An Interdisciplinary Methodology for Building Service-oriented Systems on the Web

Steffen Lamparter
Karlsruhe Service Research Institute
University of Karlsruhe (TH), Karlsruhe, Germany
steffen.lamparter@kit.edu

York Sure
SAP Research
CEC Karlsruhe, Germany
york.sure@sap.com

Abstract

Services can be characterized as activities in which providers and customers co-create value. The need for a tight collaboration between providers and customers is thus an important distinguishing aspect of services compared to products. Traditional software engineering methodologies provide limited support for collaborative processes at runtime and are therefore not directly applicable to service engineering. In particular, the design of coordination mechanisms between service provider and customers as well as the design of a common vocabulary in an interorganizational setting is not adequately addressed. To face these shortcomings, we present an interdisciplinary approach integrating just recently proposed Web service engineering methodologies with market and ontology engineering to one coherent service engineering methodology.

1. Introduction

In recent years, Web services have developed to a mature technology, standardized languages and protocols have become available, and consequently more and more interorganizational, service-based systems have emerged. As building such service-oriented software systems involves a wide range of technical, business and legal aspects, the engineering process is highly complex and requires an interdisciplinary approach not covered by existing software or service engineering methods.

In many aspects, service-oriented architectures are fundamentally different from traditional distributed component- or object-based software systems. Most notably they introduce the roles of *providers and requesters* as first class citizens in the architecture. This is an important step towards truly inter-organizational business processes since it allows us to directly apply concepts and ideas traditionally developed for electronic commerce. Moreover, the concept of *dynamic service binding* is introduced that requires automated discovery, selection and invocation of new services at runtime of the system. In traditional software engineering these have been activities usually accomplished at design time. These fundamental distinctions lead to three types of components required for implementing a service-oriented architecture: (*i*) *Web service technologies* provide a hosting platform and a set of standardized protocols, formats and language specifications that enable interoperation between services hosted on different platforms. (*ii*) Since *Web service markets* as coordination mechanism between service providers and requesters are a cornerstone of a service-oriented architecture, results from the area of market theory have to be considered in the engineering process of a SOA. For example, determining suitable bindings may involve locales for negotiating with and selecting among many potential providers. (*iii*) In order to enable service allocation at runtime, service descriptions have to be specified in a formal, machine-understandable way. The state of the art for formally describing services are *ontologies*. They come with a standardized logical foundation providing well-defined semantics which can be utilized for matching of service descriptions and thereby improving interoperability.

This paper contributes an integrated methodology for designing service-oriented systems by identifying the relations and dependencies between service/software, market and ontology engineering and showing how they can be merged in a single engineering process. The individual methodologies are introduced in 2 and subsequently the interrelations and dependencies between the methodologies are discussed in Section 3. Finally, we conclude with a summery in Section 4.

2. Engineering Methodologies

In this section, we shortly review engineering methodologies required for software and Web services (Section 2.1), markets (Section 2.2) and ontologies (Section 2.3). Subsequently we present how these methodologies can be integrated into one coherent framework.

2.1. Web Service Engineering

First, we address the question of how a Web service and service-oriented architectures as a whole are built. Gen-

erally, there is an enormous amount of literature dealing with engineering software systems ranging from sequential methods such as the influential waterfall model [8] to iterative models which combine top down and bottom up approaches such as the spiral model [2] or the Rational Unified Process (RUP) [3]. However, although (software) service engineering can be seen as a special case of software engineering, when moving to a service-oriented architecture these methodologies have several shortcomings neither addressed by object-oriented analysis and design nor by business process management techniques: (*i*) First, the question of how services can be identified has to be answered which requires business process aspects as well as the enterprise-scale application architecture to be taken into account. Obviously, object-oriented analysis is a good starting point, but it does not address how to discover the functional units of work from a business perspective. (*ii*) Second, a paradigm shift towards explicitly appreciating the key roles found in service-oriented systems is required. These key roles are service provider, service requester and intermediaries. (*iii*) Third, the methodology should reflect the fact that services are not built for one single business line or company, but are potentially exposed to other departments or companies in which they are used in completely new business contexts.

