Modern Software Engineering Concepts and Practices:
Advanced Approaches

Ali H. Doğru
Middle East Technical University, Turkey

Veli Biçer
FZI Research Center for Information Technology, Germany

Information Science REFERENCE

INFORMATION SCIENCE REFERENCE
Hershey · New York

Senior Editorial Director: Kristin Klinger
Director of Book Publications: Julia Mosemann
Editorial Director: Lindsay Johnston
Acquisitions Editor: Erika Carter
Development Editor: Joel Gamon
Production Coordinator: Jamie Snavely
Typesetters: Keith Glazewski & Natalie Pronio
Cover Design: Nick Newcomer

Published in the United States of America by
Information Science Reference (an imprint of IGI Global)
701 E. Chocolate Avenue
Hershey PA 17033
Tel: 717-533-8845
Fax: 717-533-8661
E-mail: cust@igi-global.com
Web site: http://www.igi-global.com

Library of Congress Cataloging-in-Publication Data

Modern software engineering concepts and practices : advanced approaches / Ali H. Doğru and Veli Biçer, editors.
p. cm.
Includes bibliographical references and index.
Summary: "This book provides emerging theoretical approaches and their practices and includes case studies and real-world practices within a range of advanced approaches to reflect various perspectives in the discipline"-- Provided by publisher.
ISBN 978-1-60960-215-4 (hardcover) -- ISBN 978-1-60960-217-8 (ebook) 1. Software engineering. I. Doğru, Ali H., 1957- II. Biçer, Veli, 1980-
QA76.758.M62 2011
005.1--dc22
2010051808

British Cataloguing in Publication Data
A Cataloguing in Publication record for this book is available from the British Library.

Chapter 6
Modeling Services Using ISE Framework:
Foundations and Extensions

Veli Bicer
FZI Forschungszentrum Informatik, Germany

Stephan Borgert
TU Darmstadt, Germany

Matthias Winkler
SAP Research CEC, Germany

Gregor Scheithauer
OPITZ Consulting München GmbH, Germany

Konrad Voigt
SAP Research CEC, Germany

Jorge Cardoso
University of Coimbra, Portugal

Erwin Aitenbichler
TU Darmstadt, Germany

ABSTRACT

The Internet of services introduces new requirements for service engineering in terms of addressing both business and technical perspectives. The inherent complexity of the new wave of services that is emerging requires new approaches for an effective and efficient service design. In this chapter a novel service engineering framework is introduced: the Integrated Service Engineering (ISE) framework. With its ISE workbench, it can address the emerging requirements of Internet of services. The chapter presents the foundations on how the service engineering process can be conducted by applying the separation of concerns to model different service dimensions within various layers of abstraction. Additionally, three novel extensions are presented to the aforementioned ISE workbench in order to enrich the capabilities of the service modeling process.

DOI: 10.4018/978-1-60960-215-4.ch006

INTRODUCTION

Several advances have been made to describe and model Web services. Examples of proposed approaches include the use of ontologies to describe services and interfaces (Kerrigan, 2005) (Paolucci & Wagner, 2006), the semantic annotation of Web services (Paolucci & Wagner, 2006) (Cardoso & Sheth, 2003), and the use of UML and UML extensions for Web service modeling (Lopez-Sanz, Acuna, Cuesta, & Marcos, 2008) (Sadovykh, Hahn, Panfilenko, Shafiq, & Limyr, 2009) (Dumez, Gaber, & Wack, 2008). All these approaches targeted the modeling of a relatively simple artifact: a Web service interface which was composed of data inputs, data outputs, and operations names. While some approaches (e.g. (Paolucci & Wagner, 2006) (Kerrigan, 2005)) went a step further and have also modeled goals, precondition, participants, control, etc., their scope and technical orientation have delimited their use outside the research community.

Web services (such as WSDL or REST services) are seen as IT entities. Nevertheless, the Internet of Services (IoS) also embrace what we call IoS-based services (Cardoso, Voigt, & Winkler, 2009) and requires combining and correlating business and operational descriptions with existing IT-based descriptions. While Web services define the pipeline between two companies and semantics Web services look into and explain what goes down the pipeline, IoS-based services provide capabilities to describe the business added-value of the pipeline itself.

When contrasted to Web services, modeling IoS-based services is a more complex undertaking since they are multi-faceted and must account for aspects such as legal regulations, community rating, service level agreements, pricing models, and payment need to be factored in to design a tradable entities (Cardoso, Voigt, & Winkler, 2008).

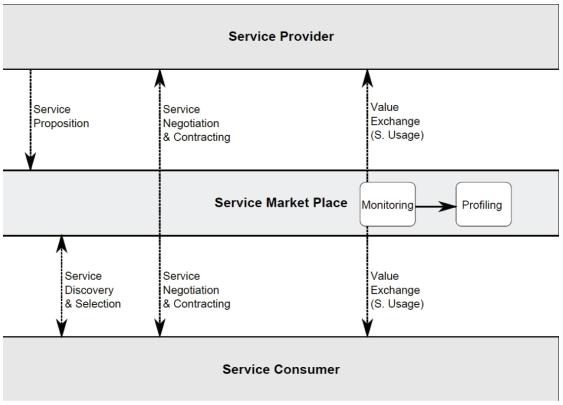
Due to the multifaceted nature of IoS-based services, their design is inherently complex. To cope with this density of facets, we conceptualize and implement the Integrated Service Engineering (ISE) framework (Cardoso, Voigt, & Winkler, 2009) (Kett, Voigt, Scheithauer, & Cardoso, 2009) and its software workbench (Scheithauer, Voigt, Bicer, Heinrich, Strunk, & Winkler, 2009) to enable the modeling and design of IoS-based services. By covering business, operational and technical perspectives, ISE provides a structured approach for service engineering. The structuring is achieved by following a separation of concerns (inspired in the Zachman framework (Zachman, 1987)) and a model-driven design.

In this chapter we present the ISE framework as two main parts. In the first part, we discuss the main characteristics of IoS-based services as an underlying motivation for the approach. Mainly, it is derived from the service concept that spans the definitions in various domains such as marketing, operations research, and information technology. The service concept allows to a generic service provisioning process that involves the actors interacting to achieve a common service goal. Then, we present the basics of the ISE framework in terms of different service dimensions and aspects required in an engineering process. ISE workbench is introduced as an instantiation of ISE framework with specific model editors and model transformations.

In the second part, we present three advanced extensions for ISE with novel techniques to guide service engineering. In this part, our contributions include: (1) techniques to model service processes using pattern matching, (2) modeling of service context, and (3) Service Level Agreement (SLA) management of composite services. The process pattern matching approach allows generating these service compositions semi-automatically by aligning business and IT. Furthermore, the semantic context modeling and service description approach provides a mechanism to enable complex service descriptions to be specified and interpreted based on context since services are subject to a vast amount of contextual information emerging dynamically during service procure-

ment. Finally, service composition results generally in more complexity in terms of functionality, resource, time and location aspects, and quality. The approach to dependency and SLA management for composite services (Winkler & Schill, 2009) supports providers to manage dependencies between services in their composition to assure its proper execution. Finally, the last two sections give an overview of the related work in service engineering and conclude our contribution with prospects about the future work.

Figure 1. Service trade



FOUNDATIONS

Internet of Services (IoS)

This section introduces ideas and concepts that are related with the Internet of Services. It is important to note that the term Internet of Services (IoS) spans ideas that are borrowed from other approaches with varying terminology. In this work, the terms (Web) Service Ecosystems and Digital Ecosystems are used synonymously to IoS.

Tertiarisation describes a structural change in developed countries concerning the sectoral composition. Countries shift from an industry economy toward a service economy. Drivers of this change include globalization, technological change, and an increasing demand for services (Peneder, Kaniovski, & Dachs, 2003). Considering this trend, it becomes clear that services and the service economy play an important role in today's and tomorrow's business. In line with this trend, Internet marketplaces for services emerge, such as Google Base, SalesForce.com, and SAP Business by Design.

The vision of IoS is an evolution of service orientation and takes services from merely integration purposes to the next level by making them available as tradable products on service delivery platforms (Barros & Dumas, 2006). They aim at trading services over the Internet between different legal bodies, compose complex services from existing ones, and IT-supported service provisioning (Janiesch, Niemann, & Repp, 2009).

Figure 1 depicts the steps involved in service trade: (1) service proposition, (2) service discovery & selection, (3) service negotiation & contracting, and (4) service monitoring & profiling.

Midst service proposition, service providers advertise their services toward potential consumers, whereas during discovery and selection, service consumers specify their service preferences toward providers. In case a service consumer selects an appropriate service, providers and consumers negotiate and finally agree on service levels (SLA) which are monitored throughout value exchange. In the event service levels are not met, compensations must be triggered. During service profiling, valuable information on services' performance is stored, which is gathered while value exchange and monitoring.

