech vendors roll out products to make them easier to build, they are likely to get more interesting to businesses." (Hayes, 2007). The author

proceeds to describe how vendors are working on commercializing mashup building tools, and uses as an example Oracle's WebCenter Suite, which includes a Java 2 Enterprise Edition development environment, Web services, and developer tools for creating mashups and other Web 2.0 work environments. Another example that M.Hays uses is IBM's QEDWiki, a free hosted service that developers can use to assemble mashups (Hayes, 2007). This tool helps business analysts and other business professionals build mashups without having to program. It uses Web services such as news feeds, weather reports, maps, traffic conditions, and so forth, to allow users to create customized business applications. It allows clients to mix the existing data and information from inside their businesses with the Web 2.0 services available for free on the Internet. Much information about this new tool can be found throughout the internet, including videos with demonstrations in YouTube.

In "Emerging Trends in SOA: Rich, Smart, Mashed, and Governed", by Ronald Schmelzer, Senior Analyst of ZapThink, LLC, an industry analysis firm focused exclusively on Service Oriented Architecture, the author claims that a mashup is a flexible composition of services within a rich user interface environment. Ronald Schmelzer also makes the following affirmations "In essence, a Mashup is a SOBA interface."; and "The mashup: leveraging the Web to compose Services into ad hoc applications." (Shmelzer, 2006). Figure 5 shows how Schmelzer sees mashups in the enterprise. According to the author, mashups coexist within the enterprise's SOA and the Web 2.0, crossing both of the boundaries.

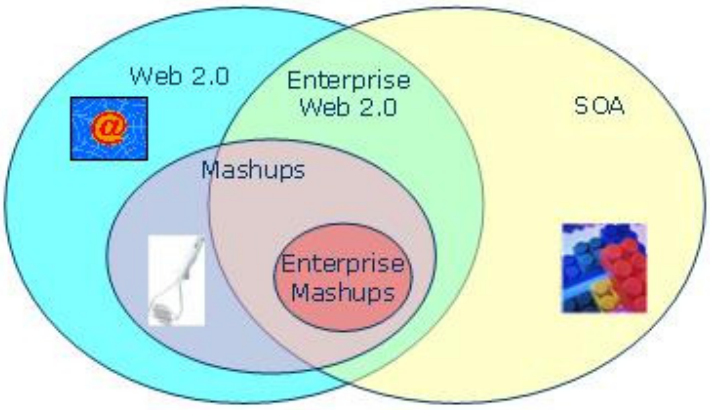

Figure 5 -. Enterprise Mashups

Source: R. Shmelzer, "Emerging Trends in SOA: Rich Smart, Mashed, and Governed"

Because of the intersection of the mashups in the SOA and the Web 2.0, the author is of the opinion that without management and governance, mashups will never be appropriate in an enterprise environment. He believes that governance is, consequently, the key to the Enterprise Mashup, and that without Governance, Mashups are Dangerous (Shmelzer,

2006). The risks: of having enterprise mashups without governance are confidentiality breaches, unauthorized capabilities, and fraud (Shmelzer, 2006).

According to Ronald Schmelzer, Service-Orientation is changing and IT must respond to change, empower users, and enable innovation (Shmelzer, 2006). In his opinion, until now, focus was on building Services, but now the focus is moving to consuming services. Whereas the focus has been on the SOA infrastructure, legacy enablement, identifying reusable services, and building loosely coupled services, the focus is now moving to finding the right service for the task at hand, composing services into SOBAs, and supporting rich user interfaces for services and SOBAs (Shmelzer, 2006). In his opinion, business users (analysts, process architects) should be able to find, identify, understand, and assemble available services in order to compose those services (Shmelzer, 2006). The business users consequently need a flexible tool appropriate for the task at hand (Shmelzer, 2006). The tool may be a Process flowcharting tool, a Mind mapping tool, a Rules engine, a Spreadsheet, or a combination of these tools, depending on the necessity of the task at hand. As previously described in this paper vendors are attempting to commercialize tools with such purposes, making mashups easier to build.

In the article “Practical Enterprise Mashups” by D. Linthicum, published in InfoWorld, the following affirmation is made: “Although mashups originate with Web 2.0, which epitomizes development on the fly, mashups in the enterprise, require preparation.” (Linthicum, 2007). D. Linthicum makes the distinction between practical mashups, which are simple composite applications, and the complex mashups, which are the traditional SOA’s composite applications. He wrote: “More complex mashups approach composite applications (those that are made up of many services), an advanced SOA concept.” (Linthicum, 2007).

He proceeds to claim that “By linking the new components of Web 2.0 with our own sets of information and services, mashups provide a quick and easy way to solve many of today's simple business problems - and should scale nicely to solve more complex and far-reaching problems in the future.” (Linthicum, 2007). According to the author, “An enterprise that can't see the new Web will have a huge strategic disadvantage in the years to come.” The author believes that it will take some time before enterprises are prepared to leverage mashups beyond the browser, and the best approach is to design and deploy an SOA with mashups in mind (Linthicum, 2007). He claims that the enterprise's systems should be made exposable to services or applications outside of its firewall, or able to consume those same services or applications (Linthicum, 2007). The services that are not owned by the enterprise need to be catalogued and tested, and an attempt should be made to mash up systems inside and outside of the firewall. The security planning should also consider this notion (Linthicum, 2007).