With these ideas in mind, several methodologies have been defined that are explicitly tailored towards Web service engineering. In the following, we look more closely at the service-oriented analysis and design methodology (SOAD) propagated by IBM [9] and the service-oriented design and development methodology proposed in [6]. While the overall process strongly conforms with the standard software engineering process [1] comprising the steps *requirement analysis*, *design*, *construction*, *testing and maintenance*, particularly the design phase has been refined as described in the following:

System Analysis. First, a bottom-up *analysis of the existing system* takes place in order to find resources that could serve as a basis for providing service functionality. System analysis is typically executed by the provider.

Domain Decomposition. The top-down approach starts with the *domain decomposition* step. Here the business process is decomposed into its subprocesses using high-level business use cases that enable identifying the functionality required as service.

Component Design. In the next step, viable *components* that implement the application logic required for a service are *identified*. In the *component specification*, the messaging and event specification, the internal flow and structure of the identified components, and other component dependencies are described. Missing components have to be custom built in the *component realization* step. Defining the component that implements a certain service is called *component allocation*. In Figure 1, the component identification, specification, realization and allocation is captured by the term *component design*.

Service Design. Similarly, *service design* can be broken down to service identification, specification and realization. The *service identification* step deals with deciding which operations of the component should be accessible via a Web service. After identification of the services, their characteristics have to be documented to enable their implementation and later reuse.(*service specification*). Finally, new services are created in the *service realization* step.

Service Binding. Once services are realized and exposed by the provider on a service market (*offer specification*), requesters can integrate them into their business process. Therefore, they have to decide in a top-down manner by means of domain decomposition, which services are required and which task of the process should be done by which service (*request specification*). Thus, the term binding captures the assignment of service requests to offers. These bindings can be specified explicitly by the developer or determined dynamically by the system using the request and offer specification. In either case, the mechanisms bringing together service demand and supply have to be carefully designed, which is the main purpose of the field market engineering discussed in Section 2.2.

Evaluation. Finally, the constructed architecture is *evaluated* in terms of functionality, robustness, efficiency, etc. While these step mainly corresponds to the traditional software engineering methods, some additional criteria are important, such as reusability of services, efficiency of provider selection, etc. The latter is addressed in the next section.

2.2. Market Engineering

An electronic market for coordinating service requests and offers has to meet certain requirements, such as welfare optimization or maximizing the transaction volume in the market. As meeting these requirements can be very complex, the engineering process is broken down into less complex sub-phases. Although structured according to similar phases as software engineering, market engineering focuses on different aspects and goals. In this section, we briefly introduce the different phases. A more fine-grained process accompanied with a detailed discussion for each phase can be found in [5].

Environmental Analysis. The *environmental analysis* deals with gathering information about the concrete setting for which the market is designed, including information about the participants, about the products to be traded, about possible intermediaries, etc. The phase leads to a set of requirements for the market mechanism.

Design and Implementation. After identifying the requirements, the market algorithms and infrastructure can be defined and implemented. First, in the *conceptual design phase*, the market is set up in an abstract way, i.e. the institutional rules are defined without specifying the way they are implemented. During the *embodiment phase* the abstract

conceptual design is concretized, but still remains platform independent. Finally, in the *implementation phase* the market platform is realized.

Evaluation. Once the market infrastructure is set up, it can be *evaluated* whether the desired market outcome can be realized. Usually these evaluations are done with respect to the requirements specified in the environmental analysis and include technical aspects (e.g. performance, system reliability) and economic aspects (e.g. allocation efficiency).

Since the expression of offers, requests and contracts in a market typically includes complex description of goods or services that may involve broad domain knowledge as well as a wide-range of different parties, defining an appropriate market language easily becomes a cumbersome task. Therefore, ontology engineering provides structured means that support this task. The ontology engineering process in sketched in the following section.

