Web-oriented Architectures: On the Impact of Web 2.0 on Service-oriented Architectures

Gunnar Thies, Gottfried Vossen
European Research Center for Information Systems (ERCIS)
University of Muenster
Leonardo-Campus 3, 48149 Muenster, Germany
{guth|vossen}@wi.uni-muenster.de

Abstract

Service-oriented architectures (SOAs) have been discussed intensively in science and industry since their first appearance in the mid-nineties. However, even today there is no generally accepted method for modeling a SOA or for converting other concepts (e.g., monolithic architectures, CORBA landscapes) into a SOA. There are numerous standards available in the SOA context, but instead of easing the creation of a SOA, they typically make their realization more complicated. On the other hand, recent developments in the Web 2.0 context show how easy it can be to link or compose ("mesh") IT components dynamically, so that original SOA goals like flexibility or reduction of complexity can be achieved by relatively simple means. An interesting concept in this context is the Web-oriented architecture (WOA), which represents a specialization of SOAs obtained by using simple Web 2.0 technologies and standards. Since for a WOA no generally accepted modeling techniques are available either, this paper reviews SOA models in light of their applicability to WOAs. A case study is used to discuss a possible realization and to preliminarily investigate the question of whether the intended simplicity can indeed be achieved.

1. Introduction

Service-oriented architectures (SOAs) have been discussed intensively and controversially in science, industry, and the literature since their first appearance in the mid-nineties. The properties that characterize a collection of IT components as a SOA are beyond dispute; they include services encompassing the functionality of various third-party systems in an atomic and stateless fashion, and also well-defined interfaces for communication and composition. The goals that can hopefully be achieved with a SOA include code reusability, cost and complexity reduction, as well as high flexibility. However, SOAs cannot be considered a general success yet, since they are somehow stuck in various loopholes. In this paper we look at a specialization of SOAs, *Web-oriented Architectures* (WOAs), and argue that an appropriate incorporation of Web 2.0 technologies can make it easier to achieve what SOAs have promised.

Until today there is no generally accepted method for modeling a SOA or for converting other concepts (e.g., monolithic architectures, CORBA landscapes) into a SOA. Rather, there exist several procedural methods, including ones from IBM and SAP, which are used in practice. There are numerous standards available in the SOA context (including Web Services for communication, Universal Description, Discovery and Integration or UDDI as a repository standard, and SOAP as a message format), but instead of easing the creation of a SOA, they typically make their realization more complicated [14]. Indeed, this large number of standards reflects an "over-standardization" which makes SOAs difficult and complex to implement. Web Services alone are defined by over distinct 70 specifications that cover numerous individual aspects, yet some of them (like UDDI) are barely used [8]. The implicit overhead of a SOA infrastructure is reminiscent of the Common Object Request Broker Architecture (CORBA), which had problems from its very start due to the fact that it was too complex to manage and proper implementation was often left to the application builder.

On the other hand, recent developments in the context of what is commonly termed "Web 2.0" show how easy it can be to link or compose ("mesh") IT components dynamically, so that original SOA goals such as flexibility or reduction of complexity can indeed be achieved by relatively simple means. Examples include mashups based on GoogleMaps (like www.housingmaps.com) or applications like Yahoo!Pipes (pipes.yahoo.com); these are based on Web Application Interfaces (Web APIs) which allow using the functionality of a Web application by a simple (REST-

978-0-7695-3473-2/08 $25.00 © 2008 IEEE
DOI 10.1109/APSCC.2008.140

based) service layer. An interesting concept in this context is the *Web-oriented architecture* (WOA), which represents a specialization of SOAs obtained by using simple Web 2.0 technologies and standards. Its important aspect is the fact that no additional standards have been defined, but existing ones such as HTTP, SSL, or XML are employed.

Since for a WOA no generally accepted modeling techniques are available either, this paper reviews several SOA models in light of their applicability to WOAs. To give the reader a feeling of where exactly the differences between the two approaches lie, a fictitious case study is introduced. This case, which is introduced next, is used to discuss a possible technological realization and to preliminarily investigate the question of the intended simplicity can indeed be achieved.

