

Towards An Interdisciplinary Methodology for Service-oriented System Engineering

Veli Bicer*, Steffen Lamparter†, York Sure‡§, and Ali H. Dogru ¶

* Forschungszentrum Informatik (FZI), University of Karlsruhe (TH), Karlsruhe, Germany

Email: bicer@fzi.de

† Corporate Technology, Siemens AG, Munich, Germany

Email: steffen.lamparter@siemens.com

‡ GESIS - Leibniz Institute for the Social Sciences, Bonn, Germany

Email: york.sure@gesis.org

§ University of Koblenz-Landau, Institute for Computer Science, Information Systems and Semantic Web (ISWeb)

Email: sure@uni-koblenz.de

¶ Department of Computer Engineering, Middle East Technical University, Ankara, Turkey

Email: dogru@ceng.metu.edu.tr

Abstract—Service concept is evolving as a broad idea recurrently discussed in recent years as it originates from many disciplines such as marketing, operations and computer science. Despite this interdisciplinary nature of service concept, existing methodologies for building service-oriented systems provide limited support to include a complete coverage of various engineering phases for a unified development and are therefore not directly applicable to service-oriented architectures. In particular, the design of coordination mechanisms between service providers and customers as well as the design of a common vocabulary in an inter-organizational setting is not adequately addressed. To face these shortcomings, in this position paper, we present an interdisciplinary approach integrating recently proposed methodologies into one coherent engineering methodology. The methodology aims to build a system that extends the basic find-bind-execute paradigm of Web services on an electronic market platform.

Index Terms—Services Delivery Methodology, Services Delivery Platform, Web Services.

I. INTRODUCTION

In recent years, Web services have developed to a mature technology as standardized languages and protocols have become available, and consequently more and more interorganizational, service-oriented systems have emerged. As building such service-oriented software systems involves a wide range of technical, business and legal aspects, the engineering process becomes highly complex and requires an interdisciplinary approach that is not entirely 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. Although one might think that a service implementation is mostly realized as a component or an object of an information system, the *service* concept represents a designated abstraction as a major step in offering service capabilities to several service consumers. This abstraction, mostly fostered by Web service standards, is required to introduce new capabilities for today's enterprise systems in order to realize collaborative processes in a large span,

e.g. over the Web. Technologies like automated discovery, selection and invocation of new services, and the introduction of new roles (e.g. service providers, consumers, and brokers) are all developed with the basic assumption of loosely-coupled services. In addition, these technologies and roles are mainly driven by the requirements of business dynamics (i.e. as companies focus more on core competencies and outsource some services) as a result of service-based thinking. In brief, these major distinctions lead to three types of components required for implementing a service-oriented architecture [1]:

- 1) A *hosting platform*. This is where service providers can deploy and operate their service. The hosting platform also has to provide means for consumers to access and invoke the service.
- 2) An *intermediary* or *broker* that connects service providers and consumers. This intermediary determines the optimal allocation of services to consumers dynamically at runtime of the system. It requires employment of certain components that utilize service descriptions to perform management activities.
- 3) *Standard conventions* to ensure that services can interoperate with each other irrespective of their implementations. This involves communication language as well as interaction protocol aspects.

A toolbox of different technologies is required to realize these components: (i) *Web service technologies* provide a hosting platform and a set of standardized protocols, formats and language specifications that enable interoperability between services hosted on different platforms. Therefore, they partly cover components No. 1 and 3 introduced above. (ii) *Web service Markets* – as the place where service providers, consumers and intermediaries come together on the Web to advertise their services and requests – are the cornerstone of service-oriented architectures. Thus, results from the area of market theory should be considered when designing such a platform that provides mechanisms to dynamically determine

suitable bindings and methods for closing legally enforceable and manageable contracts. Determining suitable bindings may involve locales for negotiating with and selecting among many potential providers. Since markets bring together consumers and providers they are required to realize component No. 2. (iii) In order to enable service allocation at runtime, a service has to be annotated with all types of information needed to perform management activities (which may include automated discovery, selection, negotiation and composition) in addition to technical descriptions. The state of the art for formally describing services and their input/output data are *ontologies*. They come with a standardized logical foundation providing well-defined semantics that is required for matching of descriptions and for realizing a high degree of interoperability. Therefore, they are important for both components No. 2 and 3. (iv) *Processes* are the actual means of value creation in service-oriented architectures. Although they are indirectly related to the components stated above, their operation over the architecture is needed to be discussed in order to make use of services, ontologies and market in a unified understanding to service consumers. Broadly speaking, in this article, we consider processes as links between service consumer and architecture as they specify how autonomous services are converted into a business value.