2.3. Ontology Engineering

Several ontology engineering methodologies have been proposed in literature serving different purposes or addressing different domains [7]. For example, for ontology building from scratch TOVE, ENTERPRISE, METHONTOLOGY, the OTK-methodology and DILIGENT have been proposed. Pinto and Martins [7] compare ontology engineering methodologies using a general process containing the stages *specification*, *conceptualization*, *formalization*, *implementation*, and *maintenance*. Although the tasks classified within a certain stage differ slightly from one methodology to the next, they are sufficient to clarify the general dependencies to the other methodologies.

Specification. The objective of the specification stage is to identify the scope of the ontology. In this context, the domain that has to be captured and the intended users have to be specified. This also involves to determine requirements regarding the expressivity of the ontology language.

Conceptualization. In a second step, the identified specification is described with a conceptual model. Depending on the concrete methodology used, different conceptualization models ranging from informal models, such as mind maps^{TM}, to semi-formal models, like binary relations diagrams, might be used. These conceptualizations describe the basic concepts and relations relevant in a domain.

Formalization. After describing the required vocabulary and the relations between the vocabulary terms in a conceptualization, the conceptualization has to be formalized in order to get an unambiguous definition of the terms. The formalization stage involves defining concepts by restricting their interpretation to certain individuals in the domain. Thus, concepts and relations are mathematically well-defined, but are not yet serialized in a computer-interpretable format.

Implementation. In the implementation stage, the formalized and semantically well-defined model of the ontology is represented by means of a machine-interpretable syntax, as provided by OWL, for instance.

Evaluation. In this stage, the quality of the ontology is technically judged by a knowledge engineer. According to [7], this includes verification (i.e. is the ontology correct according to the accepted understanding of the domain), validation (i.e. does the ontology meet the specified requirements), and user assessment (i.e. judging the usability and usefulness of the ontology and its documentation).

Maintenance. During testing of the service-oriented architecture as well as of the market mechanism, updating and correcting of ontology modules might be required. Each update or correction should be verified carefully and the implemented ontology should be checked for consistency.

3. An Integrated Methodology

As the development of a SOA infrastructure involves traditional software engineering as well as designing a coordination mechanism and a communication language for exchanging offers, requests and contracts, an integrated engineering methodology is required that handles interdependencies in terms of time and required information. These interdependencies are captured by Figure 1 and discussed in the following.

3.1. Requirements Gathering

As a first step, the analysis of the existing software system and the domain decomposition of the planed application has to be carried out. The environmental analysis directly makes use of information derived from domain decomposition. Examples for requirements derived from the domain decomposition are the numbers of providers/customers that have to be handled or multi-attribute product descriptions. Such language-specific requirements are also direct requirements for the expressivity and the vocabulary of the ontology. The environmental analysis of the market engineering process represents a good starting point for the ontology engineering process. In addition, one cannot finish the environmental analysis in the market engineering process, before the specification phase of the ontology engineering has not been concluded. In all, the requirements gathering phases for services, markets and ontologies have to be finished, before the design and realization phases can be approached.

3.2. Design and Realization

The conceptual design of the market and the conceptualization of the ontology can be started during the component design phase. The conceptual design of the market requires to fix the language primitives and therefore depends on the

