

Web Services as Product Experience Augmenters and the Implications for Requirements Engineering: A Position Paper*

Pascal van Eck *Roel Wieringa*

Centre for Telematics and Information Technology, University of Twente
P.O. Box 217, 7500 AE Enschede, The Netherlands
Email: {vaneck,roelw}@cs.utwente.nl

Abstract

There is currently little insight into what requirement engineering for web services is and in which context it will be carried out. In this position paper, we investigate requirements engineering for a special kind of web services, namely web services that are used to augment the perceived value of a primary service or product that is itself not a web service. We relate requirements engineering to a common enterprise architecture pattern and derive from this a number of research questions for further study.

1 Introduction

There is currently an abundance of research in technical aspects of web services and the semantic web, but at the same time insight into how web services and the semantic web will be used is minimal. In this paper, we start from the assumption that most web services will not be independent economic offerings that customers are willing to pay for, but are mainly used to support the sales of other products or services. We call this kind of web services *product experience augmenters*. Given this assumption, we can derive a number of implications with respect to how web services have to be designed and deployed in organizations.

Most organizations that will employ web services in the future already exist today. It is therefore interesting to study the architecture of information processing already in place at those organizations and derive from this implications for requirements engineering, especially for *semantic* web services. In this paper, we do

*This research was partially sponsored by the Telematics Institute. See <http://www.telin.nl/NetworkedBusiness/Graal/ENindex.htm>.

so using a common enterprise architecture pattern found in the GRAAL project¹. This results in a number of research questions that need to be answered to get a better insight into what requirements engineering for web services amounts to, how it will be carried out, and by whom.

In this paper, unless otherwise indicated, we use the term *web service* as a blanket term for e-services (i.e., offering services via the Internet, as opposed to e-retailing), web services in the sense of computational elements that are described by WSDL and accessible via SOAP, and semantic web services (i.e., web services described by ontologies).

2 Web services as product experience augmenters

The most important assumption underlying this work is that *most web services will augment existing products or services* (not necessarily information products), rather than constitute an independent economic offering. We take as an example the (ubiquitous) travel domain. There are already a number of airlines and travel agencies that offer their clients updates on departure data of their flight via text messages (SMS) delivered to their mobile phones for free. This service is not in itself an economic offering, but rather extends the perceived service quality of the primary service (transportation). Currently, these kinds of extra services are offered via text messages, WAP, or a normal website. We expect that in the future, more and more of these services will take the form of WSDL-described and SOAP-accessible web services, which will be made available to customers via portals that take care of, among other things, personalization.

From this assumption, we derive the following characteristics for web services:

- Web services are part of the primary product or service offering and therefore, functional maintenance of web services is a responsibility of the Marketing & Sales department or the departments that offer the primary product or service.
- Web services have to fit in the business processes and supporting information systems already deployed by these departments.
- There will be a lot of web services, many of them will be short-lived and need to be very cheap as customers are not paying for them independently from the primary product or service.
- Web services are part of the brand identity of the organization that offers them; customers want this because it increases trust, providers want this because it offers a possibility to differentiate from competitors.

A web service is itself not a monolithic thing, but will be composed of other services. We distinguish four service tiers, as depicted in Fig. 1. End customers use