An enterprise that has so far only run stationary shops plans to extend its sales operation beyond regional borders. To this end, it wants to move its operation to the Web, but in such a way that available legacy systems are incorporated into the new IT landscape; moreover, core business processes need to be captured and realized, where some parts can be processed automatically, others only semi-automatically.

The enterprise's core business is the imprinting of textiles and other merchandise with user-defined templates. Customers are offered a wide variety of imprintable articles, which can be decorated with writings or graphics. Orders handled can range from small (e.g., for individuals or clubs) to large (e.g., for companies). Up to now the enterprise only runs an ERP system for warehouse and product management. Lithographs are developed together with the customer within a vector graphics application. Moreover, the enterprise runs several machines for printing as well as for flock coating, which are controlled by a central server. The server software is able to accept print commands via a Web service call over the company's Intranet. Since machine capacity is not fully used but the company wants to expand, the business model is to be extended onto the Internet; it shall consequently be supported by a new IT system.

The target system is expected to process customers' product designs and orders entirely over the Web, so that the customer base can be enlarged considerably. After a transition period, it is even planned to abandon the stationary business. Thus, the goal is to implement and run a Web shop, and the existing ERP system and the available machine control systems are to be enhanced by the following functionality:

- Online payment system,
- data mining on the Web server's log data (e.g., for shopping basket analysis or customer profiling),
- long-term storage of business transaction data in a customer-relationship management (CRM) system,
- use of a data warehouse for analyzing all business-related data.

Several suppliers offer Web interfaces, in order to make order processing and subsequently warehouse management more flexible; this shall be integrated into the overall architecture as well. Figure 1 summarizes the intended functionality.

The remainder of this paper is organized as follows: In Section 2 we present a sample SOA solution for our case study. In Section 3 we then summarize the essential Web 2.0 features and their impact on SOAs. This motivates Section 4, where the case study is now realized as a WOA; this results in a number of differences to the previous solution. Finally, in Section 5 we discuss open problems and future work.

2. A SOA-Based Solution

In this section we present a sample SOA solution for our case study which essentially follows the IBM approach. We do so since it is our experience that this approach is feasible and distinguishes the main steps of SOA design in a satisfactory way.

2.1. Basics

The general aim of a SOA is the creation and provisioning of interfaces to (sub-) processes (services) of available (internal and possibly external) IT systems and the mapping of an enterprise's business processes to these services. The description of the interfaces will typically be registered with a directory service which manages an enterprise-wide repository. This facilitates the finding and reuse of services. The advantages which SOA is supposed to have affect the reusability as well as the maintainability of such services; especially the property of reusability, if realized appropriately, can lead to a considerable cost reduction for software development. Moreover, a great emphasis is placed within a SOA on the flexibility to be able to react dynamically to changes and modifications of business processes, for example by changing individual steps of a given workflow (cf. [9]).

In the center of the SOA concept are services; these are used by three protagonists in a "classic" SOA approach: the *service provider*, who offers and ultimately provide a service; the *service user*, who searches for services needed and eventually finds a suitable provider; and the *service broker* who potentially knows about available services and can establish contact between a provider and a user or client.

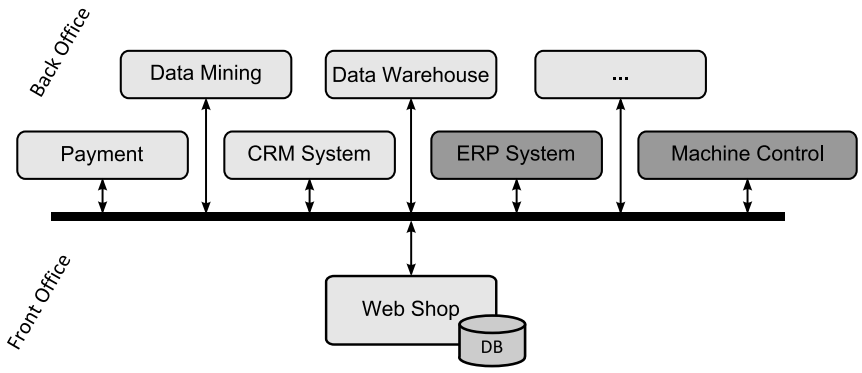

Figure 1. Case study systems and components.