According to Linthicum Practical Mashup preparation can be divided into six stages which are the following: (1) Requirements; (2) Design; (3) Governance; (4) Security; (5) Deployment; and (6) Testing (Linthicum, 2007). In the first stage of Requirements, the author recommends that some sort of requirements document for mashups is needed to surface the issues prevalent to the enterprise (Linthicum, 2007). The author advises that the unique business drives should be considered, as well as the current architecture. The mashups that will be valuable should be identified, and how much change must occur to be able to build those mashups should be estimated. The second stage of Design consists in figuring out how the systems should be configured and which enabling technology and standards should be applied to provide the best SOA-based mashup platform. Some key

questions in this stage are what interfaces should be exposed and how; how will scalability be handled; how will visual and nonvisual mashups be approached; how will services and interfaces delivered over the Web be leveraged; how will the exposure of interfaces and services to others on the Web be managed (Linthicum, 2007). The third stage is Governance, which is the creation and enforcement of design time and runtime policies. Given that mashups are made up of services and may become services themselves, services may need to be managed across their entire lifecycle (Linthicum, 2007). It is necessary to build and maintain a mashup-aware registry/repository (Linthicum, 2007). It is also necessary to avoid overloading mashups with policies and procedures, or else developers will be discouraged from creating them (Linthicum, 2007). Security is the fourth stage in Practical mashup preparation and it is critical, because interfaces, content, and services that the enterprise neither created nor owns are being leveraged (Linthicum, 2007). According to Linthicum, security policies and technology layers should be implemented with the objective of protecting the value of the mashup platform (Linthicum, 2007). Mashup security should mesh with the enterprise's SOA security or become an extension to it (Linthicum, 2007). In the fifth stage, for the proper Deployment of mashups, the proper enabling technology and standards must be selected (Linthicum, 2007). How the technologies chosen will link to governance and security should be considered (Linthicum, 2007). The key products leveraged to support mashups within the SOA, and how will they be linked to the enabling technology solutions already implemented should also be considered (Linthicum, 2007). In the last stage of practical mashup preparation, Testing should be completed by considering all sorts of usage patterns and creating a test plan that reflects them (Linthicum, 2007). It should be ensured that the enterprise's SOA and external "mashable" components can work well together, and that the enabling technology and standards are fulfilling all the expectations (Linthicum, 2007). The test plan should be linked with design, governance, and security (Linthicum, 2007).

In the paper "Enterprise Information Mashups: Integrating Information, Simply", presented in an ACM conference, hypothesis are made about a new class of integration technologies designated by "enterprise information mashup fabric" (Jhingran, 2006). These technologies will serve to fulfil the need of Situational Applications, which are applications that come together for solving some immediate business problems. Another task of the "enterprise information mashup fabric" is augmenting structured data with unstructured information.

5. Conclusions

The enterprises' composite business applications can be formed by Web service orchestration using the BPEL programming language, which permits the execution of business process logic. Enterprise Mashups, simple composite applications that are usually used for ad-hoc situations, do not need to be orchestrated, and could be composed using lightweight Web services. Such Web services are aggregated using current tools that are simple to use.

Web service orchestration is necessary for the applications that are more complex and critical to business. The development of the traditional composite applications, orchestrated using BPEL, was a hard task. BPMN simplified this task by introducing a graphical notation for drawing business processes in workflow charts that enabled the generation of executable

BPEL. Through Web service orchestration, applications adequately represent the business logic of the processes; provide asynchronous as well as synchronous invocations of Web services; support long-running transaction; and manage failures, exceptions and recovery. It is obvious that orchestration using .BPEL, or another orchestrating language, is inevitable in the enterprise. Business critical processes need to be mapped into business critical services to form composite applications, and this could be achieved by Web service orchestration.

Mashups are regarded by most authors as Web 2.0 applications. The term "enterprise mashup" is used to describe Web applications that combine content from more than one source into an integrated experience, which share many of the characteristics of SOBAs (Wikipedia, 2009). There is an ongoing debate by most of the authors about "the collision of Web 2.0, mashups, and SOA" (Wikipedia, 2009). Bringing together all the definitions presented by the authors referenced in the current study, it seems to be of common opinion that because mashups are Web 2.0 applications, they use lightweight services and are associated to the lightweight programming models recommended for the Web 2.0. Most authors believe that mashups should be simple and used only for ad-hoc situations. According to some authors, governance is very important, due to the fact that mashups cross the enterprise barriers into the Web 2.0.