Thus, a seamless integration of the different technologies is essential for realizing a service-oriented architecture. This work proposes an integrated, interdisciplinary methodology for designing service-oriented systems. We realize this by considering some existing methodologies separately designed for market, ontology, service, and process engineering and by identifying their relations and interdependencies. It is also aimed to show how these methodologies can contribute to the integrated methodology. We conceptually introduce the integrated methodology in Section II. In this context, we discuss the rationale for an integrated methodology as it is required for identifying the main building blocks of the service-oriented architecture. Additionally, we present four types of engineering phases (i.e. market, ontology, service and process) each of which is further explained in the corresponding subsections by detailing its steps to be followed and its contribution to the integrated methodology. Section III presents a conceptual architecture of a brokerage platform based on the results of an ongoing project. Finally, we conclude with a short summary and future research directions in Section IV.

II. AN INTEGRATED ENGINEERING METHODOLOGY

In order to obtain a system that seamlessly integrates different technologies, the integration has usually to happen already at design time. As discussed in the previous section, developing a SOA infrastructure also requires developing a coordination mechanism such as an electronic market bringing together service consumers and providers. This mechanism is realized as a result of a market engineering methodology which is a theoretically founded procedure of constructing electronic market platforms [2]. Each market platform requires market-level mechanisms such as algorithms for discovery/allocation of resources, a communication language for

exchanging offers, requests and contracts, and ontologies as a common vocabulary of economic activities. Although each of these components comes with an individual development methodology, overall development process cannot be executed independently, since there are several interdependencies in terms of time and required information. For example, one cannot perform service publication or process deployment in the architecture, before the engineering of domain ontologies has been concluded. These interdependencies are represented in Figure 1 as four engineering phases and their related concepts.

In addition to these interdependencies, the need for an integrated approach also results from the question of how a service-oriented architecture as a whole is built. Generally, there is an enormous amount of literature dealing with engineering software systems ranging from sequential methods such as the influential waterfall model [3] to iterative models which combine top down and bottom up approaches such as the Rational Unified Process (RUP) [4]. When we look more closely at the existing methodologies of service-oriented system engineering, it can be seen that realization of service-oriented architectures is considered as a special case of software engineering, where a functional decomposition exists from requirements to services (or components). Although this is useful (especially for a process-centric design), in reality, service-oriented architectures include other aspects such as the market platform, associated mechanisms, domain knowledge etc. Moreover, these aspects are mostly modeled by different parties and in different time frames. Therefore, a complete methodology needs to cover all these aspects and need to clearly identify the relationships among them.

With these ideas in mind, we present four types of engineering phases (see Figure 1) for our integrated methodology in the remainder of this section. Although, for the sake of brevity, we put each engineering phase as a separate subsection, their relationships with each other and the corresponding service-oriented technologies are clearly defined in the paper. Although the overall methodology advances starting from market engineering to process engineering, we consider it to be iterative and incremental performed by various actors self-reliantly. For example, by gathering knowledge about the market platform, a service provider can utilize the service engineering phase for a number of times in order to expose new services depending on its core competencies and position in the market.

A. Market Engineering

Long before the exchange of services between consumers and providers takes place, the mechanisms of underlying electronic market need to be defined. Building an electronic market that meets certain requirements, such as profit optimization or maximizing the transaction volume in the market, is a complex process. Therefore, the *market engineering phase* breaks down this complex process into less complex stages very much as software engineering does it for the implementation of complex software systems. Although it is structured according to similar steps as software engineering, market engineering

