Grant Agreement N° 215483

Title:	*Models and Mechanisms for Coordinated Service Compositions*
Authors:	*USTUTT, UOC, UPM, VUA, UCBL, TUW*
Editor:	*Martin Treiber (TUW)*
Reviewers:	*Harald Psaier (TUW), Manuel Carro (UPM)*
Identifier:	*CD-JRA-2.2.2*
Type:	*Deliverable*
Version:	*1.2*
Date:	*16 March 2009*
Status:	*Final*
Class:	*External*

Management Summary

This deliverable describes the research roadmap and initial research work in the context of models and mechanisms for coordinated service compositions. It provides the foundations for the research in the WP JRA-2.2 by establishing a preliminary framework for QoS-aware adaptable service compositions. We present initial research results in some areas of this framework, in particular on models of service compositions, top-down development, and monitoring and adaptation of service compositions. The work will be continued and extended in the follow-up deliverables.

The research leading to these results has received funding from the European Community's Seventh Framework Programme FP7/2007-2013 under grant agreement n° 215483 (S-Cube).

Members of the S-Cube consortium:

University of Duisburg-Essen (Coordinator)	Germany
Tilburg University	Netherlands
City University London	U.K.
Consiglio Nazionale delle Ricerche	Italy
Center for Scientific and Technological Research	Italy
The French National Institute for Research in Computer Science and Control	France
Lero - The Irish Software Engineering Research Centre	Ireland
Politecnico di Milano	Italy
MTA SZTAKI – Computer and Automation Research Institute	Hungary
Vienna University of Technology	Austria
Université Claude Bernard Lyon	France
University of Crete	Greece
Universidad Politécnica de Madrid	Spain
University of Stuttgart	Germany
University of Hamburg	Germany
Vrije Universiteit Amsterdam	Netherlands

Published S-Cube documents

All public S-Cube deliverables are available from the S-Cube Web Portal at the following URL:

http://www.s-cube-network.eu/results/deliverables/

The S-Cube Deliverable Series

Vision and Objectives of S-Cube

The Software Services and Systems Network (S-Cube) will establish a unified, multidisciplinary, vibrant research community which will enable Europe to lead the software-services revolution, helping shape the software-service based Internet which is the backbone of our future interactive society.

By integrating diverse research communities, S-Cube intends to achieve world-wide scientific excellence in a field that is critical for European competitiveness. S-Cube will accomplish its aims by meeting the following objectives:

- Re-aligning, re-shaping and integrating research agendas of key European players from diverse research areas and by synthesizing and integrating diversified knowledge, thereby establishing a long-lasting foundation for steering research and for achieving innovation at the highest level.
- Inaugurating a Europe-wide common program of education and training for researchers and industry thereby creating a common culture that will have a profound impact on the future of the field.
- Establishing a pro-active mobility plan to enable cross-fertilisation and thereby fostering the integration of research communities and the establishment of a common software services research culture.
- Establishing trust relationships with industry via European Technology Platforms (specifically NESSI) to achieve a catalytic effect in shaping European research, strengthening industrial competitiveness and addressing main societal challenges.
- Defining a broader research vision and perspective that will shape the software-service based Internet of the future and will accelerate economic growth and improve the living conditions of European citizens.

S-Cube will produce an integrated research community of international reputation and acclaim that will help define the future shape of the field of software services which is of critical for European competitiveness. S-Cube will provide service engineering methodologies which facilitate the development, deployment and adjustment of sophisticated hybrid service-based systems that cannot be addressed with today's limited software engineering approaches. S-Cube will further introduce an advanced training program for researchers and practitioners. Finally, S-Cube intends to bring strategic added value to European industry by using industry best-practice models and by implementing research results into pilot business cases and prototype systems.

S-Cube materials are available from URL: http://www.s-cube-network.eu/

Contents

Chapter 1

Introduction

The goal of work package JRA-2.2 is to establish the foundation of QoS-aware adaptable service compositions. QoS-aware adaptable service compositions adapt as reaction to changes in the QoS characteristics of SBAs. The QoS characteristics of an SBA are specified on all three SBA functional layers. This means that the service compositions and their adaptation are influenced by all these characteristics in a combination. The work in this WP is hence relying on input and QoS requierements from the BPM and Service Infrastructure layers and addresses them them throughout the service composition lifecycle, including modeling and verification, execution, monitoring and adaptation. This deliverable presents the initial roadmap towards organizing the research in this WP and first research results for models and mechanisms of QoS-aware adaptable service compositions. The research work on the foundations of QoS-aware adaptable service compositions will be refined and extended in the follow-up deliverables.

The document organizes the discussion of the planned research work according to the life cycle of service compositions, which is a part of the life cycle of SBAs in general. We first deal with modeling of QoS-aware service compositions establishing formal models for service compositions. We then show how QoS-aware adaptable compositions can be developed and executed based on corresponding languages. Finally, we deal with monitoring and adaptation of service compositions.

As we take into account the requirements from the business layer and requirements and influences of the service infrastructure layer, we are closely collaborating with the other two workpackages in JRA-2. An additional goal is to identify the potential contribution of this work package to the engineering and design principles work package JRA-1, both by giving input on engineering and design principles considering usage of our models and mechanims for service compositions and providing those models and mechanisms.

The deliverable is structured as follows. In Section 2 we present the research roadmap of the whole work package JRA-2.2 giving a short overview of the way we structure the work according to service composition life cycle and the challenges we want to tackle. We also discuss relations of our work to other work packages. In the following sections we then present these challenges in more detail and present initial research results. Section 3 presents our first results and findings on service composition models. Section 4 deals with service composition languages and development approaches for service compositions. In Section 5 we present our current work on monitoring, analysis, and adaptation of service compositions.

Chapter 2

Research Roadmap Overview

The subject of research in this work package are QoS-aware adaptable service compositions. In this section, we will present how we organize the research work in this WP and identify approaches and mechanisms needed to address the open issues in the field of service compositions and in the context of S-Cube. We will use this organization as a guideline for the research work and refine it in the follow-up deliverables of this WP. We begin by describing the established service composition lifecycle which will be used as a basis for the following discussion on research challenges.

2.1 Lifecycle of Service Compositions

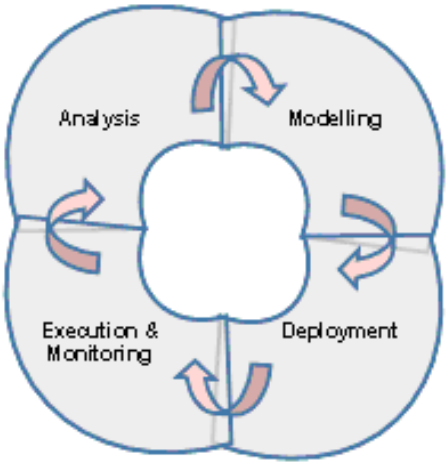


Figure 2.1: Lifecycle of Service Compositions

In this document we use a simplified service composition life cycle containing four major phases. This life cycle is not a complete one and is used for organizing the discussion; it by no means reflects all possible life cycle definitions used in the SoC and BPM communities. The life cycle will be adjusted to the research results produced by the integration activities in S-Cube and the work in JRA-1, which deal with engineering principles for SBAs in an integrated manner.

Service compositions follow four major phases during the course of their existence:

1. Modelling phase - a.k.a. build or design time (definition from the SOC and BPM communities): during this phase the service compositions are created either automatically, generated from another representation of the composition or developed from scratch

by a service composition developer. The output is the executable representation of the service composition.

2. Deployment phase - the executable service composition is made available for execution on an execution environment. This phase includes configuration of the service composition. The concrete steps to be performed and the information needed to configure the composition depend on the service composition language used and on the execution environment on which it is to be deployed.

3. Run time and monitoring: service compositions are executed during this phase. While executed they are also monitored to allow tracking of the status of the composition, the data and resources it utilizes, and process metrics. The monitoring data may also be used for run-time analysis purposes, where analysis may be done during the execution of the compositions or after that, using the so-called audit trail or execution history. The analysis during the execution of service compositions may by used to support decisions to adapt the composition or parts of it.

4. Analysis phase - during this phase the execution history of the service composition is used for analysis. The analysis may involve process mining, fault pattern detection, resource consumption and throughput analysis. All the results of the analysis phase can be used for the purpose of improving/optimizing the quality of the service composition with respect to various criteria. The result of the phase are recommendations for adaptation of service compositions in general or per case (instance, e.g. a concrete customer order, a specific item processing, etc.). The adaptation recommendations may be produced and may be enforced during both the modelling and run time phases.

2.2 Summary of Research Objectives for QoS-aware Adaptable Service Compositions

We will explain the position of the service composition research in the scope of JRA-2 based on the layering shown in Figure 2.2. The interactions with the other two work packages in JRA-2 are also presented.

When using a top-down approach for developing service compositions, the modeling phase comprises the creation of a service composition based on the input from the upper level in the architecture of SBAs (the BPM layer). The input is provided in the form of business process models in a notation and format specified in the work on the BPM work package. The business process models comprise high-level orchestrations and choreographies, which must be transformed into executable service compositions. Such preliminary approaches already exist, based on existing service composition models, however the QoS-awareness (*QoS* not constrained only to technical QoS but including also quality characteristics on process (process performance metrics) and business layer (key performance indicators)) of such compositions is not supported.

In the existing models for services and service compositions, QoS awareness in combination with the creation of service compositions and their adaptation are not completely addressed. The available languages for executable compositions also abstract away or consider in isolation the QoS characteristics of the compositions as a whole and of the individual services. Validation and verification of the service compositions exploiting the QoS characteristics are hence hampered. The derivation of QoS characteristics for service compositions and the composed services is not yet enabled, i.e. the following two kinds of transformation on conceptual level are not yet addressed: (a) the transformation from the BPM level requirements (KPIs) to the service composition requirements and (b) derivation of QoS characteristics of individual service from the overall QoS description of the composition.

The service compositions generated or created manually based on the input from the BPM level will be executed on a service composition execution environment. Due to the identified partner expertise, the methodology and approaches will be shown to work for the process based approach for service compositions and hence will be executable on a process execution environment or a process engine. Nevertheless, we will identify the necessary functions an execution environment must possess in order to execute QoS-aware adaptable service compositions, regardless of the implementation technology. One major feature of such an environment we intend to focus on is the support for monitoring and adaptation mechanisms.

Monitoring of service compositions in turn will enable the analysis of service compositions and is a necessary prerequisite for identifying the need for adaptation of service compositions. For the purpose of monitoring, we will deal with monitoring across layers taking into account key performance indicators (KPIs) on business level, process performance metrics (PPMs) on service composition level, technical QoS metrics on service infrastructure level, and their relations. On top of the monitoring framework, we will devise new analysis techniques which enable dependency analysis between the metrics on the different layers. The monitoring and analysis framework will be integrated with adaptation mechanisms. Mechanisms for adaptation of service compositions will be identified and classified and will be demonstrated for the process-based implementation approach. The triggers for the adaptation of service composition will also be identified and classified and the respective subsequent adaptation reactions on behalf of service compositions recommended. Thereby, we will in particular deal with process fragmentation and pro-active adaptation based on monitoring and analysis results.

2.3 Summary of Targeted Research Results

Currently we have identified the following major groups of research challenges, which will be refined in the course of the project and in cooperation with the other WPs in S-Cube.

Formal Models and Languages for QoS-aware Service Compositions:

- Models of services and service compositions, including formal representation, incorporating QoS and behavioural features
- Languages for service compositions that reflect the above mentioned models
- Mechanisms for deriving QoS characteristics of compositions from QoS descriptions of the individual services and vice versa
- Mechanisms for decomposition of business performance characteristics (KPIs) to QoS characteristics of service compositions and services and vice versa

Monitoring and Analysis of QoS-aware Service Compositions:

- Mechanisms for monitoring performance characteristics of service compositions based on both process performance metrics on process level, and QoS metrics on service infrastructure level
- Mechanisms for cross-layer performance analysis and prediction, i.e., dependency analysis between KPIs, Process Performance Metrics, and QoS Metrics
- Integration of monitoring, analysis, and adaptation mechanisms

Adaptation of QoS-aware Service Compositions:

- Adaptation mechanisms for service compositions to react to different triggers, including those from the BPM and Service Infrastructure levels

- Mechanisms for pro-active adaptation based on monitoring and analysis results (in particular based on prediction)
- Mechanisms for fragmentation of service compositions to improve reusability and flexibility of SBAs

2.4 Relationship with Other Workpackages

In order to better understand how this deliverable is interrelated with other workpackages, we give here an account of these dependencies as an inverse index: for each workpackage for which we have identified a clear dependency / feedback, we state where in this deliverable that dependency appears. We hope that this to result in a better cohesion between workpackages and to contribute to an enhancement of collaboration and cross-fertilization possibilities.

2.4.1 WP IA-1.1 (Convergence knowledge model)

The present workpackage, as the rest of the S-Cube workpackage, contributes to the knowledge model with terms, definitions, and interrelationships between them.

2.4.2 WP JRA-1.1 (Engineering Principles)

Service engineering differs from traditional software engineering mainly due to their focus and aims [1, 2]. In particular, the focus of service engineering is shifted from engineering applications to composing pools of services; the control of software (services) is passed from their users to other owners (i.e. users of services do not have the control of them), and the aims are redirected from quality of software (e.g. performance, security, maintainability) to the ability to adapt to ever-changing requirements (e.g. flexibility, dynamicity).

These differences are reflected by a number of aspects that are crucial to communicate the service development process and cross-cut all the layers in service-based applications, including service compositions.

- Service aspect 1: Cross-organizational collaboration Cross-organizational collaboration is especially critical since multiple roles collaboratively develop service-based applications. The roles coexist in a service-based application rather than having an active-passive relationship (e.g. outsourcer and supplier). Collaboration becomes *white-box* in that it enters the details of a service development process that is now scattered across multiple partner enterprises. This makes their relationship tighter but also demanding clearer governance and agreements.
- Service aspect 2: Increased importance of identifying stakeholders Since cross-organizational collaboration becomes more critical, the importance of clearly identifying stakeholders increases accordingly. If stakeholders are identified at a very detailed level, the interaction represented in a development process model also becomes more elaborated. However, if stakeholders are identified at a too high level of granularity, the represented interaction remains not fully specified. This leads to unclear responsibilities among collaborating enterprises and thus decrease in trust and possibly in success. Because the level of details matters, the identification of stakeholders directly decides the level of detail expressed in a development process model.
- Service aspect 3: Increased effort at run-/change time

  The main goal of SBAs is not only to deliver high quality but also agile and robust services which are able to meet the *ever-changing* business requirements. Consequently, much more development effort is shifting from design time to run-/change time. For

instance, components identification is often performed at design time in TSE; the service engineering equivalent activity is service discovery, which is encouraged to be performed at runtime and it is regarded as one of the major challenges in the service engineering field.