¹ See <http://is.cs.utwente.nl/GRAAL>

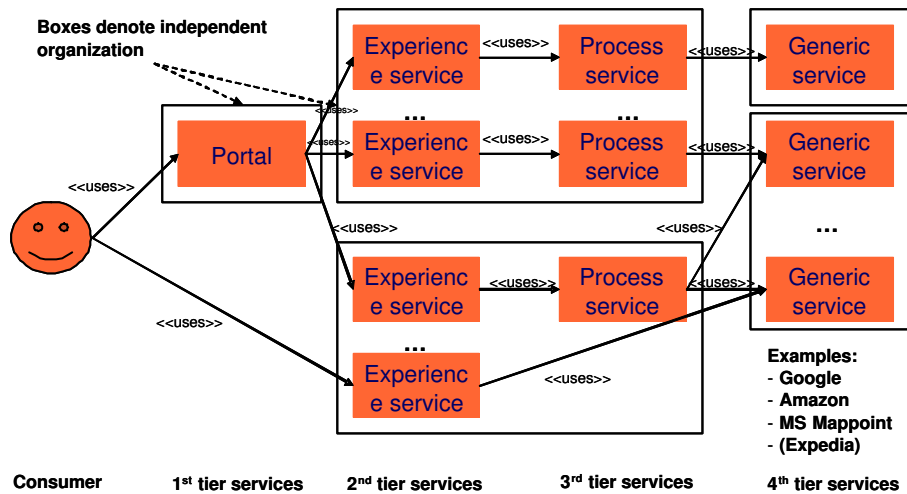


Fig. 1: A four-tier service architecture.

web services via portals which are typically offered by providers of access network services as a value-added service. These portals provide customization, personalization and context-awareness. Via these portals, customers access second-tier services, which are the services associated with primary products (e.g., the flight data update service in our example). Inside the service provider organization, product experience augments services are themselves in most cases composed of basic services in which functionality common to all products is factored out (these are the third-tier services). The third-tier services in turn will use services that provide generic data such as telephone directory information, web search, etc. We expect that there will be only relatively few fourth-tier services as the generality of their data results in natural monopolies.

We think that the difference between web services directed at end consumers (B2C) and at businesses (B2B) is not very important. All the above points hold in both cases. We expect that B2C services will be developed in a similar way as mass-marketed end-consumer products are developed, e.g. using focus groups. We expect that B2B services in e.g. long-term relations between a supplier and its customer will be co-developed by the business partners (or even an entire business sector) on the basis of a negotiation process.

3 A common enterprise architecture pattern

In the GRAAL project, we study architecture documentation provided by our business partners to find patterns in enterprise architecture. So far, we have focused on large transaction processing organizations in the financial service industry (banking, insurance), and in government. Fig. 2 shows a pattern that we have found in most of them. This figure shows a typical application architecture at the enterprise level, which is at a higher level of granularity than in software architecture [1, 3, 5]. In the

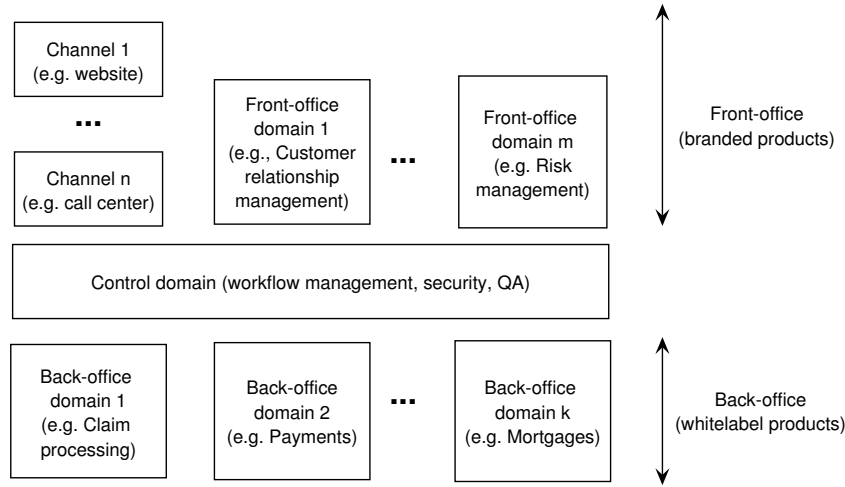


Fig. 2: Typical application architecture in large financial service provider.

GRAAL project, we identify a number of architecture layers [8]. In the current paper, we only present the architecture at the application layer (i.e, Fig. 2 depicts the structure of the software applications employed by a typical service provider). Applications support business functions and business processes at the business process layer. In turn, applications themselves are supported by an infrastructure consisting of hardware, operating systems, database management systems, etc. at the infrastructure layer.