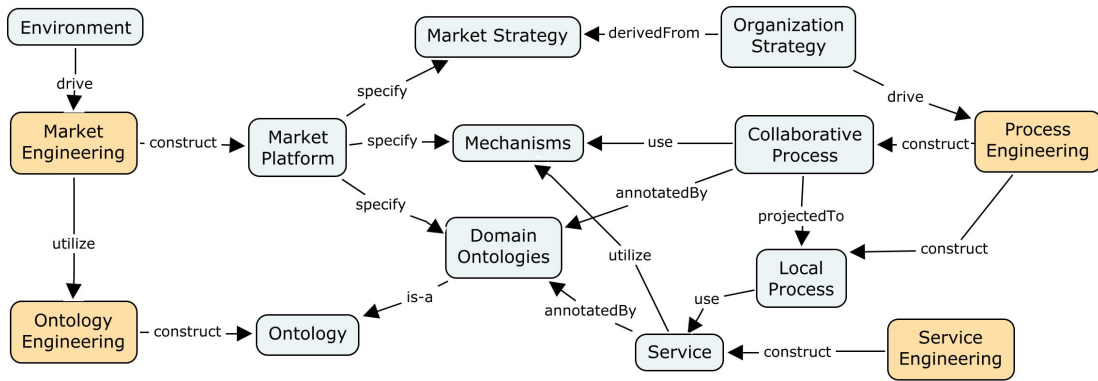


Fig. 1. Conceptual overview of an integrated methodology for service-oriented systems

focuses on different aspects and goals. The results of this process are very important for other phases in terms of specifying the requirements for domain ontologies, determining the target domain and identifying the overall market strategy. In this section, we briefly introduce the main stages to indicate its contribution to integrated methodology. A more fine-grained process accompanied with a detailed discussion for each stage can be found in [2].

Main Stages

Environmental Analysis. This stage deals with gathering information about the concrete setting for which the market is designed, including information about the participants, about services to be traded, possible intermediaries, etc. As first step *environment definition* analyzes the economic environment for which service-oriented architecture is built in order to construct a market definition. Based on the environmental definition, *market requirements* are also derived. Examples for such a requirement are scalability due to a high number of providers and customers or multi-attribute service descriptions to support Web services differentiability. Some requirements regarding information modeling are also direct input to the specification stage of ontology engineering as introduced in the next subsection.

Design and Implementation. After identifying the environment, market algorithms and infrastructure are defined and implemented. First, in *conceptual design*, the market is set up in an abstract way, i.e. institutional rules are defined without specifying the way they are implemented. This mainly requires the specification of service publishing and discovery mechanisms, the design of a bidding language and other market mechanisms such as allocation and contract formation functionality. Based on conceptual design, the market is implemented. In this step, the abstract conceptual design is concretized, but still remains platform independent.

The findings of this step is also used for the conceptualization/formalization of ontology engineering. For example, the information about the market mechanisms (e.g. discovery, publication etc.), or the bidding language is captured by formalizing the appropriate core, domain and application ontologies using a concrete ontology language. Therefore, this step can be done in parallel to the realization of ontologies.

Finally, after implementation, the market as platform is activated. This involves, for instance, the implementation of

required matching and allocation algorithms, integration of an appropriate ontology reasoner, or installation of a hosting platform for the deployment of market mechanisms, ontologies and/or services.

Evaluation. Once the market infrastructure is set up, it is *evaluated* whether the desired market outcome is realized or not based on the requirements specified in the environmental analysis.

Contribution to integrated methodology

As it precedes the other phases in the integrated methodology, one can come up with an analogy between domain engineering and market engineering. However, it should be noted that market engineering more focuses on the configuration of mechanisms how service value is co-created by interactions of several partners. In networked environments like Web service markets, where partnering model and integration of customers gain increased significance, this process involves a number of facets such as partner models, legal issues, profit model, and customer model to be specified. It is also considered that market engineering phase is intended to be conducted by service brokers, who mainly regulates service exchange process by specifying policies that consumers and providers need to obey, and maintained in corresponding brokerage platforms (e.g. service registry, or Web ecosystems). This slightly differs from domain engineering that aims to build a domain model and its associated domain theory. In this regard, we assume that a domain model is already present to conceptualize a mature domain of platform.

B. Ontology Engineering

Several ontology engineering methodologies have been proposed in literature serving different purposes or addressing different domains [5], [6]. Pinto and Martins [6] 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 other, they are sufficient to clarify the general dependencies to the other phases.

Main Stages

Specification. The objective of specification stage is to identify the scope of ontology. In this context, a domain is

selected to be captured and intended users have to be specified. This also involves the determination of requirements regarding the expressivity of an ontology language. As properties of the market are studied in detail in the environmental analysis, the results of the environmental analysis represents a good starting point for the ontology engineering process.

Conceptualization/Formalization. As second stage, the identified specification is described in a conceptual model. Depending on the concrete methodology used, different conceptualization models ranging from informal models, such as mind mapsTM, to semi-formal models, like binary relations diagrams, might be used. These conceptualizations describe basic concepts and relations relevant to the domain to be represented. Moreover, vocabulary is clustered into groups for modularization purposes. After describing the vocabulary and the relations between the vocabulary terms, 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.