These aspects are not only relevant to service composition but are also peculiar to service engineering in general. By analyzing these aspects and visualizing them in a service development process model, different service engineering approaches can be better interpreted and investigated. From the perspective of the development process, these aspects are part of the engineering knowledge that should be *explicitly *captured. This corresponds with knowledge models, which define service oriented computing processes and modeling, presented in S-Cube deliverable CD-JRA-1.1.2.

2.4.3 WP JRA-1.2 (Adaptation and Monitoring)

- Service composition inspired on biological models (Section 3.1.1) are also related with adaptation, since fitness can be used to measure how well the evolution of a composition interacts with the rest of the environment. The result of fitness functions can therefore be taken into account by adaptation layers in order to decide about this adaptation.

- Timed automaton (used in Section 3.4.1 to represent orchestrations and choreographies) include time constraints which restrict when state changes can happen. It is possible to used these constraints to derive sound upper and lower bounds for *clocks*[1] in each state of the system. Checking these bounds can be taken care of by actual code which is generated in an at least systematic, if not semi-automatic way. This will help monitoring mechanisms to trigger the appropriate adaptation / readjusting / alarm procedures in case deviations larger than admissible are detected.

- The proposed approaches to monitor service compositions (Section 5.1) naturally need to establish a link with the general, project-wide monitoring techniques, and use the mechanisms therein proposed in order to assess the quality of the running compositions.

2.4.4 WP JRA-1.3 (End-to-End Quality Provision and SLA Conformance)

The Quality Reference Model (S-Cube deliverable CD-JRA-1.3.2) serves as classification of QoS attributes for whole S-Cube framework and, in particular, for QoS-aware service composition.

- The *fitness* notion of biologically-inspired models (Section 3.1.1) is a parameter which can have a complex definition adapting to the different points of view of stakeholders. In this respect, some definitions of fitness can be very similar to, or be greatly influenced by QoS characteristics.

- The semantics of time constraints in Section 3.4.1 very similar to that of clocks in a timed automaton does —and, in fact, the former can be mapped onto the latter. These time constraints could be used to reflect some conditions (i.e., maximum or minimum bounds) on one of QoS attributes presented in CD-JRA-1.3.2, *time behavior*.

- As mentioned previously (Section 2.4.3), lower and upper bounds on clocks can be used to instrument the composition so that alarms are triggered. Alternatively, by measuring how well a systems behaves (time-wise) with respect to its initial design, a degree of conformance to this design can be worked out. This conformance can also be seen as a measure of efficiency compliance (also in CD-JRA-1.3.2).

[1] Which customarily represent time.

- The work on replaceability (Section 3.4.3) and interoperability (Section 3.4.3) also gives an initial technique to automatically decide on replaceability, thereby linking directly with the quality attribute presented in CD-JRA-1.3.2. While in the present state of work only coarse qualitative (yes, no, perhaps) answers to replaceability are being given, expanding our approach to also generate measures of replaceability is in progress.
- The service composition approach based on refinements from WS-CDL and WSDL to WS-BPEL and WS-Policy (Section 4.1) explicitly takes into account QoS characteristics, and therefore contributes to JRA-1.3 in maintaining end-to-end quality.

2.4.5 WP JRA-2.1 (Business Process Management)

- Some fitness measures of service compositions (Section 3.1.1) can in principle be defined to approximate KPIs, and thus they can be used to measure the overall ability of some service composition to fulfill business goals.
- The formalism in Section 3.4.1 represents quite closely a business process, and can be intuitively understood as such by practitioners in the field. However it can also be used to check for compliance and, to some extent, to guide realizations of service compositions aiming at implementing such a business process.
- One challenge in the BPM area is to adequately represent transactions, which can be trans-organizational and very complex. The proposals on transactions and transaction patterns (Section 3.3.2) can be used, to some extent, to answer to this need and provide a means to map these high-level goals onto to composition layer.
- The analysis and monitoring proposals put forward in Section 5.1 explicitly aim at taking into account the different metrics (KPIs, PPMs) of the BPM layers and monitoring, therefore implementing cross-layer monitoring.

2.4.6 WP JRA-2.3 (Self-* Service Infrastructure and Service Discovery Support)

- Part of the actual data monitoring needed by Section 5.1 in order to ensure that some service composition is faithful to its design has to be gathered from runtime characteristics and the profile of the actual execution which are available only the infrastructure level. This establishes a strong relationship between workpackages JRA-2.2 and JRA-2.3.

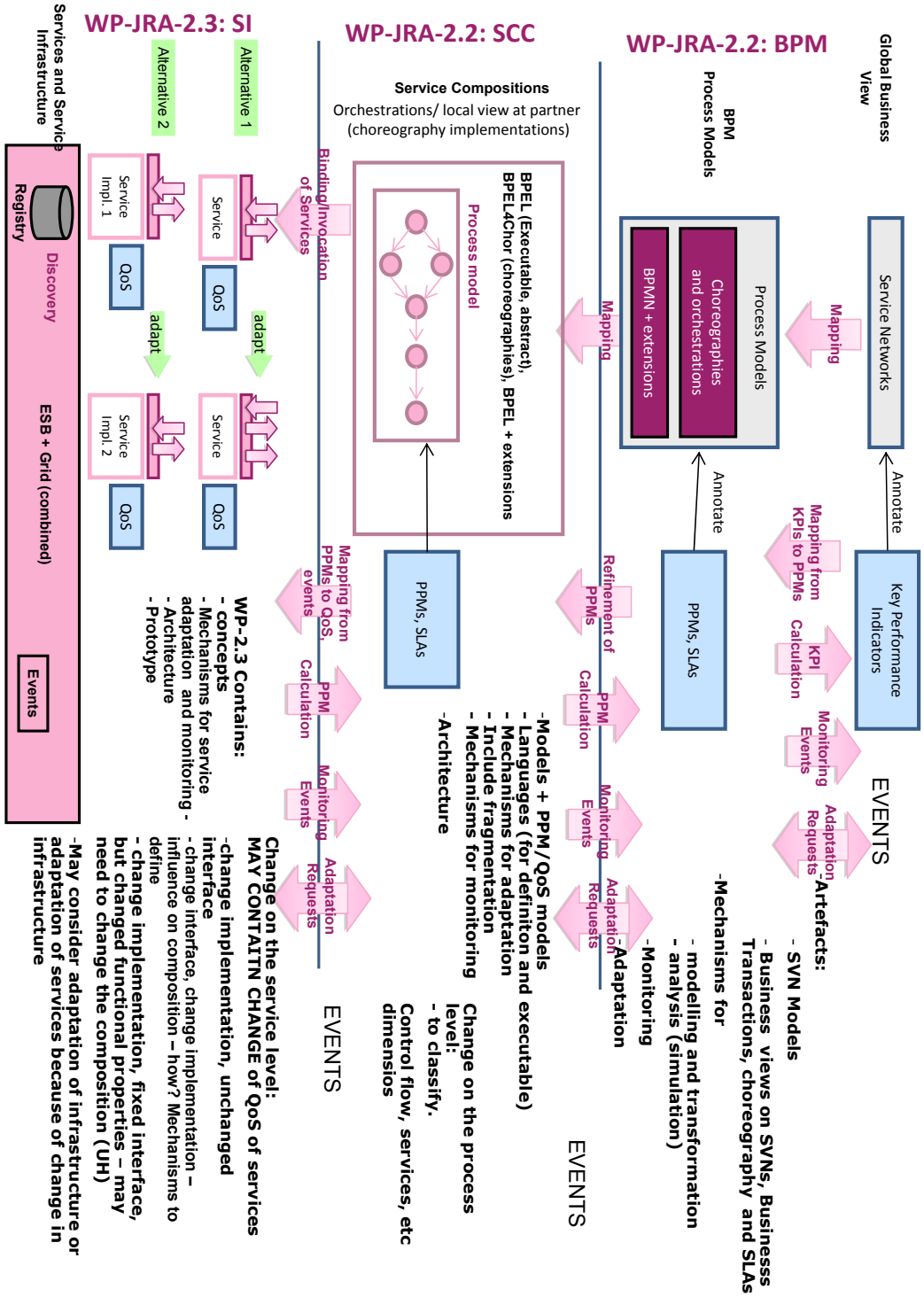


Figure 2.2: JRA-2 Layering

Chapter 3

Service Composition Models

3.1 Service Models

In this section, we discuss two different approaches for the modeling of services, (i) biological inspired service models and (ii) semantic service models, that both can serve as foundation for the service life cycle introduced in Section 2. Both models extend the current service model with additional descriptions that provide means for additional meta information.

Biological inspired service models focus on dynamic aspects of services and study the evolution of services. The goal is to understand factors that influence the service during its life cycle and to analyze the impacts of service related changes in the context of service ecosystems.

Semantic service models make use of rich meta data, like ontologies, to extend existing service descriptions to allow for reasoning on services. With this additional, well structured and semantically enriched data, adaptation processes can be supported, without the need for human intervention on a technical level.

3.1.1 Biologically Inspired Service Models

Services are subject to changes during their life cycle. These changes, or adaptions, have different causes, like the change of service requirements, change in the service usage, etc., as discussed in [3]. We can observe various types of changes, for instance QoS changes, interface modification or changes in the overall semantics of services [4]. These change processes can be regarded as *service evolution*, since these processes have some similarities with the evolution in a biological sense. As discussed in [5], there is no exact mapping of biological terms to services. Instead we focus on a core set of concepts that we borrow biological evolution to explain service evolution. Like in biology, we intend to consider services as individuals that are able to reproduce itself in a service ecosystem. Unlike living beings, services do not actively procreate, but depend on external mechanisms that allow the procreation of services. In the context of service evolution, composition can be regarded as service procreation, since the result of a service composition is a new service that exists within the service ecosystem. A central aspect of services with regard to evolution is the notion of *fitness*. In a strict biological sense fitness refers to the possibility of survival of an individual in a certain environment. If we transform this idea to services, fitness defines the degree to which a service is adapted to its environment and how well it is adopted by users. This degree of service fitness can be defined with various metrics, an example is the classification by QoS attributes (response time, availability, cost, provided data quality, etc.) [6]. A deeper discussion of the semantics of the term evolution is out of the scope of this deliverable, for a detailed discussion about the terms, the interested reader is pointed to [7].

Another aspect that requires extensive investigation concerns the environment in which services operate [8]. We consider the environment as set of artifacts and stakeholders that

have impact on the service. In such environments, we can observe dependencies of services on other elements of the environment, ecosystem respectively. From a business perspective, we can observe KPIs that measure and evaluate how successful a service is and thus serve as input for service fitness. From a technical perspective, we can observe two categories of dependencies, (1) a service may depend on other services (e.g., a service calls another service as part of a service composition) and (2) a service may depend on other resources (e.g., database, computational resources). If these external - from a service point of view - resources change, we can observe also impacts on the service itself.

As mentioned above, complementary to technical dependencies, services also depend on the stakeholders interact in a service ecosystem [3]. Stakeholders are either persons or organizations that have interest in service. An example is an organization (e.g., non government organization, etc.) that offers information services to its customers. These stakeholders control the actual life-cycle of services. In particular, in service ecosystems we can observe five interacting stakeholders which we define as follows:

- Service developer. Service developers creates the service by implementing the service. Thus, service developers control the source code of services and manage the development of services.
- Service user. Service user define requirements that services must fulfill and use them. Service users, service usage respectively, can be regarded as major indicator for the fitness of a service.
- Service integrator. Service integrators integrate (external) services into service based applications that are used by the end user.
- Service provider. Service providers are responsible for the actual offering of the service.
- Service broker. Service broker manage information that is available for services and support the service discovery process. They provide access to repositories that store service related information, like ontological description of services, etc.

The interaction process between the stakeholders is highly dynamic and may result in changes of available resources. Note that we do not consider the different stakeholders as mutual exclusive, for instance, a single organization can easily be service provider, service integrator and service user at the same time. As briefly mentioned above, new requirements for a service user can result in service interface changes which may have impact on other services that use the service. These services may not be able to use the service, due to a new interface. Another example is the addition of competing services to the ecosystem which also may have consequences for the service usage.

To conclude, services change during their life cycle, because of changes in of service KPIs, requirements, implementation, etc. These changes originate from various sources, like service user, service integrator, etc. In order to benefit from a biological service model, we envision the use of various (historical) service related information as input to such a model. We intend to use different (QoS) monitoring tools and put the data into the evolutionary service model. Based on observations of the past, our ultimate goal is to learn from the past and make predictions of the future service behavior.

The achieve this goal, we need expressive languages that are able to capture the dynamics of services in service ecosystems, which we are going to study in the future. Such languages must provide the means to define inter-dependencies between services and their required resources. Especially run time changes of service quality characteristics and potential effects of changes need to be described in such languages and integrated into a service model.

3.1.2 Semantic Service Models

Semantically rich service models attempt to address the need for automated and dynamic service discovery and composition by providing an elaborate description of the service behavior which is machine-interpretable and machine-processable. Previous service models such as WSDL [9] have been based on syntactic descriptions, essentially offering an invocation interface while any other supplementary information is only human-understandable. Semantic Web technologies provide us with languages and tools to replace these purely syntactic descriptions with complete representations of what a service is supposed to do under any circumstances.

Semantic service models utilize ontologies and rule languages to express assertions for the behavior of services, in addition to their inputs and outputs. For instance, OWL-S proposes the addition of assertions, called preconditions, that must be satisfied prior to the service execution in order to ensure success, as well as assertions that describe the state of the world after a successful execution, called effects. Effects are closely related to outputs in the OWL-S service model, by using the term result to refer to a coupled effect and output. Assertions can be also expressed on the conditions under which a result occurs. The semantic service models proposed by other Semantic Web services efforts such as WSMO [10] and SWSO [11] are similar. In WSMO, assertions that must be satisfied prior to the service execution are either called preconditions or assumptions, depending on whether they deal with the information space or the state of the world, respectively. Also, conditions on results are called postconditions.

Semantic service models such as the ones briefly outlined here succeed in capturing constraints that deal with the state of the world before the execution of the service and the state of the world afterwards. Thus, one can be informed of the conditions that must be met before the execution of a particular service and the conditions that will be true after a valid execution of that service. The resulting descriptions are far more elaborate than syntactic descriptions while at the same time assertions are expressed using Semantic Web rule languages, effectively making them machine-processable. Thus, the goal of automated and dynamic service discovery and composition is much more feasible than before. However, there are some issues that have not yet been addressed by any existing semantic service model. We will address some of these issues in the rest of this subsection.

Augmenting Semantic Service Models with Invariants

In [12], it is argued that current semantic service models are inadequate because they are incapable of describing more complex assertions, such as conditions that must be satisfied both before and after service execution. Existing assertions such as preconditions, postconditions, assumptions and effects refer either to the state before service execution or the state after. For instance, we cannot express that a given predicate must have the same truth value both before the service is executed as well as afterwards. To that end, a new kind of constraints, named invariants, is introduced. Invariants serve a role similar to the integrity constraints in databases, ensuring the consistency in service execution. A correct service execution is achieved when preconditions and invariants are satisfied in a state before the service executes and postconditions and invariants again are satisfied in a state after service execution. If invariants are not satisfied both before and after service execution, then the resulting state is inconsistent and we cannot assert that the service has executed correctly.