The rest of this section follows this structure: the next subsection introduces a service taxonomy that distinguishes between services in a general sense as well as their electronic counterpart and implementation. While the subsequent subsections outline the Internet of Services as an evolution of service-orientation, the following subsection introduces actor roles for the IoS. Additionally, IoS requirements or impediments will be discussed. The final subsection delineates a life cycle concept for services

Service Taxonomy

Before diving into definitions for IoS, this section outlines a comprehensive service taxonomy. The concept of a service is investigated in different research communities and is subject of different domains. This leads to different interpretations of the concept of a service in these fields. More precisely, it is defined differently in business science, information science and computer science. Baida et al. (Baida, Gordijn, & Omelayenko, 2004) surveys different definitions of the service leading to a taxonomy that distinguishes business services, eServices, and technical services. They directly relate to the service concepts that are in the focus of the three research fields mentioned above.

Distinguishing between business services, eServices, and technical services is useful because it directly relates to the process of transforming requirements derived in the business domain into software artifacts in the IT domain. Moreover, it will help to understand, which business services are amendable to be implemented as technical services. The remainder of this section will survey them more closely.

Business Services. A large variety of definitions for business services exist. The concept of a business service is not only a concept from a research perspective, but economists categorize companies according to this definition. Classically, services were defined as one of the three sectors in an economy: agriculture, manufacturing, and services, where services are everything that is neither considered as agriculture nor manufacturing (Sampson & Froehle, 2006) (Teboul, 2005). Thus, services were defined as a residual of concepts. In recent years, this residual has contributed an ever larger part of the total economic value creation and employed an increasing percentage of people. Another common classification for business services is to distinguish between Business-to-Business services (e.g. financing or logistics), Consumer services (banking, insurance, or education), and Self services (washing salons).

E-Services. The definitions of a service are largely developed in the business sciences. The scope of these definitions of services includes a large variety of economic fields including public services, health care services, transportation, or travel industry. Information sciences investigate how business services in these economic fields relate to information technology and refer to this subset of services as *e-services* (Baida, Gordijn, & Omelayenko, 2004).

Technical Services. The first two types of services in the classification taxonomy specify the service concept from a high-level point of view, especially with the interpretations in business science and information technology. Technical services, on the other hand, are described as an aggregation of the functionality specified in the other types and as the *realizations* by an underlying technological platform, e.g. Web services. Therefore, they can be regarded as an extension of the interdisciplinary service concept into computer science (Baida, Gordijn, & Omelayenko, 2004). According to the W3C Web Services Architecture Group (Booth, et al., 2004), a service is defined as "*an abstract resource that represents a capability of performing tasks that form a coherent functionality from the point of view of providers entities and requesters entities. To be used, a service must be realized by a concrete provider agent.*" As a specific incarnation of a service, they define a Web Service as "*a software system designed to support interoperable machine-to-machine interaction over a network. It has an interface described in a machine-processable format (specifically WSDL). Other systems interact with the Web service in a manner prescribed by its description using SOAP-messages, typically conveyed using HTTP with an XML serialization in conjunction with other Web-related standards*." These definitions of a technical service, in particular a Web Service, is consistent with other definitions (Papazoglou, Traverso, Dustdar, Leymann, & Kramer, 2008) (Kopecky, Vitvar, Bournez, & Farrell, 2007) (Preist, 2004). A service description is based on the

assessment the goals the service aims to achieve. These goals include non-functional properties, key performance indicators (KPI), or legal aspects which are related to the business level of a service. But the service description also needs to provide a description of its technical interface, message formats, and semantics of operations. This functional and technical perspective is linked to the technical *realization* of the service.

Internet of Services as an Evolution of SoA toward Marketplaces

In general, IoS comprises two main concepts. Firstly, it is a network architecture that tells how actors or peers or services interact with each other. Secondly, it is a marketplace that shows how to trade services over the Internet.

(Barros & Dumas, 2006) see Web Service Ecosystems (WSE) as an evolution of Service-oriented Architecture (SoA). The authors describe SoA as a novel paradigm in order to combine legacy applications, automate business processes as well as foster technical integration between different legal bodies. Contrary to implementing business logic into hard-wired applications, software developers define technical services as fine-grained, reusable, loosely coupled functionality, which in turn can be wired according to actual business requirements. Barros and Dumas refer to WSE "… *as a logical collection of web services* …" Recent developments show that once companies adapt to this paradigm, services are treated as valuable assets which can be exposed to other companies. Companies may offer and procure, and hence, trade these assets beyond organizational boundaries.

(Chang & West, 2006) on the other hand, who relate to the term Digital Ecosystems (DE), address the way of how actors interact with each other. The authors ascribe that this new development will shift the business to business interaction from "…*centralized, distributed or hybrid models into an open, flexible, domain cluster, demand-driven, interactive environment.*"

(Briscoe & De Wilde, 2006) see potential for optimization in the current way companies conduct their business in that they relate biological ecosystems to business ecosystems. Furthermore, the authors attribute the Internet as an enabler for this optimization.

(Janiesch, Niemann, & Repp, 2009) define IoS as service networks where a service is provided by different actors. The authors acknowledge that realization of such networks involves business services as well as technical details involving web service technology. Internet of services' main aims is foster service trade, ability to bundle services, which in turn open new markets for small and medium enterprises, so the authors say.

Actors in Service Trade

Following the discussion of different views on IoS this section outlines diverse players in service trade. Existing literature reviews in the area of service ecosystems (Barros & Dumas, 2006) (Riedl, Bohmann, Leimeister, & H, 2009) (Blau, Kramer, Conte, & van Dinther, 2009), business value webs (Tapscott, Ticoll, & Lowy, 2000) **and** IoS (Janiesch, Niemann, & Repp, 2009) find evidence for different roles for actors. All the same, actors may play more than one role in service trade. Table 1 gives an overview of different actor roles.

(Tapscott, Ticoll, & Lowy, 2000) distinguish between consumer, context provider, content provider, commerce service provider, and infrastructure provider. *Consumers* demand and consume goods and services. *Context providers*

Table 1. Overview of actors

Tapscott et al.		**Barros and Dumas**	
Consumer	Provider	Consumer	Provider
• Service consumer	• Context provider • Content provider • Commerce service provider • Infrastructure provider	• Service consumer	• Service provider • Mediator • Broker

provide a single face to the customer. They lead the process of value creation, in terms of orchestrating IoS in such a way that value meets consumer needs. They also provide a set of rules for each stakeholder in IoS. *Content providers* are the main value contributors. They actually design, create, and deliver goods and services to meet customer needs. *Commerce service providers* offer services with a cross sectional character. These services include financial management, security, logistics, and monitoring for example. They enable the stream of value creation in IoS. *Infrastructure providers*, finally, offer services in terms of communication platforms, computing, buildings, networks, facilities, and roads.

(Barros & Dumas, 2006) on the other hand, identify next to service consumers three different roles for actors in service ecosystems. *Service providers*, who provide services in the first place. *Service brokers* offer services from different providers. Their business model is to bring providers and consumers together, or enhance services with delivery functions for convenient service provisioning. *Service mediators*, on the other hand, generate value by customizing provider's standard services toward consumer's needs.

Requirements / Infrastructure and the Internet of Services

While the previous text outlines the IoS as a means for trading services over the internet, the following paragraphs elaborate on current impediments for realizing a successful IoS. (Barros & Dumas, 2006) for example outline the following issues: service discovery, conversational multiparty interactions, and service mediation and adaption.

Barros and Dumas pinpoint that the current *service discovery* process depends on keyword-based searches. It is assumed that service providers as well as consumers use the same keywords for describing and discovering them. According to the authors, this works well in closed environments but not for multi-actor marketplaces. Barros and Dumas advocate a combination of a free-text and ontology-based search.

Additionally, while trading services over the Internet, interactions between actors will exceed traditional request-response patterns. In consequence, IoS must support *multiparty interactions* as well as a formalization for defining them. Barros and Dumas foster two technical specifications for this: firstly, the Business Process Execution Language (BPEL) and secondly, the Web Service Choreography Description Language (WS-CDL).

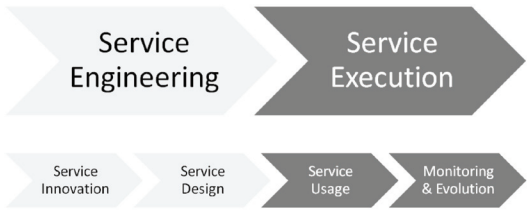
Another challenges lies in integrating purchased services into companies' internal service systems. In the scope of IoS, services may be used in contexts that were not initially considered by service providers, and hence, provide an interface that is inappropriate for others, including service mediators and brokers. This fact makes it necessary to *mediate* between services' given interface and an expected interface.

Service Lifecycle in the Internet of Services

A service runs through a number of states during its lifecycle. In general, the two states design time and run time can be distinguished. While during service engineering service ideas are transformed into operational and technical service implementations, during service execution services are consumed. This general distinction can be further refined into four phases in order to enable a fine-granulated management of these phases as well as transitions between them. Service design may be refined into *service innovation* and *service design.* Service execution on the other hand, may be refined into the stages *service usage* and *monitoring and evolution*. Figure 2 displays the four different stages.