Implementation. In this 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. Since ontology is still platform- and implementation-independent, this implementation of ontology is usually part of the design step of the market engineering process.

Evaluation/Evolution. In this stage, the quality of the ontology is technically judged by a knowledge engineer. According to [6], 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). Additionally, during testing and usage of ontologies, updating and correcting of ontology modules might be required. This triggers an ontology evolution process to make the required changes.

Contribution to integrated methodology

In a service-oriented system a wide range of information about the different services and their interrelation is typically available. In this respect, ontologies provide well-founded means to conceptualize shared information. An initial requirement for the use of ontologies is to capture the domain model. This allows to introduce concepts and semantics of global business domains to the architecture at first place. Then, we focus on the aspects that differ from the traditional software view, namely, the distinction between customers and providers. As already discussed, this distinction requires a coordination mechanism which is typically realized by means of a communication protocol. The three major communication primitives required in a service-oriented architecture are offers, requests and market policies [7, Chp. 2]. The extension capabilities (e.g. sub-classing, logical restrictions etc.) of ontologies enables the introduction these particular information as an addition to domain model. As more and more service-oriented architectures involve more than one organizational entity and thus a central point of control is missing, interoperability becomes

a crucial issue. In a networked environment ontologies also provide a means for achieving interoperability through the specification of standard syntax and semantics, and through their well-defined grounding in logics.

C. Service Engineering

The construction of market platform together with ontologies enable the service providers to expose their services into the platform with the mechanisms already agreed by the environment. Unlike the previous approaches proposed in [8] and [9], in an open market environment like Web, we assume that service development occurs independently from process development. This is mostly a bottom-up development process in which the service providers expose some functionalities of their underlying enterprise systems as low-level services in the architecture. The decision of which services to be exposed mostly depends on the core competencies of the service provider as well as the requirements of the market.

Main Stages

Service Innovation. This stage initializes service engineering with the aim of obtaining an innovative service idea to be converted into a service in the following stages. Although this terminology is rather new, it is regarded as an extension of formerly known goal-based and value-based requirements engineering approaches [10], [11]. A service idea is created by mining a number of ideas that are generated by service provider based on some collected information from user community, market or other relevant sources. Once a service idea is developed and documented, it needs to be evaluated considering consumers' needs, technological feasibility, and cost. A prototype implementation can also be implemented to support the documentation of service idea before its realization.

Component Design. *Component design* can be split into component identification, specification, realization and allocation. As a first action, 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 identified components, and other component dependencies are described. Missing components have to be custom built in the *component realization* step. After realizing application logic, its functionality can be exposed by providing an appropriate service. Defining a component that implements a certain service is called *component allocation*.

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 usage. This is done in the *service specification* step. In order to make services discoverable in a service-oriented architecture (e.g. required for dynamic binding), a service specification has to be machine-interpretable. For this purpose, WSDL and ontology-based specifications are used that allow, e.g., defining quality of service guarantees or information about the functionality offered

by the service. Typically domain-specific ontologies created as a result of ontology engineering are required in this step. If such ontologies are not mature enough to describe the services, an ontology evolution step is also triggered based on the feedbacks of service provider. Finally in *service realization*, service is deployed and advertised via publication mechanisms of market

Usage/Monitoring. In the last stage, service is *monitored* in terms of functionality, robustness, efficiency, etc. This stage mainly accounts for runtime measurements and aims to continuously improve quality of service.

Contribution to integrated methodology

Service engineering is a crucial phase performed by a service provider to transform an innovative service idea into a concrete service. During its application, a service provider benefits from other phases to conduct some tasks such as annotating or publishing Web services. For semantic annotation of Web services, ontologies are the technology of choice and several ontologies providing appropriate vocabularies have been proposed. Furthermore, market mechanisms are utilized to publish services to a brokerage platform, and allow its discovery, selection and usage by consumers. Apart from these, a successful service engineering phase allows service provider to decide worthiness of service idea, to design and implement it from its components and to maintain its sustainability for all service lifetime. Therefore, it is presented as a designated phase in the integrated methodology.

D. Process Engineering

The last engineering phase in our integrated methodology is the process engineering whose output is a business process realized over the market platform with some provided services. A process is an actual means of generating business value demanded by the customers [12]. Since they rely on the market platform for the execution, the market mechanisms (e.g. discovery, bidding language, allocation or contracting) act as a glue to bind the process descriptions with the required services in the architecture. Therefore, a top-down technique can be employed driven by the organization strategy and business goals.