Invariants have yet to be integrated in any of the existing semantic service models, although a similar kind of assertions, called execution invariants, that must be satisfied in every state of a Web service execution have been briefly described in a deliverable by the WSMO working group [13]. Due to their use in many application scenarios and their assistance in expressing assertions that are common in Web services, it can be stated without doubt that they should be included in a semantic service model.

A service model that offers the ability to describe assertions that must be satisfied both before and after service execution in addition to the usual preconditions and postconditions can be used to create rich service specifications that can capture the behavior of the service more accurately. However, this does not come without a cost. Adding a new set of conditions makes specifications more complex and more vulnerable to a family of problems that appear in formal specifications using the precondition/postcondition notation. These problems are related to a well-known problem in the field of Artificial Intelligence, called the frame problem.

The Frame Problem

The frame problem may occur in any formal specification, simple or complex, and mainly deals with finding a concise and succinct way of expressing that "nothing else changes" except for what is explicitly declared in the specification. Descriptions lacking frame axioms are considered incomplete and may hinder the ability to formally prove properties of Web services and their behavior. Let's consider the example of a Web service that withdraws an amount of money from a bank account associated with a credit card. We will only deal with a subset of the service specification that deals with the daily withdrawal limit of the account. We need to consider two different cases. If the daily withdrawal limit has been reached, the use of the card should be banned for the day. If the limit has almost been reached, the cardholder should be warned. These postconditions of the service, can be expressed as follows, using first-order predicate logic[1] (we use the variable WL to refer to the withdrawal limit):

$$withdrawalTotal'(day, account) \geq dailyLimit(account) \Rightarrow ban(day, account)$$

$$withdrawalTotal'(day, account) < dailyLimit(account) \wedge$$
$$dailyLimit(account) - withdrawalTotal(day, account) \leq WL \Rightarrow warn(day, account)$$

$$withdrawalTotal'(day, account) < dailyLimit(account) \wedge$$
$$dailyLimit(account) - withdrawalTotal(day, account) > WL \Rightarrow \neg warn(day, account)$$

These postconditions, however, do not consider all possible cases for the included predicates and as a result are incomplete. For instance, we need to ensure that all accounts not associated with the current action remain unchanged. To explicitly state that everything remains unchanged, except when stated otherwise, we need a set of new clauses. These clauses are known in literature as frame axioms. The task of writing these clauses is not a trivial one, mainly due to the inclusion of conditional axioms which leads to many different cases that need to be examined separately in order to include a different set of frame axioms for each one of them. For each of the postconditions state above, we need to add a set of frame axioms shown here:

$$withdrawalTotal'(day, account) \geq dailyLimit(account) \Rightarrow ban(day, account) \wedge$$
$$\forall x, y[x \neq day \vee y \neq account \Rightarrow ban'(x, y) \equiv ban(x, y)]$$
$$\forall x, y[warn'(x, y) \equiv warn(x, y)]$$

$$withdrawalTotal'(day, account) < dailyLimit(account) \wedge$$
$$dailyLimit(account) - withdrawalTotal(day, account) \leq WL \Rightarrow warn(day, account) \wedge$$
$$\forall x, y[x \neq day \vee y \neq account \Rightarrow warn'(x, y) \equiv warn(x, y)]$$
$$\forall x, y[ban'(x, y) \equiv ban(x, y)]$$

$$withdrawalTotal'(day, account) < dailyLimit(account) \wedge$$

[1] For the sake of readability, we assume that all functions used in first-order logic statements in this section are always defined.

$dailyLimit(account) - withdrawalTotal(day, account) > WL \Rightarrow \neg warn(day, account) \wedge$
$\forall x, y[x \neq day \vee y \neq account \Rightarrow warn'(x, y) \equiv warn(x, y)]$
$\forall x, y[ban'(x, y) \equiv ban(x, y)]$

It should be obvious from the example that stating frame axioms concisely and succinctly is a complicated task with many different parameters that need to be taken into consideration and is, in its essence, the frame problem. Moreover, the resulting specification, while being complete, is also rather lengthy and computing formal proofs based on them is a more difficult and error-prone task. If we advance even further, attempting to compose such specifications that include frame axioms, in order to create a composite service specification that is consistent will most certainly be a challenging task, since we may have to combine frame axioms with opposite statements for the same predicates. Thus, it is apparent that a major step in our attempt to create semantically rich service specifications should deal with addressing the frame problem.

Addressing the Frame Problem

Through the example presented in the previous section, it is apparent that attempting to completely state all frame axioms when devising Web service specifications is problematic, especially when the original specifications contain many different conditions. The frame axioms, as expressed in the example, offer a procedure-oriented perspective to the frame problem, explicitly asserting what predicates each procedure does not change in addition to those it changes. In [14], the authors identified this fact as the source of the frame problem and aimed to replace the procedure-oriented with a state-oriented one, which we will explore in this section.

Instead of declaring what predicates don't change in each Web service specification, we can reverse our viewpoint and declare, for each element of the service specifications we are creating, which services may result in changing them. Thus, we don't aim to write a set of frame axioms for each individual Web service specification, but we create assertions that explain the circumstances under which each predicate or function might be modified from one state to another. These assertions, called explanation closure axioms or change axioms in [14], provide a state-oriented perspective to specifications.

To be able to express the change axioms, a simple extension to the first-order predicate logic is proposed, that adds a special predicate symbol, named Occur and a special variable symbol named α. The semantics for these two additions are simple. Variable α is used to refer to services taking part in the specification. Occur(α) is a predicate of arity 1 that is true if and only if the service denoted by the variable α has executed successfully. Thus, a further goal in our attempt to create a semantically rich service model is to integrate change axioms in service descriptions.

In conclusion, it should be stated that semantically rich service models allow us to thoroughly know and, in most cases, predict the service behavior under any circumstance. This demands more expressive service specifications than ones limited to interface descriptions and may lead to problems such as the frame problem described here, which have to be addressed before one can harness the complete power that such service models can offer. It should also be noted that the research goals proposed in this subsection deal mainly with the functional properties (and, to some extent, the behavioral properties) that may be included in a semantic service model. However, a complete service model should also include non-functional properties which deal with security, performance and QoS aspects among others.

3.2 Formal Models for QoS-Aware Service Compositions

The goal of QoS-aware service compositions is to take into account QoS attributes of individual services in the composition, and express the aggregate QoS attributes of the whole composition. In this section we present a preliminary identification of some of the the problems around QoS-aware service compositions, and give some comments on the ideas towards possible solutions that are currently work in progress within the S-Cube project and which try to seamlessly combine QoS and semantic concerns. The quality attributes that are addressed by this approach can include the traditionally performance-related ones, such as execution time and availability, or more general quality attributes in the business settings, such as cost. Therefore, formal models for QoS-aware service compositions must take into account both the functional behavior, or semantics, of a service composition (accomplishment of the data processing task it is in charge of), and attainment of the expected quality standards.

The purpose of formal models of QoS-aware service compositions is twofold. They must be sufficiently expressive to describe a wide class of service compositions and QoS attributes, while being sufficiently constrained to ensure that standard reasoning tasks on such models are, at least in common cases, decidable and reasonably efficient. Besides, the formal methods in general have an advantage of having non-ambiguous meaning, as well as having inference procedures about model instances that ensure soundness of the reasoning results.

However, the challenge of formulating a formal model that is well-suited for expressing a wide class of QoS-aware service compositions, as well as for reasoning about them, is not trivial. Although there is a multitude of general computation models based on different computation paradigms, as well as a number of service-specific models based on abstract processes or variants of first-order logic and constraints, none of them encompasses, to the best of our knowledge, all concerns relevant for modeling of QoS-aware compositions, such as representing QoS characteristics themselves, accounting for the dynamic nature of compositions, and ability to handle both quality and semantic (behavioral) aspects.

One direction to devise formal QoS-aware service composition models, with the above stated desirable properties, can be based on Description Logics (DL) enriched with constraints, extending earlier work on propagation and resolution for service discovery [15]. The formalism of DL [16] is well researched and widely used as the basis for many Web-centric reasoning systems, including the Semantic Web [17] and reasoning on Web services [18]. The way in which concepts are structured and described in DL is sufficiently similar to the use of classes, inheritance, attributes and relations encountered in methods of Object-Oriented software design [19], and should therefore be relatively familiar and intuitively intelligible to a wider software engineering community. Yet, along with other advantages, DL does provide a formal and sound basis on which reasoning on services and their QoS-aware compositions can be safely performed.

In such a DL-based model, QoS attributes could be modeled as approximating functions taking into account input data, e.g., as functions on the parameters on input messages. Such data-aware QoS can potentially be applied not only to performance-related QoS dimensions, but also to other quality dimensions, such as Quality of Information (QoI), Quality of Experience (QoE), or in some cases even to Quality of Business (QoBiz), as outlined in JRA-1.3.2.

Constraints can be used to further improve expressiveness of a DL-based model, without compromising its decidability or excessively increasing the complexity of the resolution procedure. For instance, constraints can be used to model numerical QoS requirements, without having to deal with infinite domains directly at the level of DL. Another relevant improvement can be the use of soft constraints for QoS modeling [20, 21]. The key idea here is to allow more flexible constraint satisfaction, where inability to satisfy a constraint does not lead to a failure, but rather incurs a penalty on the solution. Soft constraints can be used for avoiding failures to discover good-enough solutions with minimal penalty when it is not

guaranteed that solutions that satisfy all constraints (i.e. with zero penalty) can be found (e.g., in the case of over-constrained problems).

Development of formal models for QoS-aware service composition has to be accompanied by definition of a set of standard reasoning tasks within that framework, based on a critical assessment of the existing mechanisms and frameworks. These tasks may include unified QoS-aware service selection, matchmaking, and deduction of ad-hoc compositions to serve a particular kind of query. Successful solution to the above outlined challenges would be a key to practical usability of QoS-aware service composition models, and a foundation for further advances towards defining specialized software components (possibly services themselves) in charge of performing automated reasoning tasks for end-users or client services.

3.3 Transactional Web Services

Service oriented architecture (SOA) is a paradigm for organizing and utilizing distributed capabilities that may be under the control of different ownership domains. By making resources in a distributed system available as independent composable services, SOAs reduce complexity and increase flexibility. Composability of services allows organizations to create (new) applications within their enterprise information systems just by aligning existing services. Services in an SOA tend to have a "coarse grained" nature. That is, a service often encapsulates a set of related business functions and consumes considerable computing resources.

Due to the inherent autonomy and heterogeneity of Web services, ensuring composite services reliability remains a challenging problem. Therefore, using a service assembly requires major efforts in order to deliver a coordinated collective result. Such coordination efforts may be addressed solely in the process logic that assembles the services. However, according to [22], transaction processing concepts are a superior option. By managing a group of services, transaction processing concepts guarantee that the group of services achieves a coordinated common, consistent, and mutually agreed outcome. For tightly-coupled systems, such an approach for tackling coordination is common and ubiquitous. Extending the classical control flow with a transactional flow (encapsulating a set of recovery mechanisms) is widely accepted for ensuring composite services reliability. However, current approaches define recovery mechanisms in and ad-hoc way while they have to respect consistency rules regarding the control flow.

3.3.1 Transactional behavior in composite Web services

In the loosely coupled environment represented by Web services, long running applications will require support for recovery and compensation, because machines may fail, processes may be canceled, or services may be moved or withdrawn. Web services transactions also must span multiple transaction models and protocols native to the underlying technologies onto which the Web services are mapped. However, handling failures using the traditional transactional model for long running, asynchronous, and decentralized activities has been proven to be unsuitable. Advanced Transaction Models (ATMs) [23] have been proposed to manage failures, but, although powerful and providing a nice theoretical framework, ATMs are too database-centric, limiting their possibilities and scope [24] in this context (e.g. their inflexibility to incorporate different transactional semantics as well as different behavioral patterns into the same structured transaction). In the same time, workflow has became gradually a key technology for business process automation [24], providing a great support for organizational aspects, user interface, monitoring, accounting, simulation, distribution, and heterogeneity. In our transactional CS model, we propose to combine workflow flexibility and transactional reliability to specify and orchestrate reliable Web services compositions.

Transactional Web service model

A transactional service, t_s, is a triplet $t_s = (ID \in \mathcal{O}bject,\ E \subset \mathcal{S}tates,\ T \subset \mathcal{T}ransition)$ where ID is an object designating service ID, E is the set of its states and T is the set of transitions performing the changes between states.

Each service can be associated to a life cycle statechart. A set of of states (*initial, active, canceled, failed, compensated, completed*) and a set of transitions (*activate(), cancel(), fail(), compensate(), complete()*) are used to describe the service status and the service behavior. We distinguish between internal (intra-service) transitions (*complete(), fail(), and retry()*) and external (inter-services) transitions (*activate(), cancel(), and compensate()*). External transitions are fired by external entities (other services, human actor,etc.). Typically they allow a service to interact with the outside and to specify composite services orchestration. The internal transitions are fired by the service itself (the service agent) and are invariably defined. The internal service behavior is refined to express the service transactional properties.

The main transactional properties [25] of a Web service we are considering are *retriable, compensatable and pivot*. A service *ts* is said to be ***retriable*** if it is sure to complete after finite activations. *ts* is said to be ***compensatable*** if it offers compensation policies to semantically undo its effects. *ts* is said to be ***pivot*** if once it successfully completes, its effects remain and cannot be semantically undone. Naturally, a service can combine properties, and the set of all possible combinations is $\{r; cp; p; (r, cp); (r, p)\}$.

The requested transactional properties can be expressed by extending the service states and transitions. For instance, for a compensatable service, a new state *compensated* and a new transition *compensate*() are introduced (e.g., service in figure 3.1.b). Figure 3.1 illustrates the states/transitions diagram of a retriable service (3.1.c) and states/transitions diagrams of services combining different transactional properties (figures 3.1.d and 3.1.e).