Innovation processes in a service system may be quite different to the ones we know from dealing with (software) products because of the inherently different nature of services in comparison to products. In this section, we argue that cus-

Figure 2. IoS lifecycle



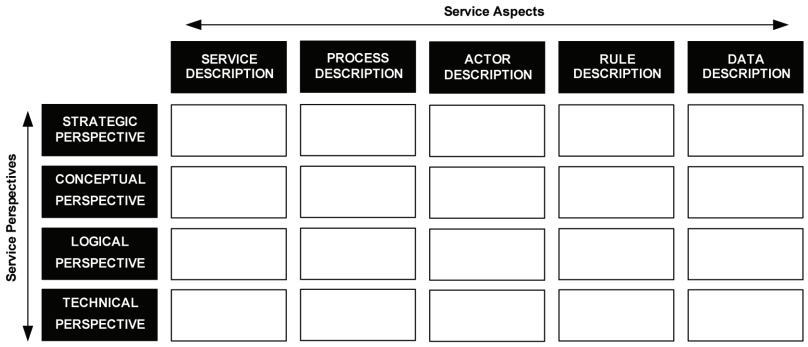
tomer input required during service provisioning is the main opportunity but also the main challenge for innovation in the services sector. An innovation usually implies the novelty of an idea linked to its (successful) realization. Today, the link between the innovation phase and its realization in the engineering phase is established in an ad-hoc way. Proprietary tools for brainstorming, idea evaluation and idea documentation are used. Successful service innovators rely on a collaboration tools and innovation processes which interlink the proprietary innovation tools using SOA technology.

Service engineering for both, service-oriented architectures and evolving service marketplaces in the Internet is still a challenge due to dynamic environments, high uncertainties, and increasing coopetition of market participants. An approach must supports service engineering in terms of planning, designing and implementing services, which are traded over the Internet, in addressing stakeholders from business & IT, acknowledgement of different service aspects, and utilization of model-driven architectures. This approach should not be limited to computing services; rather, it also should target business services, e.g., insurance & financial services, civil services, marketing services, and telecommunication services.

Service usage as the third phase relies on an expressive service description and embodies the following sub-phases: service discovery, service selection, and composition of services. The first step to realize services is to express them in terms of service descriptions in order to expose the functionalities and capabilities to the service consumer (e.g. human or software agent). The initial attempt in this direction has been to provide a service interface - borrowing the idea from previous component-oriented approaches (Herzum & Sims, 2000). This enables the software artifacts to be abstracted in a well-defined, platform independent way and hides the implementation details to achieve a loosely-coupled architecture (Booth, et al., 2004). As a common standard, Web Service Description Language (WSDL) (Christensen, Curbera, Meredith, & Weerawarana, 2001) fulfils this need by describing service operations, input and output parameters, and endpoints. The services, expressed through service descriptions, need to be discovered by potential consumers to whom they offer a business value. Technically, this is initially addressed by the Web service registries, namely UDDI (Bellwood, et al., 2002) and ebXML (Fuger, Najmi, & Stojanovic, 2005). They enable the service providers to publish the service grounding to a central repository and annotate it within a basic classification scheme. The consumer can then select a service suitable to her needs. In fact, both Web service registries are basic implementations of a broader conceptual component that is called *discovery framework* (Studer, Grimm, & Abecker, 2007). It is a harmony of all the mechanisms and tools required to utilize discovery. Basically, a discovery framework relies on three essential elements: capability descriptions of services, request descriptions of consumers, and comparison mechanisms to match the capabilities and requests. For the instance of ebXML registry, an external WSDL document, registry information model, or filter queries can be stated as the examples of such mechanisms. The usage of Web service registries are often limited for the service discovery although there are some approaches to extend them with semantics (Dogac, Kabak, Laleci, Mattocks, Najmi, & Pollock, 2005).

While *service monitoring* IT services (such as WSDL or REST web services) are usually seen mainly as a technological problem, the monitoring of business services adds the requirement of

Figure 3. Service perspectives and aspects in the integrated service engineering (ISE) framework



also monitoring business aspects. Monitoring IT services usually targets to measure network attributes such as latency, packet loss, throughput, link utilization, availability and connectivity, one-way delay, one-way packet loss, round trip delay, delay variation, and bulk transfer capacity. (Moser, Rosenberg, & Dustdar, 2008) recognize that web services currently lack monitoring mechanisms and they provide a solution based on the interception of SOAP messages exchanged during runtime. The emphasis is on technical aspects. On the other hand, the monitoring of business services can only achieve its full potential when it addresses the business level and accounts for organizations' strategies. Compared to IT monitoring, business monitoring is more complex since services are intangible, often inseparable, immersive, and bipolar.

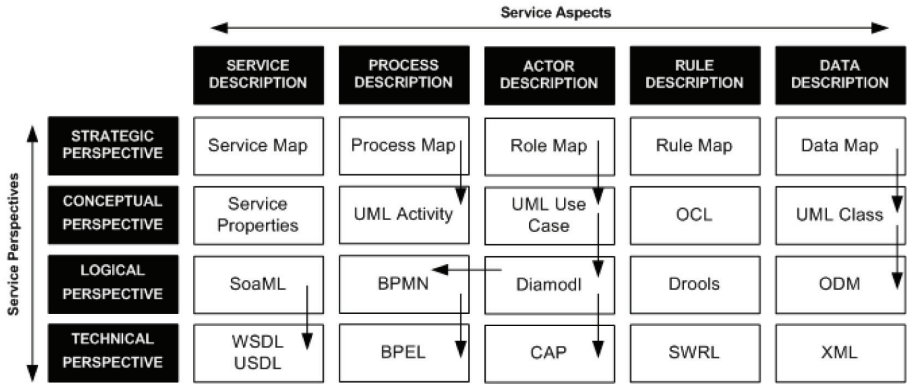
ISE Framework

Based on a state-of-the-art study of existing frameworks, (Kett, Voigt, Scheithauer, & Cardoso, 2009) argued that existing frameworks for service engineering either address the business perspective or the technical perspective. To overcome the gap between these approaches, the ISE Framework is introduced as depicted in Figure 3. The framework builds on the Zachman framework (Zachman, 1987) and a service engineering methodology for service products (Bullinger H., 2003). The vertical axis shows four perspectives of the engineering process and is named *service perspectives*. Each perspective relates to a specific role with appropriate skills and offers different sets of tools and methods. It also implies the chronology of the framework for they are linked to phases of the service engineering process. The horizontal axis shows five different *descriptions of a service*. Each description is valid for each perspective. Each intersection in the matrix is placeholder for a meta model, a notation, and activities, which are appropriate for the respective perspective and the modeling aspect.

Service Perspectives

Business strategists pick up new service ideas and focus on requirement analysis in the *strategic perspective*. (Kett, Voigt, Scheithauer, & Cardoso, 2009) depicted a basic underlying model for this perspective: the Business Model Ontology (BMO). Eventually, a decision is made whether to implement a new service or not. The *conceptual perspective* focuses on operationalizing and implementation of strategic artifacts from the owner's perspective. The final artifact is a service design which is neither technical nor platform-

Figure 4. The integrated service engineering (ISE) workbench implementing the ISE framework



specific. Conceptual artifacts are transformed into formal models during the *logical perspective* by IT analysts. This perspective offers a bridge between service design and technical service implementation. Finally, the IT developer transforms the logical artifacts into platform-dependent software artifacts, e.g., BPEL (Alves, et al., 2007) and WSDL (Christensen, Curbera, Meredith, & Weerawarana, 2001), etc., during the *technical perspective*.

Service Aspects

The *service description* embodies services' value proposition toward potential customers. This includes functional, financial, legal, marketing, and quality of service properties as well as other meta data for service proposition, discovery, selection, contracting, and monitoring. The *process description* addresses services' behavioral aspect, which includes core capabilities and sequence flows. The *actor description* offers means to model and to refine human resources, and to assign tasks. Intangible assets, terms, and concepts as well as their relationships are defined in the *data description*. The *rule description* addresses organizational rules. These are defined to elicit and formalize domain knowledge to guide services' behavior.

ISE Workbench

The Integrated Service Engineering (ISE) Workbench implements the ISE Framework (cf. Figure 4) and supports an interdisciplinary structured service engineering process to develop services that can be traded over the Internet. The work on the workbench started in April 2008 and is a prototype, which is still under development. Developers add new features as well as improve existing ones. For example, the business rule aspect is not implemented, yet. The ISE Workbench builds on Eclipse's Rich Client Platform (RCP), which allows an integration of existing tools as well as offers a platform for novel tool development. The workbench embodies a total number of 20 editors in order to model the five service aspects for each of the four perspectives. OMG's Query View Transformation (QVT) specification is the basis for model transformation implementation, e.g. BPMN (White, 2004) to BPEL (Alves, et al., 2007).

Main Functionality & Notations

In order to support the ISE Framework with its 20 intersections, available notations were analyzed. Figure 4 depicts the resulting 20 modeling notations. This set of notations is only one possible

selection. For each chosen notation, a suitable editor was integrated into the workbench to design all service aspects from different angles. The *strategic perspective* uses the mind map notation to elicit the information. The *conceptual perspective* employs mostly the UML diagrams, a semi-formal graphical notation. Whereas, the *logical perspective* makes use of formal notations, the *technical perspective* applies formal languages, such as BPEL and WSDL.