Fig. 2 shows a number of domains, which are sets of information systems or applications. Domains are identified differently by each organization that employs this architecture, but in most cases the division is based on generally recognized areas of expertise in the organization. Thus, each domain is a knowledge domain. There are four kinds of domains:

Front-office domains Front-office domains are responsible for customer contacts and producing products and services that are sold to groups of customers. Each of these products or services carries one of the brands of the organization.

Back-office domains Back-office domains are responsible for business processes needed to produce the products or services offered by the front-office domains. These business processes are not specific for a particular group of customers and are not branded.

Channels Each channel domain represents a customer contact channel: a website, a call center, branch offices, etc. Customer interactions are decoupled here from the channel and processed in a channel-independent way by the front-office domains.

The control domain The control domain manages workflows between domains, both between front-office domains and back-office domains as well as between one front-office domain and another and similar for the back-office.

Perpendicular to the business process, application and infrastructure layers is the structure of IT management within the organization. Looijen [4] distinguishes three kinds of IT management: functional management (end-user support and defining functional specifications, usually carried out not by IT personnel but by domain experts), application management (corrective and adaptive maintenance of software applications carried out by software engineers) and infrastructure management (responsible for availability of the infrastructure, carried out by systems administrators and network engineers). One of the business partners in the GRAAL project uses the domains as depicted in Fig. 2 to structure functional management: for each domain, a manager is appointed who is responsible for functional management of the applications in his or her domain.

As an aside, the architecture depicted in Fig. 2 can itself be viewed as a service-oriented architecture in which one domain provides services to another domain. In fact, one of the business partners in the GRAAL project is already taking some first steps in making this a reality. However, for the current paper, this is not of importance.

4 Embedding web services in the enterprise architecture: research questions

Given the characteristics of web services presented in Section 2 and the architecture pattern presented in Section 3, in this section we discuss how support for web services can be ‘embedded’ in an organization. Currently, the embedding presented in this section has not been validated in any way and therefore has the status of a research hypothesis.

In the front-office and back-office domains depicted in Fig. 2, four kinds of processes can be distinguished:

1. The processes needed to produce and/or deliver the primary products or services for which the domain is responsible. These are existing processes, possibly supported by a workflow management system. If they are described formally, this is usually in a human-readable way in for instance quality manuals and procedure manuals.
2. The processes associated with the web services that augment these primary products or services. These processes are usually described using BPEL, DAML-S, or similar.
3. The processes needed to define the previous two kinds of services: interaction with focus groups in B2C e-business, negotiation with important customers in

B2B e-business, definition and management of service level agreements with customers.

4. The processes needed to implement web services: interaction with application and infrastructure management and with providers of services at a higher-numbered tier.

We state that requirements engineering for web services is that part of the third kind of processes in which processes of the second kind are designed. The web service processes that are designed are related to the first kind of processes: web services provide input to these processes and/or use results of these processes.

The design of the kind of processes that in our vision, requirements engineering is responsible for consists of *defining a service* in terms of its input and output, and *designing the workflow* (described in for instance BPEL) needed to deliver the service. At a high level, defining services consists of composing a dictionary that describes the meaning of messages exchanged between the service consumer and provider. As twenty years of experience in EDI have shown, this is very difficult.

As stated in Section 3, the domains in Fig. 2 are knowledge domains, and they can be the basis for functional management of the applications in each domain. It therefore seems natural to assign responsibility for each web service an organization offers to one (and only one) domain. In this case, each web service is assigned to the domain that is already responsible for the part of the primary product or service supported by the web service. In general, second-tier services become the responsibility of front-office domains, while third-tier services become the responsibility of the back-office domains.

As a consequence, the semantic definition of services takes place in the context of a relevant knowledge domain. Part of requirements engineering for semantic web services is therefore establishing a mapping from messages exchanged with a service consumer to the mechanism used to deliver the service. This mechanism consists of interactions with processes that produce the primary product or service and interactions with services delivered by a higher-numbered service tier. The challenge for the requirements engineer is to construct this mapping such that it is isomorphic to the (implicit) mapping the service consumer uses to give meaning to messages exchanged with the service provider. If this mapping is not isomorphic, the service consumer will perceive this as bad service quality. This is comparable to the GAPS model for service quality used in the area of service marketing [9].