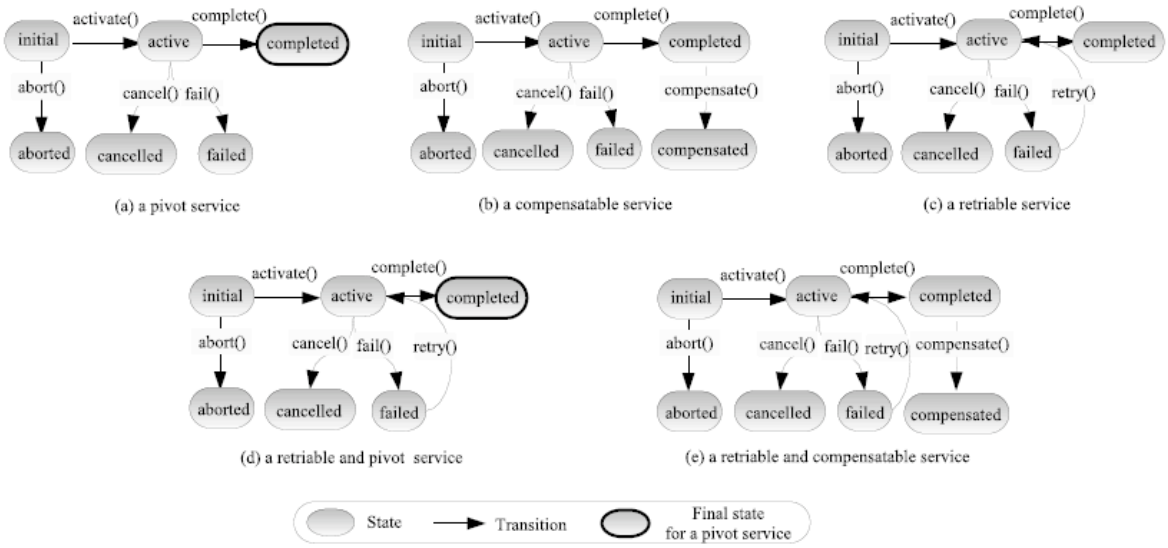


Figure 3.1: Services states/transitions diagrams according to different transactional properties [26]

Transactional composite service model

A composite service is a conglomeration of existing Web services working in tandem to offer a new value-added service [27]. It orchestrates a set of services, as a composite service to achieve a common goal. A transactional composite (Web) service (TCS) is a composite service composed of transactional services. Such a service takes advantage of the transactional properties of

component services to specify failure handling and recovery mechanisms. Concretely, a TCS implies several transactional services and describes the order of their invocation, and the conditions under which these services are invoked.

Dependencies between services

A TCS defines services orchestration by specifying dependencies between services. They specify how services are coupled and how the behavior of certain service(s) influence the behavior of other service(s). A dependency from service $s1$ to service $s2$ exists if a transition of $s1$ can fire an external transition of $s2$. A dependency defines for each external transition of a service a precondition to be enforced before this transition can be fired. We distinguish between "normal" execution dependencies and "exceptional" or "transactional" execution dependencies which express the control flow and the transactional flow respectively.

The control flow defines a partial services activations order within a composite service instance where all services are executed without failing, canceled or suspended.

The transactional flow describes the transactional dependencies which specify the recovery mechanisms applied following services failures (*i.e.* after *fail()* transition). We distinguish between different transactional dependencies types (compensation, cancellation and alternative dependencies). Alternative dependencies allow to define a forward recovery mechanisms. A compensation dependency allows to define a backward recovery mechanism by compensation. A cancellation dependency allows to signal a service execution failure to other service(s) being executed in parallel by canceling their execution.

Transactional Recovery Rules

We argue that the recovery mechanisms defined by the Transactional Models (TM) result from more general recovery rules. The recovery mechanisms are just an interpretation of these rules according to the structure and transactional semantics considered by the model.

We detect these rules by analyzing the recovery mechanisms defined by the Flexible Transactions Model (FTM) [28]. The rational for choosing FTM is its relative complex structure, its set of recovery mechanisms (TF considers both backward and forward recovery through compensation and alternative respectively) and its use of transaction typing (retriable, compensatable and pivot as in our transactional model).

A Flexible Transaction (FT) has to respect some structural rules ensuring its reliability:

- after a pivot sub transaction (that corresponds to a component service in our model), a set of ordered alternative paths are possible such that the last one must be sure to complete (that means all its sub transactions must be retriable).
- all sub transactions between two pivot sub transactions (or before the first pivot) must be compensatable.

Respecting this structural rules ensures that a FT is recoverable through compensation if it does not reach its first pivot, otherwise it will always terminate successfully. Indeed, in the case of the failure of a sub transaction (which is compensatable), it is possible to recover through compensation till the previous pivot and try another alternative. By making abstraction over the structure constraints, we notice that these recovery mechanisms result from the following more general recovery rules:

R1: when a sub transaction or a path fails, always try another alternative if possible,

R2: when one or a set of sub transaction fails causing the abortion of the whole transaction or path, compensate the partial work done so far,

However, the FT model does not consider parallel executions. In order to take into account this parallelism, we add the following rule in the vein of R2.

R3: when one or a set of sub transaction fails causing the abortion of the whole transaction or path, cancel the running executions.

3.3.2 Transactional WS Patterns

The use of workflow patterns [29] appears to be an interesting idea to compose Web services. However, current workflow patterns do not take into account the transactional properties (except the very simple cancellation patterns category [30]). It is now well established that the transactional management is needed for both composition and coordination of Web services. That is the reason why the original workflow patterns were augmented with transactional dependencies, in order to provide a reliable composition [31].

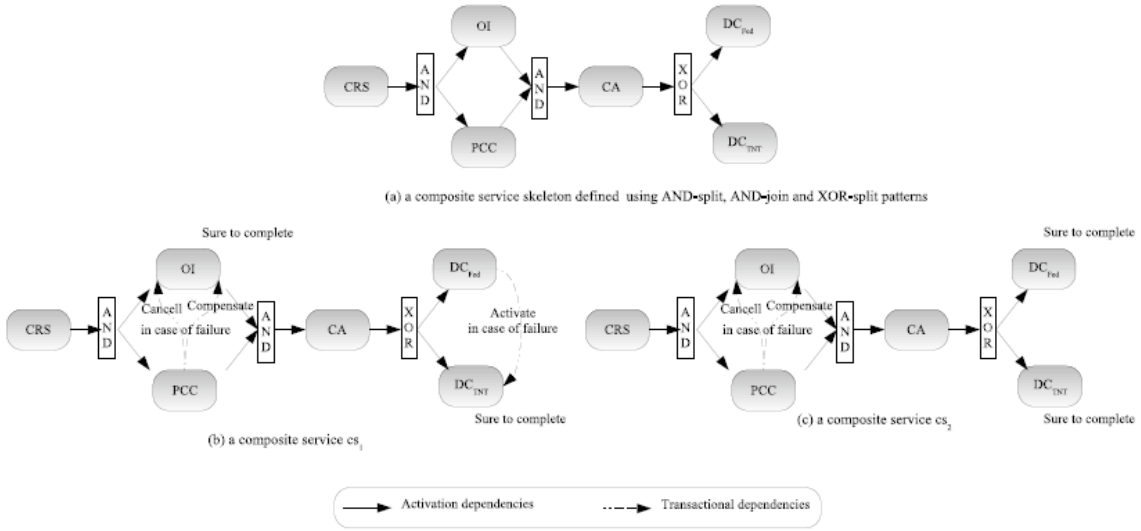


Figure 3.2: Two composite services defined according to the same skeleton

To fulfill the above objective, we use workflow patterns to describe TCS's control flow model as a pattern composition. Afterwards, we extend them in order to specify TCS's transactional flow, in addition to the control flow they are considering by default. Indeed, the transactional flow is tightly related to the control flow. The recovery mechanisms (defined by the transactional flow) depends on the execution process logic (defined by the control flow). Figure 3.2 illustrates this by considering an application dedicated to the online purchase of personal computer. This application is carried out by a composite service. Services involved in this application are: the Customer Requirements Specification (CRS) service used to receive the customer order and to review the customer requirements, the Order Items (OI) service used to order the computer components if the online store does not have all of it, the Payment by Credit Card (PCC) service used to guarantee the payment by credit card, the Computer Assembly (CA) service used to ensure the computer assembly once the payment is done and the required components are available, and the Deliver Computer (DC) service used to deliver the computer to the customer (provided either by Fedex (DC_{Fed}) or TNT(DC_{TNT})).

3.3.3 Validation

Several executions can be instantiated according to the same TCS. The state of an instance of a TCS composed of n services is the tuple $(x1, x2, ..., xn)$, where xi is the state of service si

at a given time. The set of termination states of a TCS cs, $STS(cs)$, is the set of all possible terminationstates of its instances.

In order to express the designer's requirements for failure atomicity, we use the notion of Accepted Termination States ([32]). In other word, the concept of ATS represents our notion of correction. An accepted termination state, ats, of a composite service cs is a state for which designers accept the termination of cs. We define ATS the set of all Accepted Termination States required by designers.

An execution is correct iff it leads the CS into an accepted termination state. A CS reaches an ats if (i) it completes successfully or (ii) it fails and undoes all undesirable effects of partial execution in accordance with designer failure-atomicity requirements [32]. The execution of a composite service can generate various termination states. A composite service is not valid if it exists some termination states that do not belong to the ATS specified by the designers.

3.4 Service Coordination Models

As the adoption of SOA grows in enterprises and businesses, more and more complex information systems are reworked or designed anew on the basis of the emerging service-oriented paradigm. As a consequence, the complexity of the conversations that take place among services provided by the different players increases accordingly, and connecting complex services by "pairing" them on a one-to-one basis does not suffice any longer. Instead, supporting complex, possibly long-lasting multi-party conversations involving a diversity of services becomes critical.

Additionally, and unlike in the case of short-lived partnerships, adequately managing the evolution of service-enabled systems that make it possible continuous multi-party conversations becomes paramount. This includes dynamically replacing partners which take up roles in a business protocol[2] and assessing how this impacts the overall conversation. Operations exposed by services can not any longer be restricted to a single service provider and service requester, but have to scope up to multi-party environments, where each service describes how it can consume and produce messages in conversations involving an arbitrary number of participants.

At the communication level, business protocols can be very flexible, ranging from conventional inter-organizational point-to-point service interactions to fully blown dynamic multi-party interactions of global reach within which each participant contributes its activities and services.

This section will be devoted to giving a global overview of some joint work performed in collaboration with WP-JRA-2.1. As not all parts of this piece of work have already been accepted as formal publications, we think that restricting ourselves to attaching just these accepted papers would give a partial view of this work which would fail to convey the ideas behind it. Therefore, we have decided not to physically attach papers on this issue to the present deliverable, and instead include a more complete set of papers in a later deliverable, probably within WP-JRA-2.1.

3.4.1 Representation Formalisms

The research community is investigating the formalisms best suited to represent and reason with multi-party environments.These are collectively called *business protocol languages*. Proposals for business protocol languages range from adapted business process languages like BPEL-light [33, 34], more formal approaches based on different calculus (e.g., π-calculus [35, 36]), Deterministic Finite Automata (DFA, especially timed automata [37, 38]) and Petri-Nets [39]. Business protocol languages represent a bridge between business protocols seen from their

[2]Bearing in mind that the relation partner-to-role need not be a one-to-one mapping.

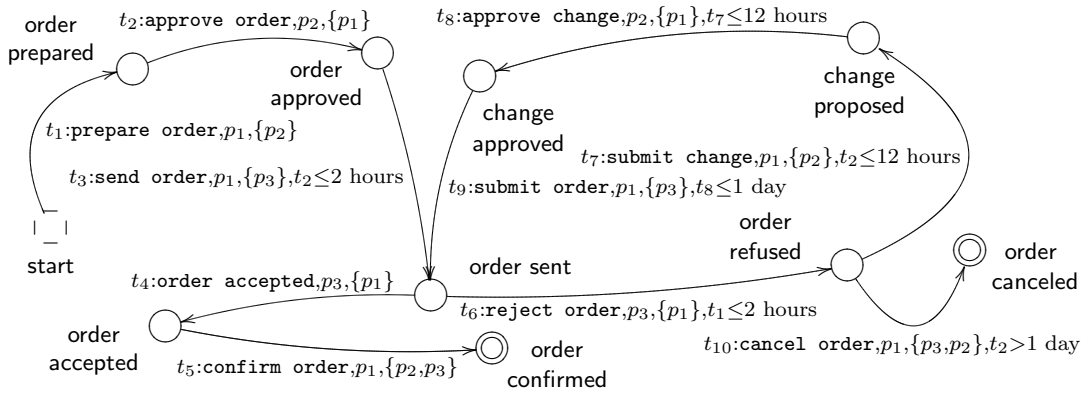


Figure 3.3: *Purchase Order*: an example of a business protocol [40].

structural point of view, i.e. how they should interoperate, and their functional aspect on the other side, i.e. which tools / mechanisms can enable their interoperation. In other words, they provide a hinge to link abstract views of business protocols and the languages and frameworks used to implement them, by stating the needs of the format in a formalism closer to the latter. Within the realm of the S-Cube project, this opens the possibility of acting as one of the bonds between workpackages WP-JRA-2.1 and WP-JRA-2.2. These WPs will gear more towards providing an extended definition of coordination as a concept and coordination protocols as concrete instantiations, and classifying them accordingly. The rationale behind this decision is that not only business protocols, but also choreographies of service compositions, which are considered at the service composition level, can be viewed as coordination protocols. We consider this to be part of the future research work.

Service invocations are often represented as point-to-point uni-cast/multi-cast message exchanges, where messages are basic units of communication among services. One possible formalism, which we have used in our present work in order to describe multi-party business protocols for service networks, is a graph-based representation enriched with temporal constraints (see Figure 3.3), which permits us to give an intuitive and simple semantics to the execution of runs (i.e., sequences of time-annotated message exchanges among the different participants in a protocol). *Accepted* runs are those which are in an accepting state.

From this point of view, a business protocol can be seen as a graphical or formal vehicle for representing *conversations*, where the conversation supported by a business protocol is the set of all *runs* that are *accepted* by the business protocol. The temporal constraints in the graph determine when the transition it is associated which can be traversed. Note that even if the number of message exchanges in a run is customarily finite, the number of different runs that can take place over the same business protocol may be infinite, since, for instance, a particular message can be sent at infinitely many different times.

Such a graph-based representation makes it also easy to map business protocols to timed automata [37], therefore making it possible to apply verification procedures developed for this formalism on our framework.

3.4.2 Towards Truly Distributed Service Networks

One relevant question is up to which point a given (multi-party) business protocol can be executed in a completely distributed fashion using only the means (i.e., partner interactions) expressed in the protocol itself. In investigating this [40], we have introduced the requirement that senders must generate their messages within the correct time windows and that all

participants have to acknowledge the termination of a protocol execution they are part of. There exist another property - *awareness* that is additionally inferred and it appears sufficient for explaining soundness. *Transition awareness* is related to the ability of participants to know when some transition can be traversed; this needs additional knowledge on the associated conditions. Similarly, a participant is said to be *state-aware* of a state if it is aware of every time that the business protocol enters or leaves that state. Transition- and state-awareness, which are mutually dependent, are the keys to *participant soundness*: if participants are aware of the message-based transitions in which they are involved, they have enough understanding of the protocol not to break it without the need of additional synchronization means. Therefore, all participants can acknowledge when the protocol has terminated.

An additional relevant property of choreography business protocols, called *time-soundness*, is related to the ability of protocols to avoid stalling caused by participants not generating messages when they ought to. Service network protocols that are both time and participant sound are called *fully sound*. Fully sound multi-party choreography business protocols rely solely on message exchanges as the only means of communication and can be executed consistently in a completely distributed manner, while guaranteeing termination. Our framework allows the description of business protocols and verification of their temporal properties using model checking of timed automata. Fully sound multi-party choreography business protocols contribute towards a comprehensive theory of management of business protocols for service networks.