Next to existing notations, new languages were developed, where necessary. The *service property* notation is a domain-specific language and describes services from a provider's perspective in a non-technical fashion and includes information about capabilities, price & payment, delivery channels, rating, legal aspects, and provider details in order to facilitate service discovery. The *Universal Service Description Language (USDL)* (Cardoso, Winkler, & Voigt, A Service Description Language for the Internet of Services, 2009) is a XML specification that holds facts about business information, operational information, and technical information related to the service. The *Canonical Abstract Prototypes* (*CAP)* editor provides an abstract description of a user interface structure. Finally, the *service archive (SAR)* is an XML schema and denotes how to bundle technical models for deployment.

Model Transformations

The ISE Workbench offers model transformation for flexibility, speed, and accuracy in design. Since the union of all models defines a service they need to be integrated and synchronized. This integration task is facing major challenges because of the various people involved within the development process and the rising complexity of the models. To cope with these challenges we propose to integrate the models automatically by model transformations.

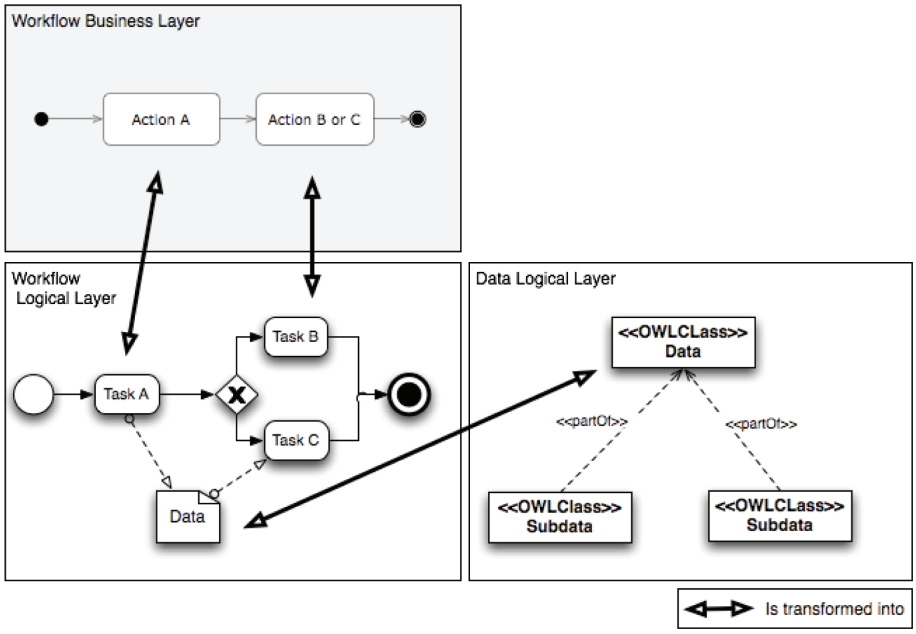
The ISE models contain artifacts representing a service's five dimensions: service, process, actor, rule, and data. Furthermore, each of these models is divided into four layers (levels) of abstraction. This leads to multiple representations of information on different layers of abstraction in the corresponding dimensions. Changes in one model have to be propagated into the affected models holding the overlapping information. This is a time-consuming and challenging task since each of the models has to be aware of changes and needs to be adjusted. For a structured approach we separate the dependencies between models into two classes: vertical and horizontal.

Vertical dependencies cover the synchronization of dependencies between models on different layers of abstraction in one dimension. It represents the bridging of layers of abstraction by transforming between multiple representations of artifacts.

Horizontal dependencies define the synchronization of models on the same layer of abstraction. This describes dependencies between models of different dimensions which refer to artifacts of other dimensions. This also includes multiple representations of an artifact on the same layer of abstraction.

These dependencies form the integration of the models and have to be implemented manually or by automatic support. Being more precise, a dependency is defined by a mapping. Formally a mapping assigns to a set of artifacts a set of artifacts; where one sets corresponds to the other. That means the different representations of information are assigned to each other. To illustrate the dependencies, Figure 5 Example for vertical and horizontal model transformation. Figure 5 shows an example which depicts the dependencies between two layers of abstraction as well as between models on the same layer but of different dimensions. The process dimension shown is specified regarding the conceptual and logical layers. The conceptual layer is represented by an UML activity diagram. The Business Process Modeling Notation (BPMN) is used to represent the logical layer. The arrows depict artifacts that

Figure 5. Example for vertical and horizontal model transformation



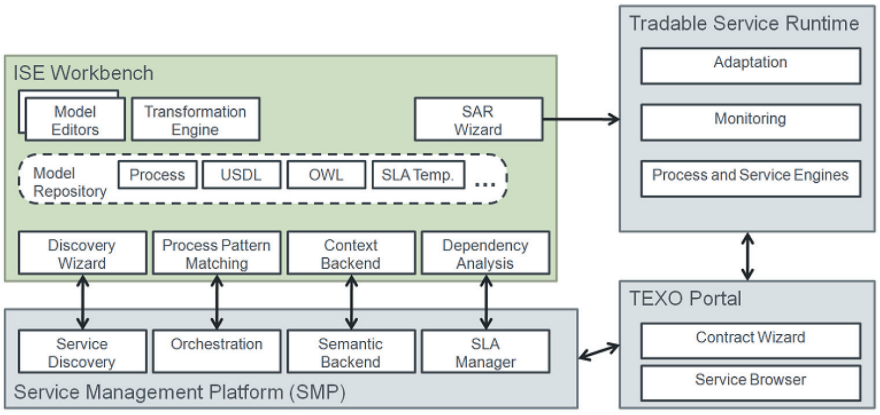
need to be synchronized and are mapped onto each other.

The Actions modeled in the activity diagram are again represented in BPMN as tasks. Therefore, *Action A* needs to be synchronized with *Task A*. That means that UML actions need to be mapped to BPMN tasks. The *XOR* between *Task B* and *Task C* of the BPMN model is mapped from *Action B or C* of the UML model. Furthermore, the *Information I* artifact used in the workflow is defined in the OWL-model (i.e., it depends on it). When one model changes (e.g. renaming or deletion), the depending models have to be updated. These updates can be done manually or by providing an automatic support. One solution to enable an automatic approach is by using model transformations for implementing mappings.

The first step to enable the implementation of model transformations is to define one common formal representation of models. This can be done using ontology formalism or more mature concepts like the Meta Object Facility (MOF). Based on this formalism, a domain specific language for model transformation can be used to define rules and apply them to the models. During the last years many model transformation languages have been proposed, both by academia and industry. For an overview, we refer to Czarnecki and Helsens classification of today's approaches (Czarnecki & Helsen, 2006). The two most prominent proposals in the context of Model Driven Architecture (MDA) are Query, View and Transformation (QVT) and the ATLAS Transformation Language (ATL).

We have chosen to rely on MDA to support model transformations because of matured concepts, well established infrastructure for model management and transformation, and available OMG standards. A model transformation is the process of converting one model to another model of the same system. Thus a model transformation is an implementation of a mapping (model dependency specification). We follow Kleppe and Warmer (Kleppe & Warmer, 2003) refining this definition to an automatic generation of a target model from a source model, following a transfor-

Figure 6. Architecture of ISE workbench



mation definition. A transformation definition is a set of rules describing formally how a source model can be transformed into a target model. Using a rule-based language like QVT to define model transformations executed by an engine allows for incremental and traceable transformations.

For automatic model integration we argue for model transformations as the implementation of mappings. Using and applying these concepts enables automatic model synchronization. This supports both the implementation of vertical and horizontal dependencies, thus reducing the complexity, effort and errors in modeling a service using ISE.

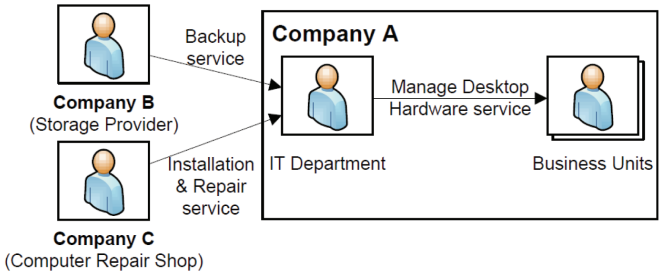
The ISE Workbench also offers deployment capabilities for seamless service execution. The service archive (SAR) is an XML schema and denotes how to bundle technical models. After service design with the ISE Workbench, the tool generates a SAR file and deploys it on a service runtime environment.

ISE Architecture

The ISE workbench is a part of larger TEXO service ecosystem architecture as depicted in Figure 6 in detail. The overall architecture mainly includes four components including the ISE workbench to perform further operations for a seamless service provisioning. The ISE workbench component is built on an Eclipse[1] platform and has several internal building blocks: Model editors, model repository and model transformation engine enables to develop services in a model-driven way as introduced in the ISE matrix above. Specifically, we have 20 separate models each of which is associated with the corresponding editors and transformation among them.