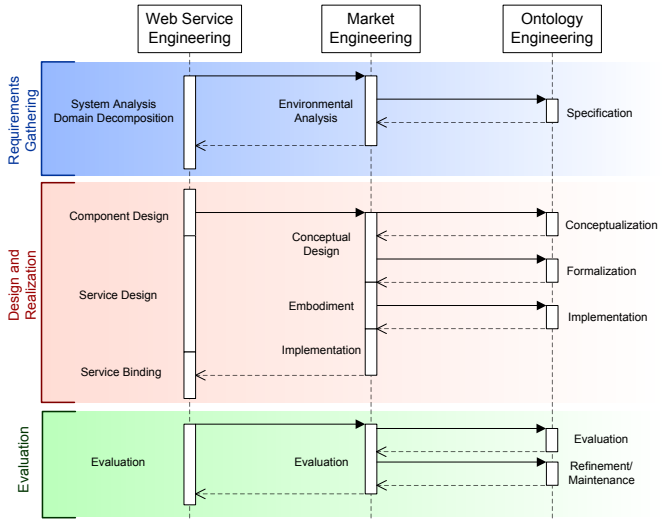

Figure 1. An integrated methodology for service-oriented systems on the Web.

conceptualization of the ontology. Before starting the embodiment phase the formalization of the ontology has to be finished since unambiguous, formal definitions of the bidding languages are required. As the result of the conceptual market design has to be a formal model, ontology formalization is also a part of the conceptual design phase of the market engineering processes. Thus, the market design and implementation phase has to be accompanied by the conceptualization, formalization and implementation of the appropriate ontologies. As ontologies are independent of a concrete implementation platform, they can be implemented in the embodiment phase. However, to enter the implementation stage of the market engineering process the concrete serialization of the ontology has to be known and thus a fully implemented ontology is required. In the implementation phase the market platform is realized. This involves, for instance, the implementation of the required matching and allocation algorithms based on an appropriate ontology reasoner. The market mechanism has to be fully implemented in order to realize service binding mechanisms.

3.3. Evaluation

In the evaluation stage all parts of the system are assessed with respect to the identified requirements. In this context, evaluation of one part may reveal problems caused by other parts. The problem has be corrected by going back to the corresponding engineering phases and repeating the following steps (possibly in all three engineering methodologies).

4. Conclusions

In this paper, developed a coherent methodology for establishing a service-oriented system on the Web. The paper is novel in showing how service, market and ontology engineering processes interleave in terms of required information and time dependencies. The entire methodology presented in this paper has been applied to design a Web service market platform for mobile, grid and enterprise services. As the discussion of these applications goes beyond the scope of this paper, the interested reader is referred to [4].

Acknowledgements

This work was partially funded by the Karlsruhe Service Research Institute (KSRI), German Research Foundation in scope of the research training program Information Management and Market Engineering, and the BMBF-project THESEUS.

References

[1] A. Abran, J. W. Moore, P. Bourque, and R. Dupuis, editors. *Guide to the Software Engineering Body of Knowledge (SWEBOK)*. IEEE Computer Society, February 2004.

[2] B. W. Boehm. A Spiral Model of Software Development and Enhancement. *IEEE Computer*, 21(5):61–72, May 1988.

[3] P. Kruchten. *The Rational Unified Proces: An Introduction*. Addison-Wesley, 3rd edition, 2003.

[4] S. Lamparter. *Policy-based Contracting in Semantic Web Service Markets*. PhD thesis, Universität Karlsruhe, 2007.

[5] D. Neumann. *Market Engineering - A Structured Design Process for Electronic Markets*. PhD thesis, University of Karlsruhe (TH), 2004.

[6] M. P. Papazoglou and W.-J. V. D. Heuvel. Service-oriented Design and Development Methodology. *Int. Journal of Web Engineering and Technology*, 2(4):412 – 442, 2006.

[7] S. Pinto and J. P. Martins. Ontologies: How can They be Built? *Knowledge Information System*, 6(4):441–464, 2004.

[8] W. Royce. Managing the Development of Large Software Systems. *Proc. of IEEE WESCON*, 26:1–9, August 1970.

[9] O. Zimmermann, P. Krogdahl, and C. Gee. Elements of Service-Oriented Analysis and Design - An Interdisciplinary Modeling Approach for SOA Projects. `http://www-128.ibm.com/developerworks/library/ws-soad1/`, June 2004. IBM developerWorks.