3.4.3 Initial Steps Towards Multi-Party BP Evolution: Replaceability and Compatibility

The communication among participants involved in a business protocol can be structured either as *orchestrations*, which describe the local point of view of one of the participants (the *subject*), or as *choreographies*, which provide a global view. Relating orchestrations and choreographies is instrumental to frame the "big picture" of business protocol evolution: how business protocol models can evolve in order to address new/updated requirements such as changes to the behavior of partners, KPIs and QoS parameters to be met, and so on.

Most of our work on replaceability is driven, on one hand, by the need for every participant (which has a role which can be exemplified by an orchestration) to send / accept the messages needed by the business protocol (this is somehow related to the *behavior* of the participant) and, on the other hand, to abide by the time constraints put forward by the choreography and represented as constraints associated by transitions of the automaton modeling the business protocol (see, again, Figure 3.3). This is a concern shared with the workpackage JRA-1.3, where timing is one of the QoS attributes. Respecting these time constraints so that e.g. partner replacements are ensured to be safe is a paramount issue in our work on replaceability, and will be a crucial point when defining the protocol projection operators.

The Interoperability Problem

We are using as basis of our study of evolution of services in long-lasting multi-party conversations the assumption of the existence of a series of primitives that cater for the evolution of the associated business protocols. Two essential operations are the analysis of *replaceability* and *compatibility* analyses. The former evaluates to what extent substituting a participant by another in a business protocol impacts the interoperability with the set of all participants. The latter assesses whether two or more business protocols can successfully interoperate with each other. These analyses help to ascertain which changes can be applied to business protocols while preserving backward compatibility. These operations have been already studied in the literature [38] for two-party business protocols encoded as orchestrations. However, existing analysis methods do not apply to multi-party business protocols, nor allow comparison between

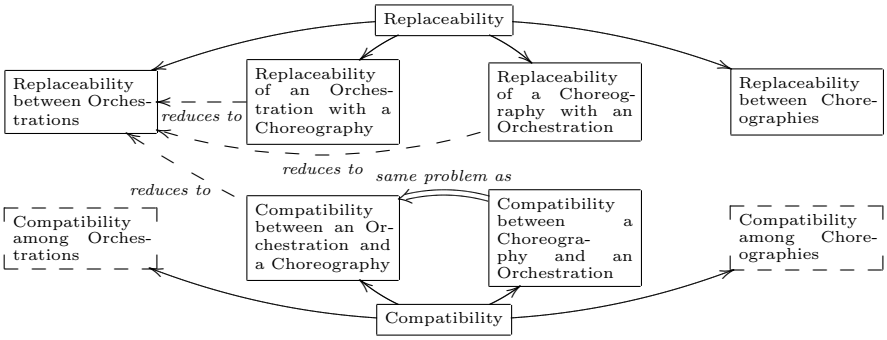


Figure 3.4: Replaceability and compatibility problems in the multi-party scenario.

choreographies and orchestrations. A classification of replaceability and compatibility scenarios for multi-party protocols shows that they are more complex than conventional two-party ones, giving rise to an extension of existing notions of replaceability and compatibility. Figure 3.4 shows a glimpse of the different possibilities.

We have introduced in our work so far, a theory and decision model to perform compatibility and replaceability analyses of multi-party business protocols. The new proposed primitives not only cater for uniform replaceability between orchestrations and between choreographies, but also for mixed flavors of replaceability and compatibility between orchestrations and choreographies, and, vice-versa. This is considered to be a novel research result in the scope of S-Cube.

In the current work we propose an approach based on graph-rewriting rules to systematically extract the orchestration representing the point of view of one of the participants of a choreography on the overall conversation, and base the replaceability and compatibility decisions on that "artificial" orchestration by means of language inclusion in timed automata [37]. Our extraction process aims at optimizing the *similarity* between the initial choreography and the resulting orchestration. That is, it tries to generate orchestrations that share as much of the structure of the initial choreography as possible (by e.g. maintaining the same states wherever possible). By doing this, we aim at decreasing (or removing) problems such as incompatibilities or inconsistencies when running the services.

We have studied compatibility and replaceability between several combinations of orchestrations and choreographies: replaceability between two orchestrations, compatibility between an orchestration and a choreography (and vice-versa), replaceability of an orchestration with a choreography and replaceability of a choreography with an orchestration. All these can be reduced to the first case - replaceability between two orchestrations (see Figure 3.4). The rest of the combinations (compatibility among orchestrations and compatibility among a number of choreographies) are left to further research, since they are actually related to problems of composition of business protocols.

Chapter 4

Service Composition Approaches and Languages

This section focuses on service composition approaches. In the work in this WP we intend to deal with models for services and service compositions that include QoS and behavioural information, corresponding languages for describing these models and possibly formal representations. These will be used to develop novel approaches for service composition that take into account (i) the interdependencies between the SCC layer and the BPM and Service Infrastructure layer and (ii) the QoS-characteristics relevant to SBAs and help enable adaptation of SBAs. Additionally, we intend to map some of the approaches to a concrete service composition technology and improve this technology if necessary to be able to demonstrate the feasibility of the chosen approaches. Our intention is also to investigate approaches for coordination of service compositions, in particular for the out-sourcing and in-sourcing scenarios. We shall concentrate on defining coordination models needed in service compositions and their classification.

In this document we cover only some preliminary research results and approaches. As the project work proceeds we shall improve these approaches and create additional once that cover the QoS and Adaptation aspects of SBAs from the point of view of the Service Composition layer and in a coordinated effort with the other WPs in S-Cube.

4.1 Business Process Development using WS-CDL and WS-BPEL

The approach presented in this section can be considered a top-down approach since it starts with gathering requirements for the top-level composite service and then refines it to an executable service composition.

Motivation

In this section we briefly summarize a top-down modeling approach, published in [41], that uses WS-CDL (Web Service Choreography Description Language) as the choreography description language and WS-BPEL as the orchestration language. The WS-CDL description captures the global model of all the participants and their service interactions. Using this global WS-CDL model as input, the orchestrations in WS-BPEL for each partner are generated. The motivation for using WS-CDL is based on the fact that it has been one of the first pure publicly available choreography languages and BPEL did not provide support at the time of implementing the approach. The novelty of this approach is the consideration of QoS from initial choreography modeling in form of Service Level Agreements that will be mapped to enforceable QoS policies in the orchestration layer.

Mapping Approach

As shown in Figure 4.1, the language constructs of WS-CDL can be mapped to BPEL allowing a choreography description to be transformed into separate BPEL processes, one for each partner in the choreography, including corresponding WSDL descriptions. We do not present each transformation step in detail, therefore the interested reader is referred to [42, 41], however, we give a brief overview of the required steps.

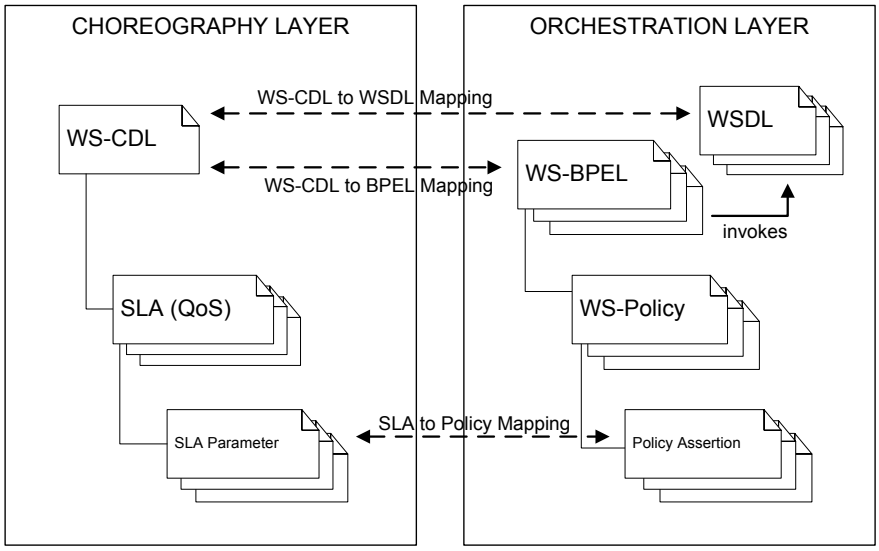


Figure 4.1: Modeling Approach

On the highest level of abstraction (the choreography layer), a number of models have to be specified which can then be used to generate specific parts for each participant in the business process on the orchestration layer.

This is achieved by transforming the models from the choreography layer to executable code in the orchestration layer as depicted in Figure 4.1. The models of the choreography layer include a choreography description in WS-CDL and one or more SLAs. The choreography is used to describe the partners in the process and the message exchanges. The SLAs define obligations and guarantees among the participants. They bridge the gap between the choreography description and the SLAs. We have annotated the choreography with the SLA references to allow a pairwise agreement on a specific SLA.

During the transformation, we map the WS-CDL choreography to a number of BPEL processes (the amount depends on the number of participants) and we generate the WSDL descriptions of the Web services each partner has to implement and provide to its business partners. The importance of QoS in cross-organizational business processes makes it necessary to consider these aspects from the beginning of the development process. Similarly, the SLAs are transformed to WS-QoSPolicy statements (our extension to WS-Policy – very briefly described below) that are directly attached to the corresponding partner links in BPEL to allow an enforcement by a BPEL engine.

Mapping WS-CDL to WS-BPEL. The main goal of transforming WS-CDL to BPEL is to allow the participants a rapid modeling and development process and generate relevant BPEL and WSDL documents which can then be used as a basis to implement the private (non-visible) business logic. The projection of such a global description to endpoint processes whose interactions precisely realize the global description is called *endpoint projection* [43].

Generating WSDL Descriptions. The WSDL descriptions define a static structure which can be extracted from the choreography without analyzing the choreography flow in detail. The necessary element mapping from WS-CDL to WSDL can be seen in [42] (Table 2). The knowledge from this mapping is then used to implement a WSDL generation, which basically works as follows: We generate a new WSDL document for each `roleType` of the choreography if the service interface is invoked somewhere in the choreography flow. The main idea is to check if the `roleType` is referenced within a `channelType` and a `variable` for this `channelType` exists that is used in an `interaction` with another partner. If this is the case, the `roleType` is in use and a WSDL needs to be generated. For details on the algorithm consult [42].

QoS/SLA Integration. The integration of QoS parameters in Web service based business process development raises the need for appropriate techniques to consider QoS at the choreography and orchestration layer. Considering QoS at the choreography layer can be achieved by using SLAs which focus (among others) on performance and dependability aspects of the underlying QoS model. In contrast, the integration of QoS at the orchestration layer can be attained by the use of Web service policies. This section describes how WS-CDL and BPEL can be extended to support QoS attributes.

As mentioned above, we use SLAs to integrate QoS at the choreography layer. For the definition of the SLAs we decided to use WSLA as it seems to be more suitable than WS-Agreement. For the actual integration, we extended WS-CDL with a construct which holds SLA references.

In order to bring QoS aspects from the choreography to the orchestration layer, SLAs have to be mapped to the corresponding Web service policies. However, the current WS-Policy specification focuses on security (WS-SecurityPolicy) and reliable messaging (WS-RMPolicy), whereas performance and dependability are not addressed. Hence, we had to extend the WS-Policy framework by defining a WS-QoSPolicy. The WS-Policy Framework therefore provides a grammar for the definition of domain-specific policies. The WS-QoSPolicy defines assertions for all QoS attributes. The normative outline of the assertions is shown in Listing 4.1. It defines `type`, `unit`, `predicate`, and `value` of the assertion. A concrete example for two such policy assertions is illustrated in Listing 4.2.

```
<qosp:[QoS]Assertion
          unit="xs:string"
          predicate="tns:PredicateType"
          value="xs:integer | xs:flow"/>
```

Listing 4.1: WS-QoSPolicy Assertions

```
<wsp:Policy>
  <wsp:All>
    <qosp:ExecutionTimeAssertion unit="seconds"
          predicate="Less" value="5"/>
    <qosp:ThroughputAssertion unit="requests"
          predicate="GreaterEqual" value="1"/>
  </wsp:All>
</wsp:Policy>
```

Listing 4.2: Assertion Example

Our extension of the WSLA schema restricts the SLA parameters to the pre-defined QoS

attributes introduced in the previous section. Therefore, the SLA can be directly mapped to the WS-QoSPolicy which consists of the following two steps: Firstly, each SLA is mapped to a policy and secondly, each SLA parameter is mapped to a policy assertion.

As each SLA may consist of one or more SLOs, we identified three different patterns:

1. One SLO is defined for each SLA parameter.
2. One SLO consists of multiple SLA parameters.
3. SLA parameters are defined in multiple SLOs.

Each of these patterns can be successfully mapped to an equivalent policy. In the first case, one `All` operator is used to contain all policy assertions. For each SLO, exactly one policy assertion will be generated. For example, an SLO `SLOServiceExecutionTime` defines an SLA parameter which corresponds to the type `ExecutionTime`. This parameter will be mapped to the corresponding policy assertion according to the WS-QoSPolicy.

The definition of a QoS policy and QoS/SLA mapping rules are the fundamental concepts for considering QoS in Web service based business process development. Yet, the question remains how to integrate the generated QoS policies in the orchestration layer. Regarding the top-down modeling approach of Web services, two integration approaches can be differentiated: Policies can either be attached to service descriptions (WSDL) or be integrated in BPEL processes.

Attaching policies to WSDL descriptions following the WS-PolicyAttachment [44] specification has two main drawbacks. Firstly, service invocations are always subject to a policy, even if the service consumer has no corresponding SLA. Secondly, the service provider cannot differ between multiple policies for the same service since policies do not contain information about the participating parties. Therefore, following the second approach the policies should be integrated in BPEL processes.

Extensibility in BPEL is achieved by allowing elements from other namespaces. The BPEL `partnerLink` element is the place to integrate the policy. For this integration, both synchronous (request-reply) and asynchronous (callback) message exchange patterns have to be considered. In contrast to the asynchronous case, the service provider has no additional information about the service consumer in the synchronous case, because the `partnerLink` has no service consumer specific details. Therefore, the policy has to be integrated at the service consumer side as illustrated in Listing 4.3. Alternatively, one could leverage WS-PolicyAttachment to do the integration at the corresponding `partnerLink`.

```
<process>
  <partnerLinks>
    <partnerLink name="POService"
                 partnerLinkType="ns1:POServiceLT"
                 partnerRole="POServiceRole">
          <wsp:Policy xmlns:qosp="..." xmlns:wsu="..."
                      wsu:Id="xs:QName"
                      qosp:operation="...">
             ...
          </wsp:Policy>
        </partnerLink>
    ...
  <partnerLinks>
   ...
</process>
```

Listing 4.3: Policy Integration in BPEL

Chapter 5

Monitoring, Analysis, and Adaptation of Service Compositions

In this Section we deal with the last three phases of the service composition lifecycle: monitoring, analysis, and adaptation. In the context of adaptable QoS-aware service compositions, our focus lies on monitoring, analysis, and adaptation based on QoS-related aspects. *Monitoring* evaluates QoS properties of service compositions whereby both business-level metrics and technical QoS metrics are considered. In the *analysis* phase explanations and predictions of monitored QoS values are provided. Finally, based on analysis results, service compositions are *adapted* and optimized.