Moreover, a SOA can be divided into different layers of abstraction, in order to be able to argue about it from a conceptual point of view. The four basic layers are: the *application layer*, which includes existing legacy systems, the *service layer*, which provides services based on the application layer, the *process layer*, in which services are orchestrated, where "orchestrating" means the assembly of services to build a process, and finally, the *presentation layer*, which makes the functionalities of SOA available to users within (Web) portals or desktop applications (cf. [5]). Through these layers service-oriented architectures abstract from a precise, technological conversion; however, a major portion of the literature links SOA directly to Web Services and XML (cf. [4], [16]). The use of a service repository is also often considered in connection with the UDDI standard.

A widely-used procedural model for the design of SOA is due to IBM and describes the three phases *Component Business Modeling*, *Service-oriented Modeling and Architecture* and *SOA Realization* (cf. [1], [7]), which is presented in Figure 2 in a simplified version.

2.2. Component Business Modeling (CBM)

In the first step of component business modeling, business processes are analyzed by the enterprise (or their SOA designer) and assembled as components (*Business Components*). The description of a component includes the purpose of business, the component's activity, resources needed, control of the component, and the use of other components (interfaces). Finally, all components are arranged in a tabular overview along the two dimensions of *Accountability* and *Business Competency* (cf. [12]). Presumably, a CBM analysis for the enterprise of our case study has already been made and is readily available.

2.3. Service-Oriented Modeling and Architecture (SOMA)

In the second step of this procedure model, service-oriented modeling and architecture, the IT services provided by existing systems are analyzed and, if possible, the components are attributed to the results of the previous CBM analysis. SOMA can be understood as a mixture of top-down and bottom-up procedural models; it considers the present business processes as well as the already existing business IT. In addition to these four horizontal layers, two vertical layers are designed: the *integration layer*, which connects all horizontal layers, and the *quality management layer*, comprising security and other control functionalities (often summarized under the term *SOA governance*).

Since only a few of the favored IT systems are supposed to be retained in our case study, it is necessary to explain which systems have to be reinserted in order to achieve the intended objectives. First, in the **application layer** legacy system "machine control" is tied to the "ERP system." As assumed earlier, the communication with the machine server and the already existing ERP application can be realized via Web Services. A database management system is added as a new component for the data warehouse (in the most elementary case, an easily manageable system such as MySQL or PostgreSQL will suffice). The CRM system is also reinserted. The presentation layer is responsible for displaying the functionalities of the ERP and CRM components, which is supposed to be realized via portlets within a *Java Application Server*.

Following common practice, Web Services whose interfaces are described by means of the *Web Service Description Language* (WSDL) are added within the **service layer**. For a better overview, the interfaces of the systems of the application layer (ERP, CRM, DWH and machine control) which are to be inserted should be registered in a UDDI directory. Typical examples for this layer are the exemplary

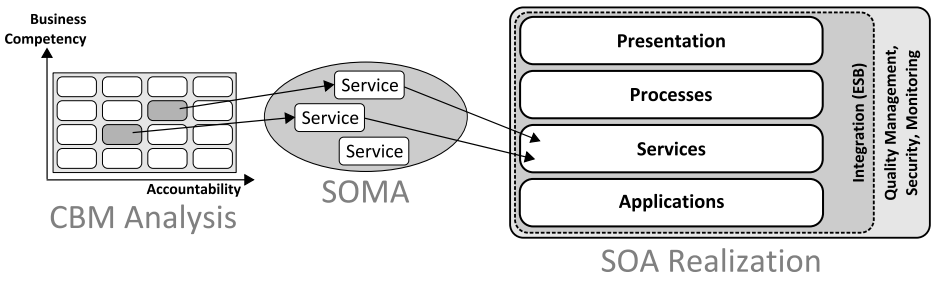

Figure 2. IBM SOA design methodology.

services "save order" and "send job to machine control." In order to insert business processes into a SOA, these are normally filed as *Business Process Execution Language* processes (BPEL) and then executed within the process layer. These processes in turn revert to Web Services, which are described in the service layer by WSDL files. Here, a BPEL process has to be filed for every IT process identified in the previous CBM analysis. In order to control such a process, a workflow engine is needed (e.g., *Apache Ode* within a *Tomcat Application Server*). Sample processes at this layer are "order item," "order processing" and "incoming goods."