In addition, SAR wizard interacts with the Tradable Service Runtime (TSR) which is a component to handle deployment and execution of a service in the Service Execution phase of the service lifecycle. It mainly includes Adaptation, Monitoring and Process/Service Engines required for runtime functionality. Besides, there are other blocks included in the ISE Workbench to enable the service engineer to interact with Service Management Platform (SMP) for further service related tasks. Discovery Wizard enables the service engineer to interact with Service Discovery that, in turn, searches the repository to discover available services to be composed into the service. Process Pattern Matching, Context Backend and Dependency Analysis are all special extensions to the ISE workbench which will be explained in detail in the subsequent sections.

Figure 7. Overall process of hardware maintenance service



Finally, the TEXO Portal is an end-user interface which does not have a direct connection to the ISE workbench, but its functionality to negotiate service agreements, to search for available services and to test and execute them is very crucial in the architecture. It allows the end-users to use the services engineered in the ISE workbench with the help of the TSR and SMP components. Currently, the ISE workbench employs some well-known editors for service engineering, which are widely adopted by Eclipse community. These include the WSDL editor or the BPEL editor. Some other editors are also developed from scratch such as the SLA model editor or the context modeler.

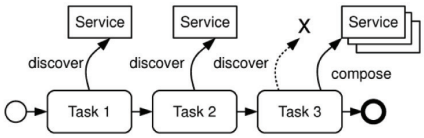
MODELING EXTENSIONS

Running Example

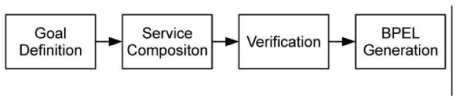
This section introduces an example that is utilized throughout the rest of the chapter to motivate the three advanced modeling ISE extensions. Figure 1 shows three companies. Company A's IT department offers the Manage Desktop Hardware (MDH) service, that allows outsourcing the purchase and the maintenance of computer hardware. The service's target customers are company A's own business units who pays for computer hardware leasing. The benefits for business units include lower transaction costs, lower labor costs, lower IT costs, and latest hardware. The MDH service is provided with four service levels: (1) out of order hardware is to be replaced within two hours, (2) new hardware is installed within two working days after ordering, (3) every 24 hours a backup is performed, and (4) backups can be played back within 30 minutes. The price that business units need to pay for the MDH service depends on contextual information, including business units' location, usage data, and whether business units are standard or premium customers. The IT department itself utilizes two services in order to offer desktop hardware management. Company B, a storage provider, performs a backup service and company C, a computer repair shop, conducts installations as well as repair services at business units' location.

In order to develop the MDH service, service provider (i.e. IT department) needs to deal with some challenges in terms of realizing a service composition, describing the realized service and ensuring its proper functioning. First, the service provider should be supported to identify suitable services that can be reused for realizing the MDH service. This is especially important to assign concrete services to high-level business tasks for an executable process realization. Furthermore, there is a need for a mechanism to describe the service in such a way that service context is taken into account. Specifically, some service properties (e.g. service price) cannot be determined in advance due to the dynamicity and dependency on the context. Another challenge for service provider is to manage the interplay of services for a proper functioning. Here, SLA

Figure 8. Transformation from BPMN to BPEL



(a) Assignment of services to BPMN tasks and service composition in case that service discovery fails.



(b) Automatic service composition: steps from goal definition to executable process.

compatibility comes into play since different problems may occur due to SLA incompatibility such as violations (e.g. negotiated composite QoS (new hardware in 24 hours) cannot be met due to negotiated atomic QoS (backup data, replace hardware, and restore data).

Service Composition by Process Pattern Matching

To manage the transition from the business perspective to the technical perspective (Cardoso, Voigt, & Winkler, 2009) (Kett, Voigt, Scheithauer, & Cardoso, 2009), ISE supports the transformation of BPMN models to executable BPEL models. As shown in Figure 8(a), two different methods are supported to assign services to BPMN tasks. The first method is a direct transformation, where the designer statically assigns a set of existing atomic or composite services to a task. Alternatively, if a suitable service does not already exist, the designer can add a goal specification to the task which is represented by a fragmented process model constructed by process patterns. This specification can then be used by the automatic service composition component to automatically compose services that satisfy the given specification.

The automatic service composition component applies process pattern matching to identify suitable services. An important foundation for this action is the use of a formal description language for processes with well-defined semantics. For that reason, we have chosen the Parallel Activities Specification Scheme (PASS) (Fleischmann, 1994) as a description language. PASS graphs allow to model processes in a subject oriented way, which is also well-suited for SOA. Subject-orientation introduces an approach that gives a balanced consideration to the actors in business processes (persons and systems as subjects), their actions (predicates), and their goals or the subject matter of their actions (objects). It is based on the fact that humans, machines and services can be modeled in the same manner. All receive and deliver information by exchanging messages. Humans, e.g., exchanges emails, office documents, or voice messages. Furthermore, the subject-oriented modeling approach enables the modeling of business processes in any arbitrary size because of the feature of composing services which is provided by the underlying model.

Reusability is one of the main motivations for the SOA paradigm and in the context of TEXO, the reusable modules are web services. In the case of a large number of services, automatic composition methods gain importance and one requirement for automatic composition approaches is the use of formal service descriptions.

The formalism of PASS is founded on top of the process algebra CCS (Milner, 1995) (Calculus of Communicating Systems) and the language constructs of PASS can be transformed down to pure CCS. Process algebras provide a suitable means for modeling distributed systems. They offer well-studied algorithms for verification and formalisms, e.g., for defining behavioral equivalences. In addition, the CCS hiding operator facilitates a hierarchization and modularization of the model, allowing to handle business processes of arbitrary size.

Figure 9. Parallel activities specification scheme (PASS) models

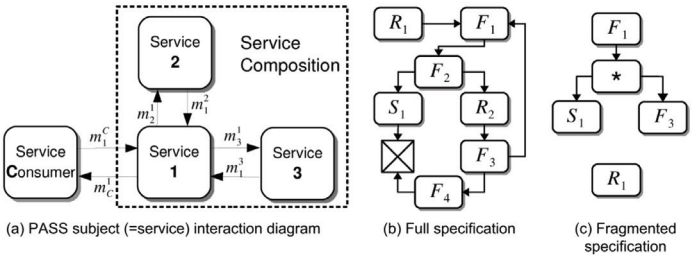


Figure 8(b) shows the most important steps of our service composition method. In the following, we briefly describe each of these steps.

Defining a Goal for Business Tasks

In order to add a goal specification to a BPMN task, we extended the BPMN editor by an additional property sheet to create a goal for each task, but this is only necessary if no suitable service could be found.

To specify goals, we have integrated the jPASS editor from jCOM1[2] into the ISE workbench. Figure 9 shows the basic concepts of the specification scheme. The two model levels, subject interactions and internal behavior of service, are available to specify a valid goal. Figure 9(a) depicts the subject interaction level and Figure 9(b) and Figure 9(c) depict examples for the internal behavior of services. The relationships between subjects and the types of exchanged messages are defined on the subject interaction level. The description of service behavior is explained in more detail below.

In accordance to our motivating scenario, the Service Consumer represents the Business Units, Service 1 the IT department, and Services 2 and 3 the Backup Service and the Installation & Repair Service. We assume that the IT department conducts the most maintenance and backup work by itself, but in certain cases it is dependent on both external services. How to model the internal behavior of each service is shown in the Figure 9(b) and 4(c). It is modeled using three different basic types of activities:

1. **S**end message.
2. **R**eceive message.
3. **F**unction (= F Figure 9 (b), (c))

The first two types enable services to exchange messages, and the function activity allows to call internal functions. The ⊠ symbol marks the end of the process description. To make the matching of activities work, it is vitally important that the activities in the search pattern and in the process description of a service are modeled using the same vocabulary. To ensure this, an activity catalogue is also introduced.

To describe process patterns, the PASS language was extended. Figure 9(c) shows a process pattern. In contrast to regular PASS graphs, process patterns do not have to be fully connected graphs and may contain the _ wildcard operator. This operator is a place holder for arbitrary subgraphs and is part of the fragment depicted in Figure 9(c). The process patterns are used for service matching and their modeling differs from that of fully-specified processes in the following two aspects:

1. In the model of the composite service, only activities which are essential for the process are specified. This simplifies modeling since the service engineer does not have to specify all functionalities and does not have to take care about each detail activity. E.g., he could omit modeling the payment branch in the process. If services have such branches, they would still be included, unless the engineer

explicitly models the exclusion of certain behavior.

2. The order of activities can be defined in a more general way as in traditional process models. The ~ operator can be used in conjunction with multiple isolated subgraphs to express an order between activities, instead of a single sequential order. This is useful, e.g., to enforce a certain behavior or communication pattern, while only concentrating on the essential parts of a process.

Service Composition

The first step in a composition is to find matching service candidates. To match goal specifications with service descriptions, we use the programmed graph rewriting system GRL. GRL stands for Graph Rewrite Library and is a Java library that provides the core functions of a graph rewriting system by supporting queries and rewrite operations. Rewrite rules are described in the declarative language GRL RDL (Rule Description Language). GRL operates on directed, attributed graphs, whose data structures are defined by the respective application. Nodes and edges of the graph can be attributed by arbitrary Java objects. Its basic building blocks are predicates (tests) and productions (rewrite rules). Rewrite rules are specified textually Complex attribute tests and transformations can be performed by calling Java methods from inside RDL programs. RDL programs are compiled, optimized using heuristics, and then executed on a virtual machine. Hence, GRL provides highly efficient graph matching. The rule description language RDL is nondeterministic.