5.1 Monitoring and Analysis

Monitoring is the process of collecting relevant information from the execution data of service compositions and involved services in order to evaluate properties of interest and report results of that evaluation. Monitored properties can be based on functional aspects (e.g., correctness properties) or non-functional aspects (e.g., QoS properties). While *monitoring* focuses on reporting of values of monitored properties (what?) in a timely fashion, *analysis* is based on monitoring results and tries to find explanations for monitored values (why?) or predict future values.

In the context of adaptable QoS-aware service compositions, our focus lies on monitoring and analysis of QoS-related aspects. We identify thereby following challenges:

- *Cross-Layer Monitoring:* Current solutions to service composition monitoring mostly focus and are constrained to one layer or very specific aspects, e.g., process metrics as part of business activity monitoring, or QoS metrics as part of SLA monitoring and do not integrate information from all layers and deal with their dependencies. As service compositions implement business processes from the BPM layer, and at the same time are based on technical QoS properties of Web services and IT infrastructure used, monitoring and analysis of service compositions should take into account both business related metrics and technical QoS metrics.

- *Cause Analysis and Prediction:* Besides monitoring of process and QoS metrics, one is also interested in providing explanations and prediction of their values. Both cause analysis and prediction should thereby be integrated into the monitoring framework and provide quick responses in order to enable timely reaction and (pro-active) adaptation of service compositions.

5.1.1 A Framework for Monitoring and Analysis of QoS-aware Service Compositions

Figure 5.1 shows a high-level overview of our framework for monitoring and analysis of service compositions. In our framework we distinguish three different layers. In the *process runtime* layer, a WS-BPEL business process is defined and executed. The process can be executed in a standard WS-BPEL compliant engine, as long as the engine is able to emit the process information necessary for calculating PPMs in form of process events, as expected in Business Activity Monitoring (BAM). In the *monitoring layer*, information about the running business process and the services it interacts with is collected in order to monitor PPMs, and QoS metrics. The user defines a set of interesting PPMs and QoS metrics which are to be monitored and should be available for later cause analysis. Based on these metric definitions, the QoS monitor, the WS-BPEL engine, and all instrumented services, emit needed events into a Complex Event Processing [45] (CEP) event cloud. A monitoring tool extracts and correlates these events and calculates corresponding PPMs and QoS metrics. The evaluated metrics are displayed in the BAM dashboard and are stored in the metrics database for later analysis. In the *process analysis layer*, the collected runtime information is analyzed by the process analyzer component. Outcomes of the analysis are again displayed in the dashboard to the users of the system, which can use this resulting information to optimize the business process.

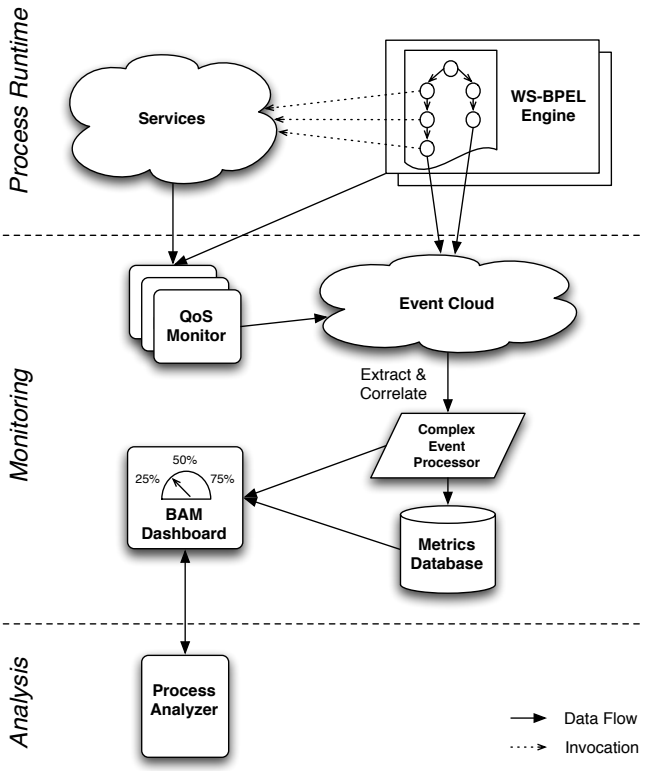


Figure 5.1: Monitoring and Analysis Framework Overview

5.1.2 Monitoring Performance of Service Compositions

Service Compositions implement business processes and at the same time they are based on IT infrastructure and implemented in terms of Web services. This is why we can distinguish between two different types of characteristics when monitoring the performance of service compositions:

- *Process performance characteristics:* As service compositions implement business pro-

cesses defined in the BPM layer, we can evaluate business process performance based on service compositions. Process performance is not only related to time-based characteristics (process duration, deadline adherence), but also to process cost and process quality. Process Performance is assessed by measuring process performance metrics (PPMs).

- *Technical QoS characteristics:* A Service Composition run on IT infrastructure and invokes other Web services for implementing their activities. Thus, its performance depends on technical QoS characteristics such as availability, response time, and throughput. These QoS characteristics have also to be measured, and are in particular important for analysis purposes, as QoS characteristics influence process characteristics and vice versa (e.g., availability of IT infrastructure influences directly process duration, and indirectly also process cost, and process quality).

In the following, we explain how PPMs and QoS metrics are monitored in our framework.

PPM Monitoring

Process Performance Metrics (PPMs) are evaluated in our approach based on process events emitted by a WS-BPEL engine or some other instrumented software system used in the service composition [46]. Most WS-BPEL engines already support an event-publishing mechanism, which in most cases is also to some extent configurable on which events to publish (via event filters). Based on the PPM metrics that have to be monitored, we determine needed events and their content and create event filters which filter those events from instrumented systems. The monitoring tool subscribes to these events and evaluates them at process runtime as they are received. Note that process events have to be correlated based on process instance identifiers. In special cases, technical process instance identifiers which are part of each event emited by a WS-BPEL engine, can be used. In the general case, however, business identifiers are needed (e.g., order ID).

QoS Monitoring

QoS metrics are typically evaluated either by probing (e.g., invoking a service endpoint for checking its availability) or by instrumentation. We focus on monitoring of the QoS characteristics availability, response time, and accuracy (defined as 1 - # failed requests/# total requests). In our approach, we use an external QoS monitor for measuring the availability of the process engine and partner Web services of the WS-BPEL process. The QoS monitor polls the corresponding endpoints and emits events which contain information on their availability at a certain point in time. An approach to measuring response time and accuracy of partner Web services of a process is to instrument the Web service invocation module of the WS-BPEL engine. That means that as part of the execution of a WS-BPEL invoke activity, QoS events are emitted which contain information on the measured QoS data (response time and in case of accuracy whether the request was successful or failed).

Correlation of PPMs and QoS Metrics

If we want to find out the dependencies between PPMs and QoS Metrics (e.g., what was the availability of the process engine while customer order X has been processed?), they have to be correlated. The correlation, however, cannot be done based on process instance identifiers, as QoS metrics are not measured specific to process instances. Assume the PPM order fulfilment cycle time and its evaluation for a specific process instance, i.e., a specific customer order. Assume also that we measure availability of process engine. The QoS monitor collects availability data in a certain time period in which many process instances (customer orders) are processed. However, as a specific order is processed in a certain time frame, we

have to calculate the availability of the process engine in the very same time frame, because outside of this time frame the availability of the engine has no impact on the PPM value of that process instance. This is why we need to correlate these metrics based on time frames (and not process instance identifiers).

5.1.3 Analysis of Factors Influencing KPIs

Besides just monitoring the metrics and providing their values in near real time, we are interested in providing explanations of their values. When KPIs (key metrics in business context with assigned target values) do not meet their target values, business users are interested in finding out the causes and the most influential factors. In our case, we want to be able to derive the most influential factors and dependencies of KPIs on process performance and QoS characteristics.

The analysis approach assumes that a set of metrics (both PPMs and QoS metrics) is monitored. A subset of this potentially big set of metrics is classified as KPIs by defining target values for them. Typically, KPIs are based on high level PPMs (e.g., order processing time), but also QoS metrics can theoretically be used as KPIs. Our goal is to find out which of the QoS and process characteristics represented by the whole set of monitored metrics has the most influence when KPIs violate their target values. The output of dependency analysis is a decision tree that presents the most important factors of influence of process performance. We refer to this tree as *dependency tree*, because it represents the main dependencies of a business process KPI on technical and process metrics, i.e., the metrics which contribute "most often" to the failure or success of a KPI of a process instance.

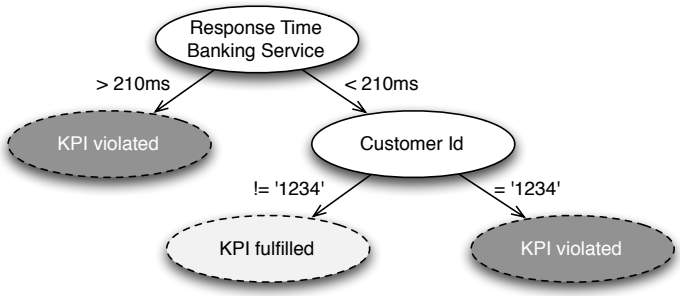


Figure 5.2: Example Dependency Tree

An example dependency tree is shown in Figure 5.2. In this example, the most influential factor is the response time of the banking service, since a delay in this service generally leads to a violated KPI. However, even if the banking service response time is acceptable (below 210 time units in this example), the KPIs are still often violated if the order is placed by the customer with the ID '1234'.

Users can use the dependency tree to learn about possible corrective actions if a process underperforms. For example, considering the example in Figure 5.2, the user can take the corrective action to replace the "Banking Service" against a service with better response time, if such a service is available. Of course, such a replacement could happen in a (semi-)automated way, if the monitoring and analysis framework is integrated with adaptation mechanisms for service compositions.

Our approach to dependency analysis is based on decision tree classification, a well-known technique from the area of machine learning [47], to automatically learn dependency trees from existing process instances. Decision tree classifiers are a standard machine learning technique for supervised learning (i.e., concepts are learned from historical classifications, in our case dependency information is learned from monitoring of previous process instances). Decision trees follow a "divide and conquer" approach to learning concepts – they iteratively

construct a tree of decision nodes, each consisting of a test; leaf nodes typically represent a classification to a category. In our case, only two categories exist (KPI has been violated, or not). The process analyzer is flexible in the concrete algorithm that is to be used to construct the decision tree (e.g., C4.5 [48], or alternating decision tree [49]) and the parameterization (e.g., whether to use pruning [50] or not). However, data preprocessing (i.e., the process of reformatting raw process data in such a way that it can be fed into the machine learning algorithm) is done automatically after selecting a KPI which is to be analyzed.

5.2 Adaptation

The process-based service composition models are described in terms of the following entities: the process logic containing control (activities or steps to be performed, their sequencing and the transition conditions) and data flow; functions that implement the single activities in the control flow (functions dimension). Some activities/steps of the process model interactions with services, which implement the activities, hence the activities are assigned service implementations either statically or declaratively (these may contain descriptions of the functional and non-functional properties the services should have). This section starts the discussion about the adaptation dimensions (?degree of freedom?) of processes (here the term service compositions (SC) is used synonymously) followed by the outline of reasons and motivation for the need of adaptation of the processes, i.e. the adaptation drivers. Finally some adaptation mechanisms for SC are presented. This document will not present a complete list of adaptation mechanisms; here, we identify and discuss three such mechanisms, namely (a) fragmentation of service compositions, (b) service re-binding and (c) biological systems related approaches for SC adaptation. Note: the classifications of adaptation drivers, the process adaptation types and the mechanisms presented here are preliminary and will undergo a refinement process in the later stages of the project, which will be reflected in the subsequent deliverables.

Work done in this deliverable as well as the WP in general overlaps with the work of WP-JRA-1.2. The definition of the overlaps is discussed in detail in the documents produced by IA-3.1 and is out of the scope if this deliverable. The information provided in this document will be refined in the later stages of the projects and will be described in the follow-up deliverable document (CD-JRA-2.2.4).

5.2.1 Adaptation dimensions

We differentiate the following orthogonal classifications of process adaptation/change types (details are presented next):

- Evolutionary changes vs. ad-hoc changes of the process. Evolutionary change/adaptation is done on the process model and thus on all of the process instances of the process model whilst an ad-hoc change is performed on single process instances.
- Adaptation of the process logic dimension. On this SC dimension tasks can be inserted, deleted or substituted. Furthermore, depending on granularity of substitution one may make coarse- or fine-grained changes of the process logic: there are adaptations tackling changes on granularity of tasks vs. process-fragments. The transition conditions guarding the transitions from one activity to the next may also be altered in terms of using another expression or other variables for its calculation. Another element of the process logic dimension is the data flow. Changes in the data flow may include: changing the data flow connectors, if such are available in the SC language or meta-model or at least the flow of data from one activity to the next and the data dependencies among activities; changing the variables used ? either the variables used or their data types; changing the data manipulation specifications.

- Adaptation on the functional dimension. The possible changes of service composition that can be done on this dimension are related to the implementation of the activity in the process model and the information available in the process model. The adaptation of the functional dimension of SC does not alter the process logic structure (control flow), but merely exchanges the information related to the service to be invoked. . Such changes can be related to the binding information (i.e. transport protocol, formats and location of the service to use) provided for the service to be invoked. In a Web Service environment, this is the binding information containing the transport protocol, the formats of the messages exchanged with the Web service and the location of the service. The binding information may also be specified declaratively in terms of non-functional properties and discovered during process execution. It is also possible to change the type of service to be used as an implementation of a process activity, which is another change type on this dimension. Such a change implies discovering the binding information for the newly identified service type.

- Design time vs. runtime adaptation. Regarding the timing of adaptation (timely location in the lifecycle of service composition) it might be performed during the design time (aka modeling time) or runtime (execution time) of the process. Adaptation done during the design (modeling) time allows for any change in the process logic as well as in the functional dimension. Runtime adaptation has to be checked for correctness, since it influences one or more process instances. Runtime changes may be of any of the kinds specified in the adaptation types presented above. Depending on the required adaptation type different mechanisms need to be devised.

- Proactive vs. reactive adaptation: adaptation done in order to avoid failures or adaptation done as reaction to the identified failures.

5.2.2 Classification of Adaptation Drivers

The need and motivation for adaptation arises because of different reasons which we denote as adaptation drivers. The adaptation mechanisms are the solutions needed to react to the need for adaptation expressed by these adaptation triggers. In the following we briefly outline the preliminary identified drivers and discuss some of them in more detail. For now we identify the following critical drivers for SC adaptation: (i) changes in requirements produced by requirements engineers, (ii) results of testing services associated to single process activities, (iii) results of analysis and monitoring and (iv) identified failures of services meant to perform process activities.