Mashups are becoming very popular in the Web 2.0, due to the simplicity of their development, and for this reason, mashups are presently emerging in the enterprises. The authors share the opinion that mashups should be easily developed with appropriate tools that are easy to use by business stakeholders. Software vendors are commercializing tools to help build mashups in an easy way, without the knowledge of any programming language. In the authors' opinions, this type of tools permits all business users to use the services and resources that are available to them both inside and outside the enterprise. The business users are finally able to easily consume the services and mesh them up with other services in order to respond to business needs.

REFERENCES

CHANNABASAVAIAH, K., HOLLEY, K. & TUGGLE, E. M. 2003. Migrating to a service-oriented architecture. *IBM DeveloperWorks,* 16.

CHEN, L., WASSERMANN, B. & EMMMERICH, W. Year. Web Service Orchestration with BPEL. *In:* 28th international conference on Software engineering, 2006 Shanghai, China. International Conference on Software Engineering, 1071-1072.

CUNHA, P. R. 2007. Advanced Topics on Information Systems. Coimbra, Portugal: Department of Computer Engeneering, University of Coimbra.

CURBERA, F., KHALAF, R., MUKHI, N., TAI, S. & WEERAWARANA, S. 2003. The next step in Web services. *Commun. ACM,* 46**,** 29-34.

EARL, T. (ed.) 2004. *Service Oriented Architecture, A field Guide to Integrating XML and Web Services,* Upper Saddle River, NJ, USA: Prentice Hall.

EDWARDS, R. 2006. Riding the Enterprise Mashup. *Enterprise Networks and Servers* [Online]. Available: http://enterprisenetworksandservers.com/opinionw/art.php?266 [Accessed March 31, 2007].

EZENWOYE, O. & SADJADI, S. M. 2006. Composing aggregate web services in BPEL. *Proceedings of the 44th annual Southeast regional conference.* Melbourne, Florida: ACM.

FU, X., BULTAN, T. & SU, J. Year. Analysis of Interacting BPEL Web Services. *In:* 13th international conference on World Wide Web, International World Wide Web Conference, 2004 New York, USA. 621-630.
GORTMAKER, J., JANSSEN, M., REN\, \#233 & WAGENAAR, W. 2004. The advantages of web service orchestration in perspective. *Proceedings of the 6th international conference on Electronic commerce.* Delft, The Netherlands: ACM.
HAVENSTEIN, H. 2007. Military Intelligence Goes Web 2.0. *Computerworld.* [Online], 41.
HAYES, M. 2007. Mashups Start To Make More Sense For Business. *InformationWeek* [Online].
JHINGRAN, A. 2006. Enterprise information mashups: integrating information, simply. *Proceedings of the 32nd international conference on Very large data bases.* Seoul, Korea: VLDB Endowment.
LEWIS, D. 2006. What is Web 2.0? *ACM Press* 13.
LINTHICUM, D. 2007. Practical Enterprise Mashups. *InfoWorld* [Online], 29.
MERRILL, D. 2006 Mashups: The new breed of Web app. Available: http://www 128.ibm.com/developerworks/xml/library/x mashups.html [Accessed March 31, 2007].
O'REILLY. 2006. *What Is Web 2.0", O'Reilly* [Online]. Available: http://www.oreillynet.com/pub/a/oreilly/tim/news/2005/09/30/what is Web 20.html?page=4 [Accessed March 31, 2007].
OASIS. 2007. *Organization for the Advancement of Structured Information Standards* [Online]. Available: http://www.oasis open.org [Accessed May 10, 2007].
PATTEN, T. 2005. Mashups put a new face on the Web. *Techrepublic* [Online]. Available: http://articles.techrepublic.com.com/51003513_11-6156271.html [Accessed March 31, 2007].
PROGRAMABLEWEB. 2010. *ProgramableWeb* [Online]. Available: http://www.programmableWeb.com/ [Accessed December 10, 2010].
SHERMAN, D. 2004. BPEL Unleashed. *The World's Leading Java Resource* [Online]. Available: http://java.sys con.com/read/44668.htm [Accessed March 31,2007].
SHETH, A., VERMA, K. & GOMADAM, K. 2006. Semantics to energize the full services spectrum. *Commun. ACM,* 49**,** 55-61.
SHMELZER, R. 2006. *Emerging Trends in SOA: Rich Smart, Mashed, and Governed* [Online]. Available: http://colab.cim3.net/file/work/SOACoP/2006_10_3031/Presentations/RSchmelzer10292006.ppt [Accessed May 14, 2007].
TAI, S., KHALAF, R. & MIKALSEN, T. 2004. Composition of coordinated web services. *Proceedings of the 5th ACM/IFIP/USENIX international conference on Middleware.* Toronto, Canada: Springer-Verlag New York, Inc.
WIKIPEDIA. 2009. Available: http://www.wikipedia.org/ [Accessed May 10, 2007].
ZIMMERMANN, O., DOUBROVSKI, V., GRUNDLER, J. & HOGG, K. 2005. Service-oriented architecture and business process choreography in an order management scenario: rationale, concepts, lessons learned. *Companion to the 20th annual ACM SIGPLAN conference on Object-oriented programming, systems, languages, and applications.* San Diego, CA, USA: ACM.