The service descriptions are used as work graphs and the goal specifications are translated into query expressions in the language GRL-RDL. To match the pattern with services, it is required that each service comes with a fully-specified PASS description. Applying graph algorithms leads to candidate lists for each specified pattern or goal.

Verification

The graph-based representation is suitable for finding candidate services, but it is not directly suitable for verification, because it lacks a theoretical foundation. For this purpose, we transform a graph into a CCS description and use this formal representation for advanced validation.

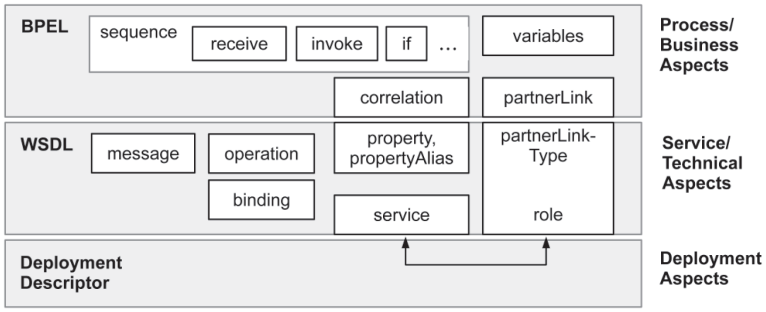
We currently use the CWB-NC Workbench[3] which supports various behavioral equivalences as well as features such as model checking. The model checking rules are described with the μ-calculus, which is temporal logic. Firstly, this allows to identify services that expose equivalent behavior. At runtime, such services might be used as replacements in case the original service fails. Secondly, a choreography conformance check can be performed. In a valid composition, it must be ensured that the involved services are able to communicate with each other. For this purpose, we have developed a method for checking the communication of each pair of services. Two requirements have to be fulfilled: First, the statical interfaces of both services have to match and second, the dynamic interfaces have to match, i.e., the communication pattern has to match.

BPEL Generation

To determine all possible combinations of services, the first step was to discover all candidate services using process graph pattern matching. Next, these combinations were checked in the verification step and all incorrect combinations were discarded. Finally, for each valid combination, an executable BPEL process is generated, which orchestrates the constituent services.

In order to deploy a process on a process execution engine, several additional files are needed beside the main BPEL file describing the process. Figure 10 shows how business, technical, and deployment concerns are separated and how the different description files are interrelated.

Figure 10. Separation of concerns in BPEL processes



The generation of BPEL starts from the subject interaction diagram as shown in Figure 9(a). In the previous processing steps, a list of candidate services has been generated for each subject in the diagram. If existing services are used, then the corresponding WSDL files, which describe the technical interface, already exist. Alternatively, if a subject has a fully specified PASS graph, then an executable BPEL process together with a WSDL interface description can be automatically generated for the subject. This involves the following parts:

- *Process:* To generate the main flow of the process, the elements of the PASS description are processed according to the control flow and corresponding BPEL elements are generated.
- *Variables* hold the state information associated with each instance of a process. Because the generator generates code to orchestrate existing services with given WSDL files, it has to generate message mediation code translating web service requests and replies to the types of the process instance variables as well.
- *Correlations* are needed during asynchronous interactions with services. When the process invokes a service and later receives the reply, it must be able to identify the correct process instance to which the incoming message belongs. The set of key variables used to uniquely identify a process instance is defined in correlation sets.
- *Properties and PropertyAliases* define mappings from the properties in variables to the properties in service-specific messages. They allow to describe which properties of different message types are equivalent, despite their different names.
- *PartnerLinks* describe the possible interactions between every interacting pair of services and defines their roles in the interaction.
- *PartnerLinkTypes* map from roles to port types and thereby define the message types exchanged during the interaction between partners.

Our generator supports ActiveBPEL and Apache ODE as output formats. While the BPEL and WSDL parts are standardized and (in principle) portable, the different engines require some proprietary supplemental files, which concern deployment aspects:

- *Deployment Descriptor* (for Apache ODE): The deployment descriptor defines which services a process provides and which services a process uses. This is done by linking the PartnerLinkType tags defined in the WSDL of the process to the Service tags defined in the WSDL files of the corresponding services. This simple deploy-

ment descriptor only points to the services of the first valid composition.

- *Process Deployment Descriptor* (for ActiveBPEL): Similar to above, this file defines which services a process provides and which services it uses, but in a different format. This simple deployment descriptor only points to the services of the first valid composition.
- *Catalog* (for ActiveBPEL): The catalog file lists all references to the WSDL files used by the process.
- *Endpoints File* (for Theseus/TEXO): The endpoints file lists the service candidates for all valid service compositions. When the process is deployed on a suitable process engine, this information can be used to bind or replace services at runtime.

Semantic Context Modeling and Service Descriptions

Another important extension of ISE is its support to annotate services semantically with the incorporation of context information emerging from service environment. Services need to operate in a knowledge-intensive environment that, in turn, affects the service provisioning and procurement process. The information captured from the environment is also known as context. The techniques that enable the exploitation of contextual information in services are generally known as "context handling" techniques, while the use of context to provide relevant information and/or services to the user, where relevancy depends on the users task, is known as "context-awareness". Context handling is of vital importance for developers and service architects since it provides dynamic service behaviour, content adaptation and simplicity in service usage.

Let us consider service price as an example. Due to the dependency on many context dimensions, it is hard to determine a fixed price for a service, especially at the time of service design. This is mostly regarded as price discrimination in the business literature (Lehmann & Buxmann, 2009), where the determination of price can be based on relevant information such as user's location, service agreement, usage data, temporary discounts, surcharges, etc., which we regard as context in this work. In such a setting, different price values can show up by the emergence of dynamic context data. Therefore, it becomes a challenging issue to obtain a consistent service description – e.g. to specify what is the price of a service – with the incorporation of this context data.

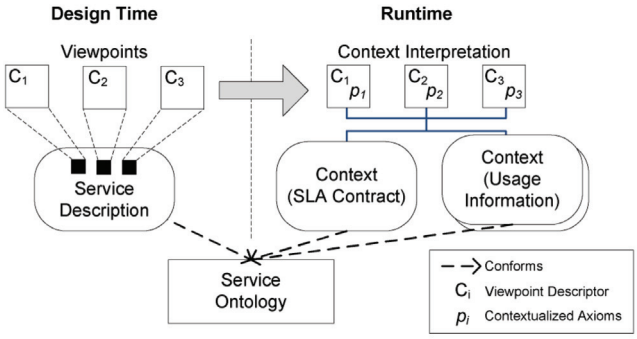
Figure 11 illustrates our approach for interpreting context information within semantic service description. All collected information conforms to a service ontology which is explained in Section "Service Ontology". Semantic IoS-based service description includes static service information (e.g. service name, provider, parameters, etc.) as well as viewpoints that are defined at design time to incorporate different perceptions of service based on possible contexts. At runtime, emerging context information is interpreted by these viewpoints to incorporate specific views during runtime procedures – e.g. service discovery, agreement, or execution.

Service Ontology

In order to capture both service descriptions and context information semantically, we rely on the service ontology which was previously introduced in detail in (Oberle, Bhatti, Brockmans, Niemann, & Janiesch, 2009). It provides a consistent and holistic way of capturing information by defining different aspects of a service and service related concepts as well as any relevant information that emerges as context.

Figure 12 presents an excerpt of service ontology for the purposes of this chapter (see (Oberle, Bhatti, Brockmans, Niemann, & Janiesch, 2009) for a detailed description). The concepts in the upper part are based on the DOLCE founda-

Figure 11. Overview of viewpoints and interpretation of context information



tional ontology (Oltramari, Gangemi, Guarino, & Masolo, 2002) providing us with a generic set of concepts and relations as well as ontology design patterns. Based on this upper part, several concepts are introduced to prescribe service information common to every service (e.g., service description, provider or parameters such as quality of service). This allows to capture service descriptions as a set of axioms within a knowledge-base (KB). For example, the MDH service in our scenario has the following ontological (assertion) axioms based on the concepts and relations in the service ontology:

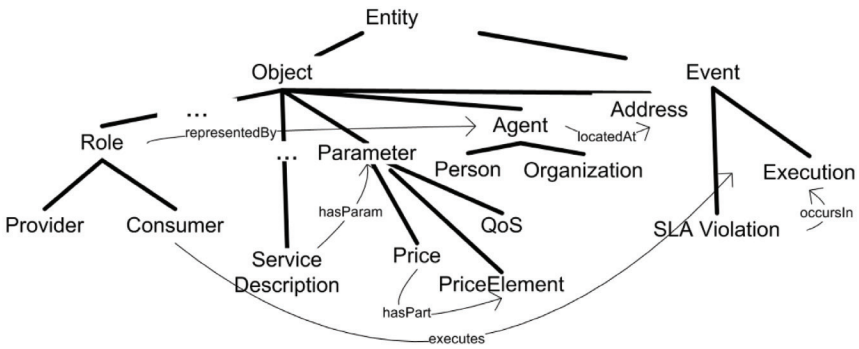
ServiceDescription(#MDH),
hasParam(#MDH;#MDH_{Price}),
provides(#CompA;#MDH)

Similarly, context information is captured as axioms based on the ontology. Although, from an ontological point of view, there is no distinction between the axioms describing services and context, in the course of service offering, context information dynamically emerges from various sources and is incorporated into KB as depicted in Figure 11. For example, information about service consumers' profile, or service contracts of a consumer for particular services can all be obtained from, e.g., Service Management Platform (see Figure 6) and be represented as axioms similar to the following:

ServiceConsumer(#$BusU_A$),
hasAddress(#$BusU_A$;#Germany)
hasSLA(#$BusU_A$;#SLA_1),
serviceType(#SLA_1;#Premium)

The central notion about using a service ontology to maintain all this information is to address

Figure 12. An excerpt of service ontology



the information integration challenge that emerges from the existence of several components in the design and offering of IoS-services. However, dynamic context information in the KB may result in different interpretations of a service description that requires the introduction of viewpoints as elaborated in the followings.