The need for adaptation of service compositions may be expressed in terms of recommendations of requirements engineers. Service compositions have to satisfy a set of requirements including business requirements expressed via Key Performance Indicators and metrics stated in Service Level Agreements, as well as technical requirements as Quality of Services. Due to the dynamics of the service landscapes and changes in the business environment new requirements might be added or already existing ones might be modified. These requirements might put new constraints on the functional and non-functional properties of the services used in the compositions. The changes in set of requirements may imply the need for changes in different dimensions the service compositions, which is in the scope of future work in this WP. Requirements engineering changes may also result in adaptation during design time or run time of service compositions, and may need to be enforced on the level of process model or on some instances only.

Analysis on formal models triggers the adaptation during the design time whilst monitoring and results of the analysis data gathered for monitoring may trigger both design time and runtime adaptation.

The testing phase of Service Based Applications in contrast to testing of classical software applications might overlap with the execution phase of SBA. Service compositions (process models and instances) may be adapted in order to avoid the errors identified by the e.g. online testing of individual services. Parallel to the execution of the process instance it?s not yet invoked services might be tested. If the online testing procedure identifies faults an alternative service should be chosen by the middleware, based on the original service discovery and selection requirements

5.2.3 Adaptation Mechanisms

The types of service adaptation to be used with service compositions depend on the flexibility of the chosen modeling style and employed service composition language. In order to accommodate certain types of adaptation the process modeling language should incorporate corresponding mechanisms on the one hand, thus the need to employ additional mechanisms would be discarded. On the other hand the support of adaptation functionality may be enabled by the middleware, like for instance, support for dynamic binding to services. For the cases where neither the service composition language, nor the middleware can enable a particular adaptation type, a new technique must be developed and implemented in terms of an adaptation mechanism.

The focus of this deliverable will be mainly on the initial overview of methods addressing the fragmentation methods as well as mechanisms of re-binding services to process activities. We will relate these mechanisms to the adaptation dimensions presented above and the currently identified adaptation drivers. In the follow-up deliverables these approaches will be further developed and others will be devised.

The adaptation through fragmentation supports the in- and out-sourcing scenarios. A process fragment is identified might be replaced by a single service from some service provides on the one hand. On the other hand one process might be split into several processes because of e.g. new requirements like improved resource utilization, organizational optimization and other requirements [51]. In order to support the process fragmentation additional coordination mechanisms may be needed. Depending on the need for fragmentation and on the mechanism and rules used for the fragmentation, different coordination protocols have to be created in order to maintain the original logic of the non-fragmented process intact.

For that fact, the formal direction we are investigating to carry out the fragmentation process and the coordination protocols can be based on temporal logic, helping to provide reasoning mechanisms upon the coordination protocols. The formalism of temporal logic [52] is well suited and enough expressive and widely used in the distributed real time systems [53, 54].

One mechanism for adaptation on the functional dimension is the dynamic binding to services during the process runtime: services assigned to the tasks may be exchanged in reaction to an adaptation driver. The middleware takes care of discovery of the alternative service satisfying the given requirements. In [55] the focus is on the modification of the assignment of service implementations to process tasks during run-time. The paper presents the mechanism for dynamic binding and substantiates how this mechanism can be used as a reaction to adaptation drivers originating from requirement engineering, online testing and due to identified service faults.

The work presented in [55] integrates knowledge from the areas of requirements engineering, online testing and state-of-the-art adaptation mechanisms for service compositions. The paper shows clearly that the dynamic binding strategy driven by pre-described service requirements is beneficial over the static binding strategy (more on service binding strategies can be found in [56]). The dynamic binding strategy is a mechanism that can be used when executing service compositions to resolves automatically all service faults, e. g. due to unavailable service implementations The mechanism requires leaving the choice of concrete service implementing

an activity open till the execution of the composition (which means that no binding information is available during modelling). During process execution the discovery and selection of the concrete service to implement the activity is delegated to the service middleware, the selection criteria must be provided by the process itself. The mechanism has been implemented in prototypes.

In contrast, the static binding strategy, however, requires a modification of the process model and its redeployment, unless the so-called parameterized processes are used, where the composition language is extended to enable overwriting the static binding during run time [57]. The dynamic binding mechanism can be used to enable ad-hoc/instance-based changes during composition run time and is a modification on the functional level of processes.

In addition, we have identified that requirements engineering and online testing procedures only need to interact with the enterprise service registry and have no other influence on the existing service middleware. While the online testing aims to remove faulty services from the registry, the requirements engineering activity aims to add new and innovative services to it, which lead to a better fulfilment of the SBA?s requirements.

We are also able to identify future research for the integration of online testing and SC modelling as well as the integration of requirements engineering with SC modelling. Online testing techniques require a certain sequence of tests, e. g. the tester needs to know, which service implementation to test first. This sequence depends heavily on the service compositions in which the service implementation is used. In addition, so far we have considered only service testing. However, it is also important to test the whole service composition (integration test). Techniques need to be developed and integrated with adaptation techniques, which will be addressed in the subsequent deliverables in this WP.

For the requirements engineering field, the most predominant problem is the mapping of the requirements to service descriptions, e. g. requirements or goal descriptions using OWL-S, WSMO or others. This mapping ensures that, for example, services identified as superior in the requirements engineering activity, are actually given preference during service discovery carried out by the middleware. In addition, process instances may be tailored to the so-called context-factors, e. g. processes may be adapted to certain users or a certain device. These context-aware SC adaptations require a tight integration of SC and requirements engineering research.

Moreover, by leveraging the emerging wave of innovations in Web 2.0 that promotes a new paradigm in which both providers and end-users (including non-expert users) can easily and freely share information and services over the Web, we will provide techniques and concepts for prediction in order to avoid unacceptable situation while composing services . These, allow to reuse "good fragments" and customize shared information from past experiences instead of developing composition tasks from scratch.

Chapter 6

Conclusions

In this deliverable we have explored several lines of work, in different stages of development, which we foresee can bring about advances in mechanisms for service composition. These do not exclude each other, as they usually tackle different aspects of the overall composition problem (e.g., quality of service, transactionality, semantics) and different approaches (e.g., biologically-inspired, formal models) and concerns common to all these approaches and aspects (e.g., adaptation and monitoring).

Since the workpackage to which this deliverable contributes is placed at the center of the software and services stack, it naturally interacts very heavily with the rest of the layers and workpackages. This is the reason why core concerns (e.g., quality of service) of other workpackages percolate this deliverable and why there are deliverable sections which explicitly address topics which are also central issues for other workpackages (e.a., adaptation and monitoring, in Chapter 5).

Bibliography

[1] W. T. Tsai, "Service-oriented system engineering: A new paradigm," in *Service-Oriented System Engineering, 2005. SOSE 2005. IEEE International Workshop*, Beijing, China, 2005, pp. 3– 6.

[2] M. P. Papazoglou, "Service-oriented computing: Concepts, characteristics and directions," in *Proceedings of the Fourth International Conference on Web Information Systems Engineering (WISE)*. IEEE Computer Society, 2003.

[3] M. Treiber, H.-L. Truong, and S. Dustdar, "Semf - service evolution management framework," *Software Engineering and Advanced Applications, 2008. SEAA '08. 34th Euromicro Conference*, pp. 329–336, Sept. 2008.

[4] M. P. Papazoglou, "The challenges of service evolution," in *CAiSE '08: Proceedings of the 20th international conference on Advanced Information Systems Engineering*. Berlin, Heidelberg: Springer-Verlag, 2008, pp. 1–15.

[5] G. Canfora, "Software evolution in the era of software services," in *IWPSE '04: Proceedings of the Principles of Software Evolution, 7th International Workshop*. Washington, DC, USA: IEEE Computer Society, 2004, pp. 9–18.

[6] Y. Ma and C. Zhang, "Quick convergence of genetic algorithm for qos-driven web service selection," *Comput. Netw.*, vol. 52, no. 5, pp. 1093–1104, 2008.

[7] R. T. Mittermeir, "Software evolution: let's sharpen the terminology before sharpening (out-of-scope) tools," in *IWPSE '01: Proceedings of the 4th International Workshop on Principles of Software Evolution*. New York, NY, USA: ACM, 2001, pp. 114–121.

[8] A. P. Barros and M. Dumas, "The rise of web service ecosystems," *IT Professional*, vol. 8, no. 5, pp. 31–37, 2006.

[9] E. Christensen, F. Curbera, G. Meredith, and S. Weerawarana, "Web Services Description Language (WSDL) 1.1," Mar 2001, http://www.w3.org/TR/2001/NOTE-wsdl-20010315.

[10] H. Lausen, A. Polleres, and D. Roman, "Web Service Modelling Ontology (WSMO)," World Wide Web Consortium, 2005, W3C Member Submission. [Online]. Available: http://www.w3.org/Submission/WSMO/

[11] S. Battle, A. Bernstein, H. Boley, B. Grosof, M. Gruninger, R. Hull, M. Kifer, D. Martin, S. McIlraith, D. McGuinness, J. Su, and S. Tabet, "Semantic Web Services Ontology (SWSO)," World Wide Web Consortium, 2005, W3C Member Submission. [Online]. Available: http://www.w3.org/Submission/SWSF-SWSO/

[12] V. Alevizou and D. Plexousakis, "Enhanced Specifications for Web Service Composition," in *ECOWS '06: Proceedings of the European Conference on Web Services*. Zurich, Switzerland: IEEE Computer Society, 2006, pp. 223–232.

[13] U. Keller and H. Lausen, "Functional Description of Web Services," ESSI WSMO Working Group, 2006, WSML Deliverable D28.1 v0.1. [Online]. Available: http://www.wsmo.org/TR/d28/d28.1/v0.1/d28.1v0.1_20060113.pdf

[14] A. Borgida, J. Mylopoulos, and R. Reiter, "On the Frame Problem in Procedure Specifications," *IEEE Transactions on Software Engineering*, vol. 21, no. 10, pp. 785–798, 1995.

[15] S. Benbernou and M.-S. Hacid, "Resolution and Constraint Propagation for Semantic Web Services Discovery," *Distributed and Parallel Databases*, vol. 18, no. 1, pp. 65–81, July 2005.

[16] F. Baader, C. Lutz, H. Sturm, and F. Wolter, "Basic description logics," in *The Description Logic Handbook: Theory, Implementation, and Applications*. Cambridge University Press, 2003, pp. 43–95.

[17] F. Baader, I. Horrocks, and U. Sattler, "Description logics as ontology languages for the semantic web," in *Festschrift in honor of Jörg Siekmann, Lecture Notes in Artificial Intelligence*. Springer-Verlag, 2003, pp. 228–248.

[18] F. Baader, C. Lutz, M. Miličić, U. Sattler, and F. Wolter, "A description logic based approach to reasoning about web services," in *In Proceedings of the WWW 2005 Workshop on Web Service Semantics (WSS2005*, 2005.

[19] D. Calvanese and M. Lenzerini, "Reasoning on uml class diagrams in description logics," in *In Proc. of IJCAR Workshop on Precise Modelling and Deduction for Object-oriented Software Development (PMD*, 2001.

[20] S. Bistarelli, U. Montanari, F. Rossi, and F. Santini, "Unicast and Multicast Qos Routing with Soft Constraint Logic Programming," *CoRR*, vol. abs/0704.1783, 2007.

[21] D. Hirsch and E. Tuosto, "SHReQ: Coordinating Application Level QoS," in *SEFM*, 2005, pp. 425–434.

[22] P. Hrastnik and W. Winiwarter, "Twso — transactional web service orchestrations," in *NWESP '05: Proceedings of the International Conference on Next Generation Web Services Practices*. Washington, DC, USA: IEEE Computer Society, 2005, p. 45.

[23] A. K. Elmagarmid, Ed., *Database transaction models for advanced applications*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1992.

[24] W. M. P. van der Aalst and K. M. van Hee, *Workflow Management: models, methods and tools*, ser. Cooperative Information Systems, J. W. S. M. Papazoglou and J. Mylopoulos, Eds. MIT Press, 2002.

[25] S. Mehrotra, R. Rastogi, H. F. Korth, and A. Silberschatz, "A transaction model for multidatabase systems." in *ICDCS*, 1992, pp. 56–63.

[26] S. Bhiri, O. Perrin, and C. Godart, "Ensuring required failure atomicity of composite web services," in *WWW*, 2005, pp. 138–147.

[27] B. Medjahed, B. Benatallah, A. Bouguettaya, A. H. H. Ngu, and A. K. Elmagarmid, "Business-to-business interactions: issues and enabling technologies," *The VLDB Journal*, vol. 12, no. 1, pp. 59–85, 2003.

[28] A. Zhang, M. Nodine, B. Bhargava, and O. Bukhres, "Ensuring relaxed atomicity for flexible transactions in multidatabase systems," *SIGMOD Rec.*, vol. 23, no. 2, pp. 67–78, 1994.

[29] W. M. P. van der Aalst, A. P. Barros, A. H. M. ter Hofstede, and B. Kiepuszewski, "Advanced Workflow Patterns," in *5th IFCIS Int. Conf. on Cooperative Information Systems (CoopIS'00)*, ser. LNCS, O. Etzion and P. Scheuermann, Eds., no. 1901. Eilat, Israel: Springer-Verlag, September 6-8, 2000, pp. 18–29.

[30] W. M. P. van der Aalst and A. H. M. ter Hofstede, "Yawl: yet another workflow language." *Inf. Syst.*, vol. 30, no. 4, pp. 245–275, 2005.

[31] S. Bhiri, C. Godart, and O. Perrin, "Transactional patterns for reliable web services compositions." in *ICWE*, 2006, pp. 137–144.

[32] M. Rusinkiewicz, W. Klas, T. Tesch, J. Wasch, and P. Muth, "Towards a cooperative transaction model: The cooperative activity model," in *In Proc. of the 21st Int. Conference on Very Large Databases*, 1995, pp. 194–205.

[33] J. Nitzsche, T. van Lessen, D. Karastoyanova, and F. Leymann, "BPEL$^{\text{light}}$," in *BPM*, ser. LNCS, G. Alonso, P. Dadam, and M. Rosemann, Eds., vol. 4714. Springer, 2007, pp. 214–229.

[34] T. van Lessen, J. Nitzsche, and F. Leymann, "Formalising Message Exchange Patterns using BPEL^{light}," in *IEEE SCC (1)*. IEEE Computer Society, 2008, pp. 353–360.

[35] R. Milner, "Elements of interaction," *Communications of the ACM*, vol. 36, no. 1, pp. 78–89, 1993.

[36] A. Lapadula, R. Pugliese, and F. Tiezzi, "A Calculus for Orchestration of Web Services," in *ESOP*, ser. Lecture Notes in Computer Science, R. D. Nicola, Ed., vol. 4421. Springer, 2007, pp. 33–47.

[37] R. Alur and D. L. Dill, "A theory of timed automata," *Theoretical Computer Science*, vol. 126, pp. 183–235, 1994.

[38] J. Ponge, B. Benatallah, F. Casati, and F. Toumani, "Fine-Grained Compatibility and Replaceability Analysis of Timed Web Service Protocols," in *ER*, ser. LNCS, C. Parent, K.-D. Schewe, V. C. Storey, and B. Thalheim, Eds., vol. 4801. Springer, 2007, pp. 599–614.