Main Stages

Survey/Analysis. The phase starts with a *survey/analysis* stage. Here, scope of business process, overall goal and (internal and external) competences for its realization are determined. Based on the required competencies, an overall process is decomposed into its subprocesses using business use cases that enable identifying the functionality required as service. This stage requires extensive knowledge about the consumer's business models and organization strategy. Finally, information gathering over the market is conducted by discovering the candidate services.

Design. Based on the findings of previous stage such as the analysis of business goals and information about the market and services, the *design* stage aims to construct collaborative process that generates business value when executed completely over the architecture. Specification of a collaborative

process can be achieved by using a service choreography. This describes a high-level process that can later be projected to a number of executable, local processes and/or services.

Service Binding/Deployment. Once a collaborative process is specified in the *design* stage, it is projected to the local processes in order to be executed. In addition, the services that are required to execute these local processes are identified among some candidate services. According to this, an agreement process is performed in this stage to reach service-level agreements with service providers in the market. Market mechanisms such as bidding language, service configuration and contract functionality are all needed for this stage. Finally, services are bound to the business process.

Operation/Monitoring. The *Operation/Monitoring* stage comprises the activities of run time execution of the process.

Contribution to integrated methodology

Process engineering is based on the observation that each application that a business provides to its customers is the outcome of a number of activities – i.e. business process. Due to focus on core competences and integration capabilities of service-oriented architecture, processes can be realized by using the services provided by other parties. Therefore, process engineering phase enables to reach a business goal by facilitating an effective collaboration to build composite service-based applications. In this phase, process designers act as service consumers to find, select and agree on certain services available in the market according to their needs. For this, they rely on market mechanisms and existence of several services.

III. PLATFORM

In the previous section, we present the four-phase integrated methodology in a platform-independent manner to detail major steps and contribution to design process. However, the methodology requires different actors in different roles to perform numerous activities in various time frames in order to enable seamless service provisioning and procurement. This introduces the necessity for an underlying platform to support it by offering tool-support, management services, and technological capability to carry out certain tasks (e.g. service innovation, ontology engineering, discovery, selection, agreement etc.). It is developed as an effort of the THESEUS/TEXO¹ research project. TEXO brings together experts in the area of Web services to realize a platform to make Web services *tradable*.

Figure 2 illustrates high-level architecture of TEXO platform in terms of its components and roles. A quick analogy can be drawn from the TEXO roles to the actors of the methodology. *Service innovator* and *provider* mainly performs service engineering stages to build a service to be deployed to the platform. *Service consumer* is actual user of the platform as well as owner of external *service consumer runtime (SCR)* where a business process is engineered and deployed. A process is connected to the architecture with the help of *integration/adaptation manager*. Furthermore, both *platform*

¹TEXO Business Webs in the Internet of Services(<http://theseusprogramm.de/scenarios/en/texo>)

host and tradable services runtime (TSR) host are the roles that are responsible for constructing/specifying market mechanisms and maintain runtime functionality, respectively. Another role, called *community member*, consists of registered and non-registered users of the *TEXO Portal* providing feedback (e.g. rating, opinion etc.) for services. Major components of TEXO platform are as follows:

- *Management services and runtime.* Management services provide core platform functionalities for other TEXO components as a Web Service interface. They include all core functionalities to manage Web services including service registration, discovery, negotiation, billing, monitoring, idea mining, etc. All these services are deployed to *management runtime* to be accessible by other components.
- *Registry and repository.* These components offer the registration and discovery of both tradable services and management services. Repository stores all required information for TEXO components including user data, community content, service descriptions, SLAs, ontologies, monitoring data, pricing models or innovation data.
- *ISE workbench.* Services are engineered in a unified development environment, namely Integrated Service Engineering (ISE) workbench. It provides a model-driven approach and divides a service into five dimensions, namely: service description, workflow, data, people and rules [13]. The data dimension is specifically designed to model ontologies by conducting ontology engineering phase.
- *Innovation cockpit.* This is the main component to carry out all service innovation activities. It integrates different innovation-related search methods and tools. Service innovator can use the *innovation cockpit* for expert identification, idea lookup/mining, definition finding, and evaluation of idea – i.e. TEXO utilizes prediction markets (virtual marketplaces) to evaluate innovative service idea.
- *Portal.* This component offers a Web-based front-end to TEXO users for *management services*.

The availability of platform and tools is a key instrument for success of the methodology. It is also crucial to gather various users with distinct background and expectations in design process.