Finally, the functionalities of the subjacent layers are made accessible for users at the **presentation layer**. Here, the web shop is accessible for clients of the enterprise, which communicates with the other systems by means of processes (e.g., "order item") or via direct requests for service. For example, the functionalities of the ERP system and, where applicable, also the functionalities of the CRM system are integrated into a portal through corresponding portlets. For that, an additional portal server (e.g., *JBoss Portal Server*) is needed.

2.4. A SOA Realization

In the final step, the following tasks have to be carried out: precise programming for the implementation of services, setting up of the *Application Server*, integration of an *Enterprise Services Bus*, and the realization of the governance layer. In this connection, all four horizontal layers have to be taken into account and connected to each other with the aid of the **integration layer**. For this purpose the concept of the *Enterprise Services Bus* (ESB) has been recommended for a while now. An ESB is a classical bus system, in which connectors for all transport protocols and message structures used in a SOA are created, and which is then used in order to impart and guard messages between the systems and the components of the layers. Therefore, no point-to-point connection between the services is needed, because all requests for service operate via the ESB (cf. [3]). Depending on the product, workflow engines, which carry out and guard BPEL processes (of the process layer), are integrated into the ESB. Figure 3 shows a sample arrangement of systems used for the SOA solution of a our case study.

3. Web 2.0 Characteristics in a Nutshell

Web 2.0, which has been frequently and intensively discussed in recent years, can be characterized by three core aspects: functionality, data, and socialization (cf. [15]). Generally, what we are experiencing in this context is a transition from a read-only Web to a read-write Web, where users no longer just consume information, but also contribute at an increasing rate. One of the most important features of the Web 2.0 movement is its simplicity and the categorical principle to convert only what is unconditionally necessary. This refers not only to the operation of an application or of its interfaces, but also to its programming. Besides tagging, communities and social networks, blogs, wikis and Ajax-based interfaces, *Rich Internet Applications* (RIAs) as well as *mash-ups* play an important role. Mash-ups are applications which are assembled from several other applications or services and which then open up new usage or service possibilities. The most popular type of mash-up currently is the *mapping mash-up*, in which data such as weather or temperature data is overlayed over a map (e.g., one obtained from Google Maps) so that location information is added to the raw data in order to make it more expressive; numerous examples of such mash-ups in a variety of categories can be found at `www.programmableweb.com`. Open Web application programming interfaces (Web APIs) of the involved applications and the data sources often establish the basis for a mash-up, which is hence easy to create. A standard which has established itself for requests of methods in this context is *Representational State and Transfer* (REST). REST is thoroughly described by [6] and views a *Uniform Resource Locator* (URL) on the internet as a resource, so that objects can easily be created, processed, and deleted by means of standard HTTP methods (GET, POST, PUT, and DELETE). Many new Web interfaces (e.g., Google, Yahoo!, Amazon Web Services) provide services which offer equivalent REST-services in addition to their Web Service inter-

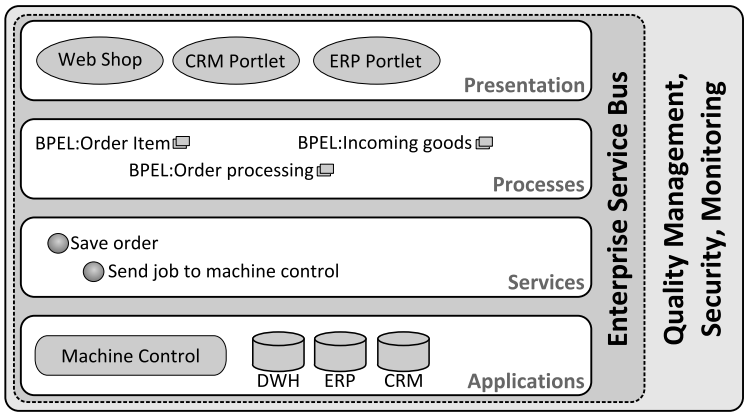

Figure 3. Sample SOA solution.

faces. Notice, however, that for none of these services entries in a UDDI directory exist; indeed, in order to be able to employ such a service, an application designer has to know where to find them, and subsequently has to produce and deploy the appropriate API calls. As a result, a considerable simplification over the original SOA concept is achieved [11], which will be used in the next section to design an alternative solution for our case study.