Viewpoints and Rules

According to the context information collected in the KB based on the service ontology, we can determine the subparts of service description, e.g. service price, tax rates, discounts etc., by using ontology-based rules. However, since all these information is managed in one KB for our service management platform, we need to create different viewpoints for different users in order to provide individualized values based on context. For example, to associate the German tax value for the business units in Germany, the following rule was defined for the MDH service:

C1: ServiceConsumer(?c) ^ hasAddress (?c,#Germany) ^ consume(?c,#MDH)

^ hasParam(#MDH, ?p) ^ Price(?p) → hasPart(?p,#GermanTax)

Similar rules can also be specified to associate further contextual information with the service descriptions such as offering discounts for SLA violations or different price values for standard and premium contracts. What is crucial in our approach is that every rule is associated with a viewpoint identifier (e.g. C1) that parameterizes the result of a rule into different viewpoints. Context modeler utilizes a reasoning mechanism defined in (Baader, Knechtel, & Penaloza, 2009) at the backend to generate different, e.g., pricing schemes for the same service.

SLA Management of Composite Services

ISE supports composite service providers when creating service compositions using existing services from the TEXO service marketplace. This is achieved by the functionality provided by the process modeling tools as shown in Figure 6. Atomic services are composed to collaboratively achieve tasks of higher complexity. The execution of atomic and composite services is regulated by service level agreement (SLA). SLAs regulate the tasks of the service, required and provided resources (i.e. what the service requires to execute and what it provides as result), different quality of service (QoS) and legal aspects, and start and end times of a service.

We developed an approach to support composite service providers to manage interdependencies between services in service compositions which they are offering. The approach is based on the assumption that information regarding dependencies between services is implicitly contained in the composite service process description and the SLAs negotiated between the composite service provider, atomic service providers, and the service consumer. We will now outline the approach and its integration into the ISE Workbench and illustrate its use based on the MDH scenario.

Dependency Management Approach

In order to manage the dependencies in service compositions, we developed an approach which captures dependencies between services in a dependency model. The model contains information about the different services involved in a service composition, the SLAs negotiated for the atomic services, the service composition, and a detailed description of the different dependencies between the dependant that depends on one or more antecedents (Winkler & Schill, 2009). This model is created at design time by a semi-automatic approach. At runtime it is used to

Table 2. Service dependencies of the MDH scenario

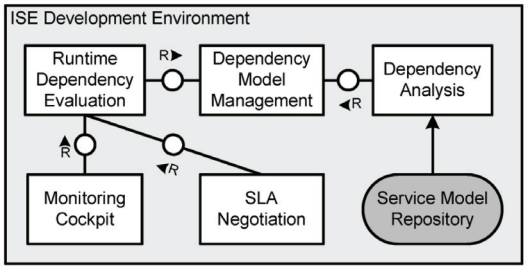
Antecedent	Dependency	Dependant
Backup Data	finish-to-start	Replace Hardware
Replace Hardware	finish-to-start	Restore Data

support the composite service provider to handle the dependencies.

The lifecycle associated with dependency model consists of four phases. During the creation and re computation phase the dependency model is created based on the composite service process description and SLA information. The created dependency model can be extended manually with dependency information, which cannot be detected automatically. The model needs to be recalculated if conflicts are detected with respect to the dependencies, SLAs change, or the process description is adapted. In the MDH scenario different time dependencies are discovered (see Table 2). The Backup Data service needs to finish before Replace Hardware can start. Replace Hardware needs to finish before Restore Data can start.

The validation phase is necessary to ensure that the created dependency model is valid. It is also necessary to validate the negotiated SLAs, which can be supported by the dependency model. In the case that problems are detected the model needs to be re-computed. In our scenario it is necessary to schedule the different services according to the dependency model, i.e. the backup of data is scheduled before the replacement of hardware and the restoring of data afterwards. During the usage phase, the dependency model supports runtime dependency evaluation tasks such as the determination of SLO (Service Level Objective) violation effects or handling SLA renegotiation requests. In the case of renegotiation, the model may need to be adapted accordingly. In our scenario company C needs to renegotiate the scheduled data for hardware replacement due to availability problems of the hardware. The request is evaluated based on the dependency model and a conflict is detected with the scheduled time for the service for restoring data. Thus, a new slot needs to be arranged for restoring the data. During the retirement phase, the dependency model is discarded when is not used or referenced.

Figure 13. Architecture for SLA dependency management



Architecture and Integration with ISE

The functionality for the handling of dependencies is provided by three main components, which are integrated into the ISE workbench (see Figure 13). They implement the lifecycle presented above. The Dependency Analysis component is used for the semi-automatic dependency analysis at design time and the recomputation of the dependency model at runtime, i.e. the first phase of the lifecycle. For the creation of the dependency model, the BPMN process description and the SLA information for the different services are analysed. Both are requested from the Service Model Repository. Temporal relationships between services are detected based on the process description. Resource and location dependencies are discovered based on the negotiated SLAs. QoS and price dependencies are calculated based on SLA information as well as the composite service process structure. While various dependencies can be discovered automatically, there is a need for extending the generated dependency model with information

which cannot be discovered. This is achieved by a dependency model editor, which is part of the Dependency Analysis component. Upon changes to the SLAs related to the composite service and the business process itself, the dependency model needs to be re-computed using the semi-automatic approach presented.

The Dependency Model Management component manages different instances of dependency models and is responsible for model creation, storage, retrieval, and removal. It is integrated with the Dependency Analysis and Runtime Dependency Evaluation components to support their work at design time and runtime. Furthermore, the validation of dependency models and the associated SLAs is realized by the Dependency Model Management component. It assures that only validated dependency models are used for runtime evaluation. It also detects conflicts between different SLAs (e.g. start/end time) based on the dependency model. Thus, while supporting the Dependency Creation & Recomputation and Usage phases, it realizes the Validation and Retirement lifecycle phases.

The Runtime Dependency Evaluation component implements the Usage phase. It uses the dependency model at runtime to evaluate the dependencies that take effect e.g. when a SLA shall be renegotiated. The evaluation of dependencies is triggered by the SLA Negotiation component upon SLA renegotiation requests. The dependency evaluation can also be initiated by the Monitoring Cockpit upon receiving information about SLO violations.

RELATED WORK

WSMF, WSML, WSMT, and WSMO provide frameworks, tools and an integrated modeling environments (see (Kerrigan, 2005) and (Paolucci & Wagner, 2006)) to describe semantic Web services. Compared to ISE, these approaches concentrate their attention on the use of ontologies to enhance the expressiveness of descriptions of technical Web services and their interfaces (i.e. WSDL). While ISE also relies on ontologies, their use is not limited to the interfaces of services and can be also used to increase the expressiveness of the organizational and IS models that can be found, for example, in the business rule and human resource aspects.

SoaML (Sadovykh, Hahn, Panfilenko, Shafiq, & Limyr, 2009), MIDAS (Lopez-Sanz, Acuna, Cuesta, & Marcos, 2008), and UML-S (Dumez, Gaber, & Wack, 2008) also follow an MDA approach for service modeling but target the development of SOA-based solutions and Web information systems. In contrast to ISE, these approaches rely uniquely on UML models and UML extensions for service modeling. The inexistence of organizational and IS perspectives, and the purely UML-based approach difficult the participation of business stakeholders (e.g. CEO, CTO, CIO) when defining IoS- based services. Furthermore, advanced modelling mechanisms, such as business process design based on patterns and context-based modeling were not yet explored. One interesting aspect of UML-S is the provision of transformation rules between UML-S and adapted Petri nets to solve to verify and validate the models created. ISE relies on the use of CCS (Milner, 1995) since it has proven to provide a suitable means for modeling business processes.