IV. CONCLUSION

In this position paper, we align different technologies, namely service-oriented architectures, electronic markets and ontologies, in order to develop a coherent methodology and architectural view for establishing a service-oriented system on a defined service market. We also introduce TEXO platform as a conceptual framework to support the methodology by gathering various actors and enabling tool-support. In this context, this paper is a first step to show how service, process, market and ontology engineering processes interleave in terms of required information and dependencies. As a future work, one unified methodology will be concretized and applied on TEXO platform.

Acknowledgments: This work was partially funded by the Karlsruhe Service Research Institute (KSRI), and also

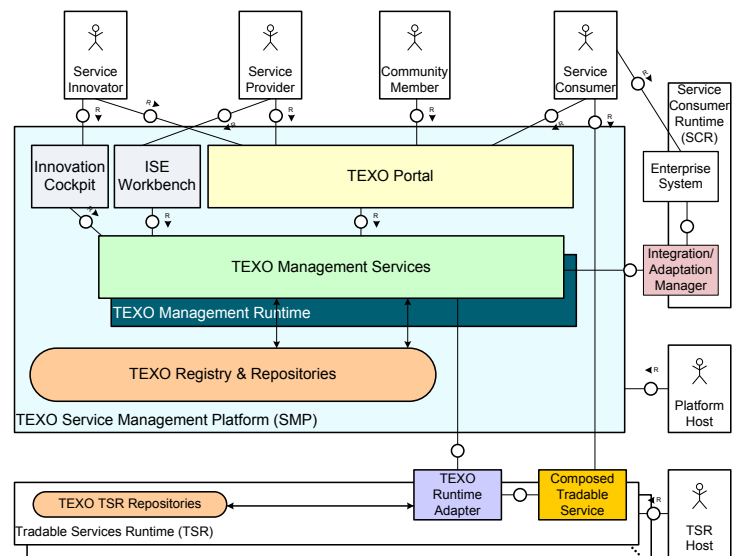


Fig. 2. Overview of TEXO Architecture

by means of the German Federal Ministry of Economy and Technology under the promotional reference “01MQ07012”. The authors take the responsibility for the contents. The authors also thank the reviewers for their valuable comments and insight.

REFERENCES

- [1] K. Govindarajan and A. Banerji, “HP Web Services Architecture Overview,” in *W3C workshop on Web services*. San Jose, CA, USA: W3C, April 2001.
- [2] D. Neumann, “Market Engineering - A Structured Design Process for Electronic Markets,” Ph.D. dissertation, Department of Economics and Business Engineering, University of Karlsruhe (TH), Karlsruhe, 2004.
- [3] W. Royce, “Managing the Development of Large Software Systems,” *Proc. of IEEE WESCON*, vol. 26, pp. 1–9, August 1970.
- [4] P. Kruchten, *The Rational Unified Process: An Introduction*, 3rd ed. Addison-Wesley, 2003.
- [5] O. Corcho, M. Fernández-López, and A. Gómez-Pérez, “Methodologies, Tools and Languages for Building Ontologies: Where is their Meeting Point?” *Data Knowl. Eng.*, vol. 46, no. 1, pp. 41–64, 2003.
- [6] S. Pinto and J. P. Martins, “Ontologies: How can They be Built?” *Knowledge Information System*, vol. 6, no. 4, pp. 441–464, 2004.
- [7] S. Lamparter, “Policy-based contracting in semantic web service markets,” Ph.D. dissertation, Universität Karlsruhe, 2007.
- [8] 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.
- [9] M. P. Papazoglou and W.-J. V. D. Heuvel, “Service-oriented Design and Development Methodology,” *Int. Journal of Web Engineering and Technology*, vol. 2, no. 4, pp. 412 – 442, 2006.
- [10] A. Van Lamsweerde, “Goal-oriented requirements engineering: A guided tour,” in *Fifth IEEE International Symposium on Requirements Engineering, 2001. Proceedings, 2001*, pp. 249–262.
- [11] J. Gordijn and J. Akkermans, “Value-based requirements engineering: Exploring innovative e-commerce ideas,” *Requirements Engineering*, vol. 8, no. 2, pp. 114–134, 2003.
- [12] M. Weske, *Business Process Management: Concepts, Languages, Architectures*. Springer-Verlag Berlin Heidelberg, 2007.
- [13] J. Cardoso, K. Voigt, and M. Winkler, “Service engineering for the internet of services,” in *Enterprise Information Systems X*. Springer, 2008.