Web 2.0 developments have also enabled another development: Complete software packages which operate via the Internet without the need for local installation or license acquisition and which require a Web browser only (or can be executed on the desktop), enjoy a rapidly increasing popularity. Application of this type are called *Software as a Service* (SaaS) and can substitute locally installed software packages in many cases, since the spectrum ranges from document processing and spread sheets to data bases. *Platform as a Service* (PaaS) applications even go a step further; here, single software services can be combined into new ones in order to make them directly accessible for other users on the very platform of creation.

As already indicated, the increasing diffusion rate of Web APIs and HTTP interfaces, which is not only limited to Web 2.0 applications, more and more influences the environment and positioning of a SOA and opens up new possibilities for connecting services which reach far beyond the boundaries of the individual enterprise. Because of the focus on Web Services this form of a SOA is called a *Web-oriented Architecture* (WOA). In contrast to a SOA, a WOA is clearly directed towards Web technologies, such as HTTP for the transport protocol, or Web APIs and REST for the transfer of services. Thus, a WOA can be be seen as a subset or specialization of a SOA; Figure 4 indicates this. However, the question arises whether existing procedural development methods and models should be applied to a WOA off-hand; the following sections tries to give an initial answer to this question.

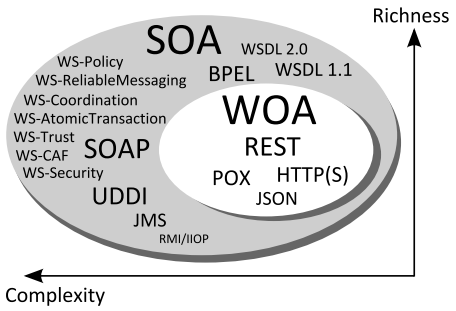

Figure 4. WOA as SOA subset (Source: www.hinchcliffe.com).

4. A WOA-Based Solution

Unlike a SOA, in which layers can be separated from each other pretty clearly, those in a WOA are less selective and less separated by a precise statement of concern; nevertheless, it is possible to build on the arrangement of SOA layers discussed and shown earlier (cf. Figure 2); this is discussed next. At the presentation layer, a browser is all that is needed and is still responsible for dealing with all presentation issues. For example, *Rich Internet Applications* (RIA) can be attributed to this layer since one of their primary tasks is to present user interfaces to Web-based applications. The process layer also remains in position and effect, only the execution of processes is completely or partially transferred to the Internet and is done there by appropriate tools (i.e., Web services, Web APIs, SaaS or PaaS applications). The service layer is divided into two sublay-

ers (similar to the original IBM model, see [2]): the *logical services* and the *service implementation*. Logical services are, for example, mash-up descriptions which encapsulate a set of services and which specify the intended flow of data from one service to the next or to the mash-up result. This perception is supported by an emerging generation of tools such as *Yahoo!Pipes* (see pipes.yahoo.com), since they allow a specification of the favored mash-up at a logical level; in essence, these tools resemble a graphical form of programming, since the intended workflow implemented by the mash-up is specified graphically. Service implementation then needs to specify through which service call a previously specified mash-up is actually realized; in the case of Yahoo!Pipes, for example, this is done by "enacting" a given mash-up specification.

While mash-ups can be attributed to any of the three upper layers, Web Procedure Calls (WPCs) [15] that call upon services exclusively available through Web APIs can be arranged as services in a WOA at the service implementation layer as well as the infrastructure layer; this is summarized in Figure 5. Communication within a WOA is favorably realized by REST services and, accordingly, by simple HTTP requests; these are usually called Web APIs. Requests for Web APIs can be used within RIAs, mash-ups, and WPCs; therefore they are connected with each of the presented parts.