5 Conclusion

Starting from the assumption that most web services will augment primary products or services that are not themselves web services, instead of being independent offerings that people pay for, we have identified some implications for the design of such services in the context of enterprise architecture. These implications have

so far not been validated in any way; they therefore have the status of research questions. Apart from these research questions, which relate to what requirements engineering for web services is, there are also interesting research questions with respect to methods and tools needed for this kind of requirements engineering. As explained in Section 4, web services are a responsibility of functional management, which is usually performed by non-IT personnel. This means that a (very) simple and strictly functional modeling method for web services is needed that models web services in terms of SLAs with customers, SLAs with providers of higher-numbered tiers, and semantic mappings as described in Section 4. At the same time, this modeling method needs to be precise enough that the semantic mapping is preserved when implementation is handed over to application management. It is still unclear if such a modeling method is already available.

The research questions proposed in this paper connect two lines of ongoing research in our group: the GRAAL project mentioned before and value-based e-business process design [2, 6]. The research questions here are presented in a way that is generic for all tiers. In another position paper [7], we elaborate upon the relation between second-tier and third-tier services and how requirements engineering differs for these two kinds. In that paper, we also discuss how service blueprinting, a diagramming technique from the area of service marketing, can be used in requirements engineering for web services.

References

- [1] L. Bass, P. Clements, R. Kazman, and K. Bass. *Software Architecture in Practice*. SEI Series in Software Engineering. Addison-Wesley, 1998. ISBN: 0-2011-9930-0.
- [2] J. Gordijn and R. Wieringa. A value-oriented approach to e-business process design. In *Conference on Advanced Information System Engineering (CAiSE 03)*, volume 2681 of *Lecture Notes in Computer Science*, pages 390–403. Springer, 2003. <http://www.springerlink.com/openurl.asp?genre=article&iissn=0302-9743&volume=2681&spage=390>.
- [3] C. Hofmeister, R. Nord, and D. Soni. *Applied Software Architecture*. Addison-Wesley, 2000. ISBN: 0-2013-2571-3.
- [4] M. Looijen. *Information Systems, management, control and maintenance*. Ten Hagen & Stam, 1998.
- [5] M. Shaw and D. Garlan. *Software Architecture: Perspective on an Emerging Discipline*. Prentice-Hall, 1996. ISBN: 0-1318-2957-2.
- [6] P. van Eck, J. Gordijn, and R. Wieringa. Value-based design of collaboration processes for e-commerce. In S.-T. Yuan and J. Liu, editors, *Proceedings 2004 IEEE International Conference on e-Technology, e-Commerce and e-Service, EEE'04*,

- pages 349–358. IEEE Press, 2004. http://is.cs.utwente.nl/pubs/2004/eck_etal_eee04.pdf.
- [7] P. van Eck and R. Wieringa. Requirements engineering for service-oriented computing: A position paper. In J. Gordijn and M. Janssen, editors, *Proceedings of First International Workshop on e-Services, ICEC'03, Pittsburg, PA, USA*, pages 23–28, 2003. http://is.cs.utwente.nl/pubs/2003/eck_wieringa_icec03ws.pdf.
- [8] R. J. Wieringa, H. M. Blanken, M. M. Fokkinga, and P. W. P. J. Grefen. Aligning application architecture to the business context. In J. Eder and M. Missikoff, editors, *Advanced Information Systems Engineering. 15th International Conference, CAiSE 2003, Proceedings*, volume 2681 of *Lecture Notes in Computer Science*, pages 209–225. Springer-Verlag, 2003. <http://www.springerlink.com/openurl.asp?genre=article&iissn=0302-9743&volume=2681&spage=209>.
- [9] V. Zeithaml and M. J. Bitner. *Services Marketing*. Series in Marketing. McGraw-Hill, 1996. ISBN: 0-0707-8250-4.