Commercial applications that target the use of multiple models to design services or SOA-based architectures are currently available from several companies. For example, Select Architect[4], Business Architect[5], and Enterprise Architect[6] typically rely on business motivation modeling, business process modeling, component-based models, and corporate data models to design IS/IT. While they rely on MDA approaches for code generation, they lack precise mapping and synchronization techniques between models. Furthermore, since these tools mainly target the design of IS/IT solutions, and do not directly target business services, important aspects of services such as

pricing models and marketing channels models are not available.

CONCLUSION

In this chapter, we presented ISE framework and its three advanced extensions to meet the requirements emerging from the inherent complexity of IoS-based services. ISE framework utilizes separation of concerns and model-driven techniques to overcome the inherent complexity in a service engineering process. The process pattern matching approach provides a semi automatic means to identify suitable services for the assignment to particular business tasks while constructing executable service compositions. Furthermore, semantic context modeling and service description extension enables an ontology-based approach to specify the service context and descriptions and to define dynamic service properties by incorporating the changes in context. Finally, the SLA management approach supports service providers to manage dependencies between the services in their composition to assure a proper execution. Future work includes further case studies to improve the modeling experience and to gather requirements from different business service domains.

REFERENCES

Alves, A., Arkin, A., Askary, S., Barreto, C., Bloch, B., Curbera, F., et al. (2007). *Web services business process execution language, version 2.0 (OASIS Standard)*. WS-BPEL TC OASIS. Retrieved from http://docs.oasis-open.org/wsbpel/2.0/wsbpel-v2.0.html

Baader, F., Knechtel, M., & Penaloza, R. (2009). *A generic approach for large-scale ontological reasoning in the presence of access restrictions to the ontology axioms* (p. 49).

Baida, Z., Gordijn, J. & Omelayenko, B. (2004). A shared service terminology for online service provisioning.

Barros, A., & Dumas, M. (2006). The rise of Web service ecosystems. *IT Professional*, 31–37. doi:10.1109/MITP.2006.123

Bellwood, T., Clement, L., Ehnebuske, D., Hately, A., Hondo, M., & Husband, Y. L. (2002). *UDDI Version 3.0. Published specification*. Oasis.

Blau, B., Kramer, J., Conte, T., & van Dinther, C. (2009). Service value networks. *Proceedings of the 11th IEEE Conference on Commerce and Enterprise Computing*.

Booth, D., Haas, H., McCabe, F., Newcomer, E., Champion, M., Ferris, C., et al. (2004). Web Services Architecture. *W3C Working Group Note, 11*(1).

Briscoe, G., & De Wilde, P. (2006). Digital ecosystems: Evolving service-orientated architectures. In *Bio-inspired models of network, information and computing systems,* (pp. 1-6).

Bullinger, H. (2003). Service engineering–methodical development of new service products. *International Journal of Production Economics*, 275–287. doi:10.1016/S0925-5273(03)00116-6

Cardoso, J., & Sheth, A. (2003). Semantic e-workflow composition. *Journal of Intelligent Information Systems*, *21*, 191–225. doi:10.1023/A:1025542915514

Cardoso, J., Voigt, K., & Winkler, M. (2008). *Service engineering for the Internet of Services* (pp. 17–25). Springer.

Cardoso, J., Voigt, K., & Winkler, M. (2009). *Service engineering for the Internet of Services* (pp. 15–27). Berlin, Heidelberg: Springer.

Cardoso, J., Winkler, M., & Voigt, K. (2009). A service description language for the Internet of Services. *Proceedings First International Symposium on Services Science (ISSS'2009)*. Berlin: Logos Verlag.

Chang, E., & West, M. (2006). Digital ecosystems a next generation of the collaborative environment. *Proceedings from the Eight International Conference on Information Integration and Web-Based Applications & Services*, (pp. 3-24).

Christensen, E., Curbera, F., Meredith, G. & Weerawarana, S. (2001). *Web Services Description Language (WSDL) 1.1.*

Czarnecki, K., & Helsen, S. (2006). Feature-based survey of model transformation approaches. *IBM Systems Journal*, *45*(3). doi:10.1147/sj.453.0621

Dogac, A., Kabak, Y., Laleci, G. B., Mattocks, C., Najmi, F., & Pollock, J. (2005). Enhancing ebXML registries to make them OWL aware. *Distributed and Parallel Databases*, *18*, 9–36. doi:10.1007/s10619-005-1072-x

Dumez, C., Gaber, J., & Wack, M. (2008). *Model-driven engineering of composite web services using UML-S* (pp. 395–398). ACM.

Fleischmann, A. (1994). *Distributed systems: Software design and implementation*. Springer.

Fuger, S., Najmi, F. & Stojanovic, N. (2005). *ebXML registry information model, version 3.0.*

Herzum, P., & Sims, O. (2000). *Business component factory*. New York: John Wiley.

Janiesch, C., Niemann, M., & Repp, N. (2009). Towards a service governance framework for the Internet of Services. *Proceedings of the 17th European Conference on Information Systems.* Verona, Italy.

Kerrigan, M. (2005). *Web Service Modeling Toolkit (WSMT)*. Techreport.

Kett, H., Voigt, K., Scheithauer, G. & Cardoso, J. (2009). *Service engineering for business service ecosystems*.

Kleppe, A., & Warmer, J. (2003). *MDA rxplained. The model driven architecture: Practice and promise*. Addison-Wesley.

Kopecky, J., Vitvar, T., Bournez, C., & Farrell, J. (2007). SAWSDL: Semantic annotations for WSDL and XML schema. *IEEE Internet Computing*, *11*, 60–67. doi:10.1109/MIC.2007.134

Lehmann, S., & Buxmann, P. (2009). Pricing strategies of software vendors. *Journal of Business and Information Systems Engineering*, *6*, 1–10.

Lopez-Sanz, M., Acuna, C. J., Cuesta, C. E., & Marcos, E. (2008). *Defining service-oriented software architecture models for a MDA-based development process at the PIM level* (pp. 309–312). IEEE Computer Society.

Milner, R. (1995). *Communication and concurrency*. Prentice Hall PTR.

Moser, O., Rosenberg, F., & Dustdar, S. (2008). Non-intrusive monitoring and service adaptation for WS-BPEL. *Proceedings of the World Wide Web Conference,* (pp. 815-824). New York: ACM.

Oberle, D., Bhatti, N., Brockmans, S., Niemann, M., & Janiesch, C. (2009). Countering service information challenges in the Internet of Services. *Business and Information Systems Engineering*, *1*, 370–390. doi:10.1007/s12599-009-0069-9

Oltramari, A., Gangemi, A., Guarino, N., & Masolo, C. (2002). *Sweetening ontologies with DOLCE*. Springer.

Paolucci, M., & Wagner, M. (2006). *Grounding OWL-S in WSDL-S* (pp. 913–914). IEEE Computer Society.

Papazoglou, M. P., Traverso, P., Dustdar, S., Leymann, F., & Kramer, B. J. (2008). Service-oriented computing: A research roadmap. *International Journal of Cooperative Information Systems*, *17*, 223–255. doi:10.1142/S0218843008001816

Peneder, M., Kaniovski, S., & Dachs, B. (2003). What follows tertiarisation? Structural change and the role of knowledge-based services. *The Service Industries Journal*, *23*, 47–66. doi:10.1080/0264206041233130088

Preist, C. (2004). *A conceptual architecture for semantic Web services* (pp. 395–409). Springer.

Riedl, C., Bohmann, T., Leimeister, J. M., & Krcmar, H. (2009). A framework for analysing service ecosystem capabilities to innovate. *Proceedings of 17th European Conference on Information Systems*.

Sadovykh, A., Hahn, C., Panfilenko, D., Shafiq, O., & Limyr, A. (2009). *SOA and SHA tools developed in SHAPE project* (p. 113). University of Twente.

Sampson, S. & Froehle, C. (2006). Foundations and implications of a proposed unified services theory. *Production and Operations Management*.

Scheithauer, G., Voigt, K., Bicer, V., Heinrich, M., Strunk, A. & Winkler, M. (2009). *Integrated service engineering workbench: Service engineering for digital ecosystems*.

Studer, R., Grimm, S., & Abecker, A. (2007). *Semantic Web services: Concepts, technologies, and applications. New York*. Secaucus, NJ: Springer-Verlag Inc.

Tapscott, D., Ticoll, D., & Lowy, A. (2000). *Digital capital: Harnessing the power of business Webs*. Harvard Business School Press.

Teboul, J. (2005). *Service is in front stage.*

White, S.A. (2004). Introduction to BPMN. *IBM Cooperation*, 2008-029.

Winkler, M., & Schill, A. (2009). *Towards dependency management in service compositions* (pp. 79–84).

Zachman, J. A. (1987). A framework for information systems architecture. *IBM Systems Journal*, *26*, 276–292. doi:10.1147/sj.263.0276

ENDNOTES

1 http://www.eclipse.org/
2 http://www.jcom1.com
3 http://www.cs.sunysb.edu/_cwb/
4 http://www.selectbs.com/adt/analysis-and-design/select-architect
5 http://www.ids-scheer.com/en/Software/ARISSoftware/ARISBusinessArchitect/3731.html
6 http://www.sparxsystems.com.au/