4.1. Planning and Selection of Web Services

The first steps in planning a WOA are the same as planning a SOA solution. Similarly to the solution above, we presume that a CBM analysis for the enterprise underlying this case study has been made and is available to the WOA designer. Next a step similar to SOMA follows: services provided by service providers are searched for which can be mapped to internal processes of the enterprise. Thereby additional benefit can be generated by mashing up information (e.g., information overlay within a map of a mapping service).

Until today there are no machine-readable directories for Web APIs, hence one has to search for appropriate services, APIs, or existing mash-ups on an individual basis (e.g., at `www.programmableweb.com/`). The IT infrastructure and the available services should be designed and chosen as much cost- and resource-saving as possible; therefore, license- and cost-free services are favored over commercial ones. So the cost of used services should be apparent at the end of the design phase (e.g., fees for the service according to CPU time or bandwidth used) to allow for an assignment of costs to individual business transactions.

The following services are applicable for a WOA solution in our case study:

- *Amazon Flexible Payment Service* (FPS) as payment system for the Web shop,
- *Google Analytics* for usage with data mining methods applied to the log data of the shop server,
- *Salesforce.com* as CRM system,
- *DabbleDB* for analyzing business data (data warehousing in a broad sense)
- and *BungeeConnect* as integration platform to connect all services mentioned above with each other and with the Web shop, the ERP system, and with machine control.

Figure 6 shows the selected services as we propose to integrate them into a WOA, together with a mapping to the layers discussed above.

4.2. Web Service Integration

The central point in our WOA specification is a component which we call the *Integrator*. All IT systems and services are linked to this component, and we even envision that existing workflows resulting from business process specifications are managed from here. Finally, the Integrator could be put in charge of what is presented to the user (via a browser). Thus, this component is roughly comparable to an ESB in a SOA, which also merges and manages services and systems. Here we exemplarily choose the PaaS provider *BungeeConnect* (`www.bungeelabs.com/`), which allows the programming of Web interfaces, integration of Web APIs and Web services, among other things, at a reasonable costing scheme.

The payment process for orders, which are placed by users through the Web shop, is redirected to *Amazon Flexible Payment Service*. This makes it possible for customers to pay, using their Amazon account, via credit card or bank transfer, and no local payment routine has to be implemented. To collect as much data (and mine as much information) as possible from user behavior within the Web shop (e.g., click-path analysis) *Google Analytics* is brought in. The large *Customer Relationship Management* (CRM) system *Salesforce* is used for managing customer data and for launching sales campains. Finally, the database service *DabbbleDB* is used for analyzing business as well as transaction data and for building a data warehouse for consolidation of financial figures.

Data exchange between single applications is mostly done through the Integrator. Thus, sales data from the Web shop is redirected to *Amazon FPS*, then returns to the *Integrator*, and finally feeds directly into the CRM and the ERP systems.

In contrast to the SOA solution described earlier, the technologies needed for a WOA solution are just using existing standards such as HTTP, SSL, and REST services.

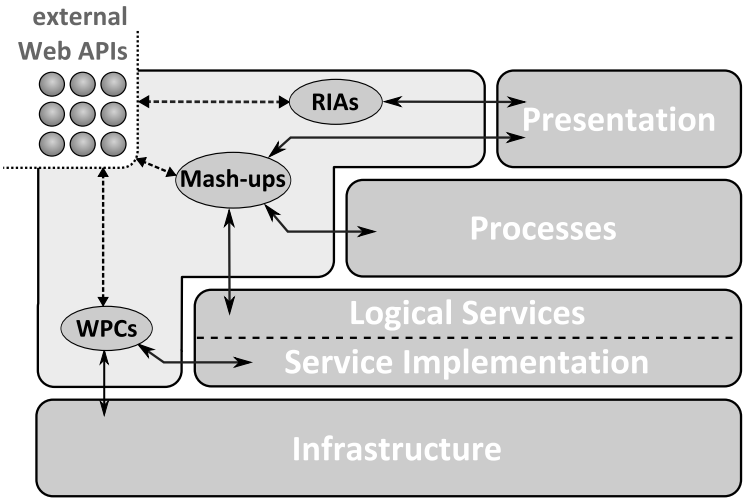

Figure 5. Simplified layered view of a WOA.

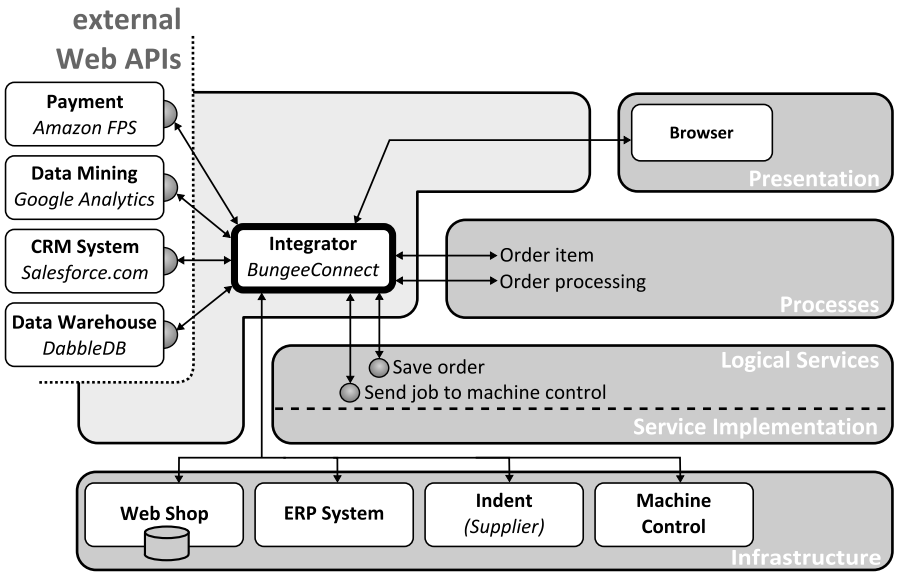

Figure 6. Case study using WOA solution.

The architecture of a WOA is reminiscent of mash-ups but differs in the complexity of the services used and their connections. While a mash-up is typically built upon just a few existing services (often just two or three) to gain additional benefit, a WOA connects single services, Web APIs, Web services, and in some cases also local IT components; thus, it is typically more complex than a mash-up. The programming and (later) the managing of the entire setup is done here with *BungeeConnect*, so that no decision about technology usage is needed, again in contrast to a SOA solution.

It should be noted that from an enterprise's point of view one of the main criticisms of the WOA concept is transport and data security. Particularly for critical or confidential business data, which is here computed and saved by *Google Analytics* or *DabbleDB*, proper *general terms and conditions* and *service level agreements* (SLAs) are essential for a service. A good example is again Amazon Web Services where SLAs are already offered for the provided services. Since this aspect is still in an early stage of development, we do not discuss it any further here.

5. Open Problems and Future Work

In this paper we have tried to contrast SOA development with more recent WOA development, and we have done so by way of a case study. The motivation for this study derives from the observation that SOAs have not been as successful to date as many people have expected when they were first announced. We see reasons for this in the fact that there is an overkill of standards around the notion of a SOA, there is a lack of open and commonly accessible service directories, the return-on-investment (ROI) is not always easy to determine, and SOA design remains a complex task, especially when performed within the context of an enterprise with an IT infrastructure that has emerged over many years. We are convinced that WOAs can provide a solution to this dilemma, we have tried to exemplify this through our case study.

As a preliminary result we note that beside the Web shop the entire functionality needed in our case can be obtained from Web 2.0 applications and services and can be integrated into a WOA in a straightforward manner. From an implementation perspective, this even appears to be easier and more flexible than with existing SOA approaches. Moreover, from an economical point of view, a WOA does not require an acquisition of new hardware or software (so that no financial investments are at stake), but instead the services employed all come with clear price tags (most often referring to CPU, storage, or bandwidth usage) so that ROI calculations become much easier than before. On the other hand, it must be mentioned that many of the services we have referred to in this paper are in a beta only, so that their sustainability cannot be guaranteed in each case at this point.

Within a SOA processes are often executed via a BPEL workflow engine; such a control mechanisms is yet to be established within a WOA context. Initial proposals have already been made in this direction, such as the extension of BPEL to RestfulBPEL [10]. Another issue is related to spotting appropriate APIs and services on the Web; with an ever increasing number of offerings the quest for a repository or directory in which these can be found is coming back. To this end, the *Web Application Description Language* (WADL) can be seen as a kind of "successor" to WSDL for describing services that even enables machine-to-machine communication. The site *ProgrammableWeb* mentioned earlier can be seen as such a directory, which, however, still needs to be searched manually.

In [13] we describe a general development methodology for WOAs that abstracts from the case study we have presented here. For properly applying this methodology to real-world situations, it will be important to identify application characteristics that can tell whether a SOA or a WOA is indeed appropriate; these could be logical or technological ones (such as two-way communication which is more difficult to realize with a WOA approach), and they need to be used as "yardstick" that can ultimately measure the quality of the resulting system.

References

[1] A. Arsanjani and A. Allam. Service-oriented modeling and architecture for realization of an soa. In *2006 IEEE International Conference on Services Computing (SCC 2006), 18-22 September 2006, Chicago, Illinois, USA*, page 512. IEEE Computer Society, 2006.

[2] A. Arsanjani, L.-J. Zhang, M. Ellis, A. Allam, and K. Channabasavaiah. S3: A service–oriented reference architecture. *IT Professional*, 9(3):10–17, 2007.

[3] D. A. Chappell. *Enterprise Service Bus*. Theory in practice. O'Reilly, Beijing, 1. ed edition, 2004.

[4] T. Erl. *Service–oriented architecture: A field guide to integrating XML and Web services*. Prentice Hall PTR, Upper Saddle River, NJ, 9. printing edition, 2004.

[5] T. Erl. *SOA: Principles of service design*. Prentice–Hall service–oriented computing series. Prentice Hall, Upper Saddle River, NJ, 2008.

[6] R. T. Fielding. *Architectural Styles and the Design of Network-based Software Architectures: PhDThesis*. PhD thesis, University of California, Irvine, 2000.

[7] J. Ganci. *Patterns: SOA foundation service creation scenario*. IBM Redbooks. International Technical Support Organization, Poughkeepsie NY, 1st ed. edition, 2006.

[8] S. Hagemann, C. Letz, and G. Vossen. Web service discovery – reality check 2.0. In *Proc. 3rd International Conference on Next Generation Web Services Practices (NWeSP), Seoul, Korea*, pages 113–118. 2007.

[9] Y. V. Natis. Service–oriented architecture scenario, 2003.

[10] H. Overdick. Towards resource–oriented bpel. In *2nd Workshop on Emerging Web Services Technology*. Aachen, 2007.

[11] C. Pautasso, O. Zimmermann, and F. Leymann. Restful web services vs. "big"' web services: making the right architectural decision. In J. Huai, R. Chen, H.-W. Hon, Y. Liu, W.-Y. Ma, A. Tomkins, and X. Zhang, editors, *WWW*, pages 805–814. ACM, 2008.

[12] G. Pohle, P. Korsten, and S. Ramamurthy. Component business models: Making specialization real, 2005.

[13] G. Thies and G. Vossen. Modelling web-oriented architectures. *submitted for publication*, 2008.

[14] G. Vossen. Have service–oriented architectures taken a wrong turn already? In *IFIP TC 8 International Conference on Research and Practical Issues of Enterprise Information Systems (CONFENIS 2006), Vienna, Austria*, pages xxiii–xxix. 2006.

[15] G. Vossen and S. Hagemann. *Unleashing Web 2.0: From concepts to creativity*. Elsevier/Morgan Kaufmann, Amsterdam, 2007.

[16] T. Zhang, S. Ying, S. Cao, and X. Jia. A modeling framework for service-oriented architecture. In *QSIC*, pages 219–226. IEEE Computer Society, 2006.