[39] H. Foster, S. Uchitel, J. Magee, and J. Kramer, "Compatibility Verification for Web Service Choreography," in *ICWS*. IEEE Computer Society, 2004, pp. 738–741.

[40] M. Mancioppi, M. Carro, W.-J. van den Heuvel, and M. P. Papazoglou, "Sound Multi-party Business Protocols for Service Networks," in *Proceedings of the Sixth International Conference on Service Oriented Computing*, ser. LNCS, vol. 5364. Springer-Verlag, December 2008, pp. 302–316.

[41] F. Rosenberg, A. Michlmayr, and S. Dustdar, "Top-Down Business Process Development and Execution using Quality of Service Aspects," *Enterprise Information Systems*, pp. 459–475, November 2008.

[42] F. Rosenberg, C. Enzi, C. Platzer, and S. Dustdar, "Integrating Quality of Service Aspects in Top-Down Business Process Development using WS-CDL and WS-BPEL," in *Proceedings of the 11th Enterprise Computing Conference (EDOC'07), Annapolis, MD, USA*. IEEE Computer Society, 2007, pp. 15–26.

[43] M. Carbone, K. Honda, N. Yoshida, and R. Milner, "Structured Communication-Centred Programming for Web Serices," in *Proceedings of the 16th European Symposium on Programming (ESOP'07), Barga, Portugal*, 2007.

[44] *Web Services Policy Attachment*, http://www-128.ibm.com/developerworks/webservices/library/specification/ws-polatt/, W3C, 2004, uRL: http://www-128.ibm.com/developerworks/webservices/library/specification/ws-polatt/ (Last accessed: May 9, 2007).

[45] D. Luckham, *The Power of Events: An Introduction to Complex Event Processing in Distributed Enterprise Systems*. Addison-Wesley Professional, May 2002.

[46] B. Wetzstein, S. Strauch, P. Majdik, and F. Leymann, "Modeling and Monitoring Process Performance Metrics of BPEL Processes," University of Stuttgart, Germany, Technical Report 2008/05, Juli 2008. [Online]. Available: http://www.informatik.uni-stuttgart.de/cgi-bin/NCSTRL/NCSTRL_view.pl?id=TR-2008-05&engl=0

[47] I. H. Witten and E. Frank, *Data Mining: Practical Machine Learning Tools and Techniques*, 2nd ed. Morgan Kaufmann, 2005. [Online]. Available: /bib/private/witten/DataMiningPracticalMachineLearningToolsandTechniques2ded-MorganKaufmann.pdf

[48] J. R. Quinlan, *C4.5: Programs for Machine Learning*. Morgan-Kaufmann, 1993.

[49] Y. Freund and L. Mason, "The Alternating Decision Tree Learning Algorithm," in *Proceedings of the 16th International Conference on Machine Learning (ICML '99)*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1999, pp. 124–133.

[50] J. Mingers, "An Empirical Comparison of Pruning Methods for Decision Tree Induction," *Machine Learning*, vol. 4, no. 2, pp. 227–243, 1989.

[51] R. Khalaf and F. Leymann, "Role-based Decomposition of Business Processes using BPEL," in *International Conference on Web Services (ICWS 2006)*. IEEE Computer Society, September 2006, pp. 770–780.

[52] M. Y. Vardi, "Branching vs. linear time: Final showdown," in *TACAS 2001: Proceedings of the 7th International Conference on Tools and Algorithms for the Construction and Analysis of Systems*. London, UK: Springer-Verlag, 2001, pp. 1–22.

[53] F. Wang, A. K. Mok, and E. A. Emerson, "Distributed real-time system specification and verification in aptl," *ACM Trans. Softw. Eng. Methodol.*, vol. 2, no. 4, pp. 346–378, 1993.

[54] M. P. Singh, "Distributed enactment of multiagent workflows: temporal logic for web service composition," in *AAMAS '03: Proceedings of the second international joint conference on Autonomous agents and multiagent systems*. New York, NY, USA: ACM, 2003, pp. 907–914.

[55] A. Gehlert, J. Hielscher, O. Danylevych, and D. Karastoyanova, "Online Testing, Requirements Engineering and Service Faults as Drivers for Adapting Service Compositions," in *ServiceWave 2008, MONA+*. Springer Berlin Heidelberg, Februar 2009.

[56] F. Curbera, F. Leymann, T. Storey, D. Ferguson, and S. Weerawarana, *Web Services Platform Architecture: SOAP, WSDL, WS-Policy, WS-Addressing, WS-BPEL, WS-Reliable Messaging and More*. Prentice Hall PTR, 2005.

[57] D. Karastoyanova, F. Leymann, J. Nitsche, B. Wetzstein, and D. Wutke, "Utilizing Semantic Web Service Technologies for Automatic Execution of Parameterized BPEL Processes," in *XML Tage 2006*. Unbekannt, September 2006.

CD – JRA – 2.2.2

Paper [1]:

Wetzstein B., Leitner P., Rosenberg F., Brandic, F., Dustdar, S., Leymann F.:

Monitoring and Analyzing Influential Factors of Business Process Performance

Monitoring and Analyzing Influential Factors of Business Process Performance

Branimir Wetzstein[1], Philipp Leitner[2], Florian Rosenberg[2], Ivona Brandic[2], Schahram Dustdar[2], and Frank Leymann[1]

[1] Institute of Architecture of Application Systems, University of Stuttgart, Germany
Universitätsstraße 38, 70569 Stuttgart, Germany
lastname@iaas.uni-stuttgart.de
[2] Distributed Systems Group, Vienna University of Technology
Argentinierstrasse 8, 1040 Wien, Austria
lastname@infosys.tuwien.ac.at

Abstract. Business activity monitoring enables continuous monitoring of key performance indicators (KPIs). However, if things go wrong, a deeper analysis of process performance becomes necessary. Business analysts want to learn about the factors that influence the performance of business processes and most often contribute to the violation of KPI target values, and how they relate to each other. We provide a framework for performance monitoring and analysis of WS-BPEL processes, which consolidates process events and Quality of Service measurements. The framework uses techniques from the area of machine learning in order to construct tree structures, which represent the dependencies of a KPI on process and QoS metrics. These dependency trees allow business analysts to analyze how the process KPIs depend on lower-level process metrics and QoS characterisitics of the IT infrastructure. Deeper knowledge about the structure of dependencies can be gained by drill-down analysis of single factors of influence.

1 Introduction

Business Process Management (BPM) encompasses a set of methods, techniques, and tools for modeling, executing and analyzing business processes of an organization [1]. Recently, BPM has been supported by a set of tools which have been integrated in order to support the business process lifecycle in a unified manner. Thereby, business analysts create a business process model, which is then refined by IT engineers to an executable model. The executable process model is deployed to a process engine, which executes the process by delegating tasks to humans and services. The execution of processes is often based on a Service Oriented Architecture [2] (SOA). In that case, the business process model is typically implemented as a service composition, for example in WS-BPEL.

An important aspect of the BPM lifecycle is the continuous supervision of business goals and timely measurement of business process performance. This is typically supported by business activity monitoring (BAM) technology, which

enables continuous, near real-time monitoring of processes based on an eventing infrastructure [3]. Analysts define Key Performance Indicators (KPIs) and their target values based on business goals (e.g., "order fulfillment lead time < 3 days"). KPIs are influenced by a set of Process Performance Metrics (PPM) [4], which are metrics based on process runtime data (e.g., "number of orders which can be served from inhouse stock"). PPMs are on a different level of granularity than KPIs: a KPI measures the success of the process as a whole, while a PPM captures only a single facet of the process, which is usually not interesting in isolation. Additionally, KPIs are influenced by technical parameters, i.e., the Quality of Service (QoS) metrics of the SOA (e.g., the availability of the process engine or the response time of Web services).

Business Activity Monitoring provides useful information on KPI achievement. However, the focus is set on the "what" rather than the "why" question. When KPIs do not meet target values the business analysts are interested in factors that cause these deviations. Since KPIs potentially depend on numerous lower-level PPMs and QoS metrics, these causes can be manifold, and are rarely obvious even to domain experts. In this paper we present an integrated framework for run-time monitoring and analysis of the performance of WS-BPEL processes. Our main contribution is the presentation of a framework for dependency analysis, a machine learning based analysis of PPMs and QoS metrics, with the ultimate goal of discovering the main factors of influence of process performance (i.e., KPI adherence). These factors are represented in an easy-to-interpret decision tree (dependency tree). We present the general concepts of our analysis framework, and provide experimental results based on a purchase order case study, identify cases when dependency trees do not show expected results, and explain strategies how these problems can be coped with.

The rest of the paper is organized as follows: in Section 2 we present a case study which we use for explaining our concepts and for experimentation and explain the research issues that this paper deals with in more detail, Section 3 explains the main ideas of runtime monitoring and dependency analysis, Section 4 describes the implementation of our prototype based on the case study and experimental results, which are also extensively interpreted, Section 5 discusses important related work, and Section 6 finally concludes the paper and presents some future research directions.

2 Case Study

In this section we present a case study which we use in the following sections for explaining our concepts and which we have implemented and used for experimentation purposes. We have chosen a purchase order scenario consisting of a customer, a reseller, its two suppliers, a banking service, and a shipping service. The reseller offers certain products to its customers. It holds a certain part of the products in stock and orders missing products from suppliers if necessary. The customer sends a purchase order request with details about the required products and needed amounts to the reseller. The latter checks whether all products

are available in stock in the warehouse. If some products are not in stock, they are ordered from suppliers. Note that the second supplier is contacted only if the first (preferred) supplier is not able to deliver. If the purchase order can be satisfied, the customer receives a confirmation, otherwise the order is rejected. The reseller waits, if needed, for the supplier to deliver the needed products. When all products are in place, the warehouse packages the products and hands them over to the shipment service, which delivers the order to the customer, and finally notifies the reseller about the shipment. In parallel to the packaging and shipment, the payment subprocess is performed. Thereby, the customer decides on the payment style and gives its payment details depending on the payment style. The reseller contacts a banking service which authorizes the customer and credits the agreed amount. From the point of view of the reseller, a typical KPI is the *order fulfillment lead time* (duration from receiving the customer order until shipment is received by the customer), as defined in the Supply-Chain Reference Model (SCOR) [5].

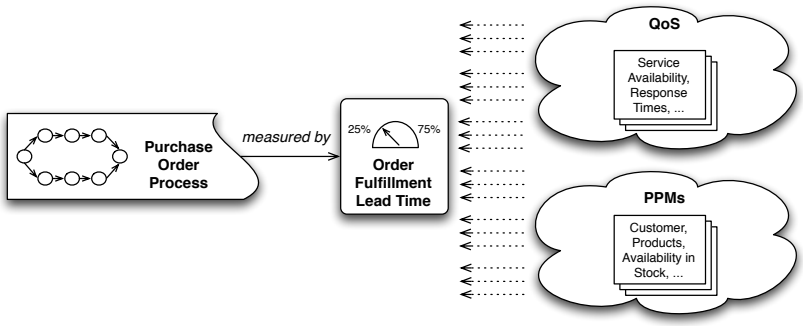


Fig. 1. KPIs, PPMs and QoS metrics

Assuming that this process is implemented using WS-BPEL, the KPI *order fulfillment lead time* is potentially influenced by a number of technical and non-technical factors, such as the response time and availability of Web services, the customer, or the products ordered (Figure 1). In Table 1, we have provided an (incomplete) list of potential factors of influence for the KPI from our case study. Factors can include simple facts from the business process instance, such as a customer identifier, a product type, or information about which branch of a process has been executed (e.g., whether the alternative branch "ordering from external suppliers" needed to be executed). All these facts are accessible from the process instance, therefore, we have not given a calculation formula for these PPMs in the table. However, PPMs on a different level of granularity are also possible, such as the duration of a whole subprocess. Finally, we have given a few simple examples of QoS metrics, which may influence the KPI performance, such as the availability of the process engine or single services that the process

relies on, or the response time of these services. A full discussion of possible QoS metrics is out of scope of this paper. The interested reader may refer e.g., to [6] for a more complete description of possible QoS metrics and their measurement. For completeness, we have also provided the possible range for these example influential factors. Generally, factors of influence can either be nominal values (i.e., take on one of a finite number of predefined values), or numeric values (e.g., integer or real values). Analysts can also define target values for metrics. In the table we have also given target values for the QoS metrics, and for one of the PPMs (the duration of the payment subprocess).

Name	**Type**	**Calculation Formula**	**Target Value**	**Range**
Customer ID	PPM			$\{Customer_1, Customer_2, \ldots\}$
Product Type	PPM			$\{Product_1, Product_2, \ldots\}$
Shipped from stock	PPM			$\{true, false\}$
Duration of Payment Subprocess	PPM	$t_{end} - t_{begin}$	< 4000 ms	$[0; \infty]$
Availability Process Engine	QoS	$\frac{\#available}{\#checks}$	> 0.999	$[0; 1]$
Availability Banking Service	QoS	$\frac{\#available}{\#checks}$	> 0.99	$[0; 1]$
Response Time Banking Service	QoS	$t_{end} - t_{begin}$	< 100 ms	$[0; \infty]$

Table 1. Potential Factors of KPI Performance

However, it is not obvious even to experts which of these factors actually influence the KPI most, and what the structure of the dependencies between factors of influence is (i.e., some factors are in turn influenced by others, such as the duration of the payment subprocess which is again influenced by service response times). These questions are not answered sufficiently by today's BAM dashboards – they can only provide status information about KPIs, but do not allow further analysis of the main causes for violations. Our approach supports this kind of analysis, which we refer to as dependency analysis (i.e., the analysis of dependencies of KPIs to PPMs and QoS metrics). Furthermore, more detailed information about internal dependencies between factors of influence can be gained by drill-down analysis, i.e., recursively applying dependency analysis to single factors of influence. This way, detailed insight into the nature of the factors that influence process performance can be learned by a business analyst.

3 Framework

In this section we describe the concepts of our framework for monitoring and analyzing factors of influence of business process performance. A high-level overview

of the main components is given in Figure 2. In our framework we distinguish three different layers. In the *process runtime* layer, a WS-BPEL business process is defined and executed. The process can be executed in a standard WS-BPEL compliant engine, as long as the engine is able to emit the process information necessary for calculating PPMs in form of process events, as expected by BAM. In the *monitoring layer*, information about the running business process and the services it interacts with is collected in order to monitor KPIs, PPMs, and QoS metrics. Note that we assume that the user has defined a set of potential influential factors he wants to monitor for a KPI in order to ensure that corresponding metric data is available later on for analysis. Based on these metric definitions, the QoS monitor and the WS-BPEL engine emit needed events into a Complex Event Processing [7] (CEP) event cloud. Our monitoring tool extracts and correlates these events from the cloud and calculates corresponding PPMs and QoS metrics. The evaluated metrics are displayed in the BAM dashboard and are stored in the metrics database for later analysis. In the *process analysis layer*, the collected runtime information is analyzed by the process analyzer component. Outcomes of the analysis are again displayed in the dashboard to the users of the system, which can use this resulting information to optimize the business process.

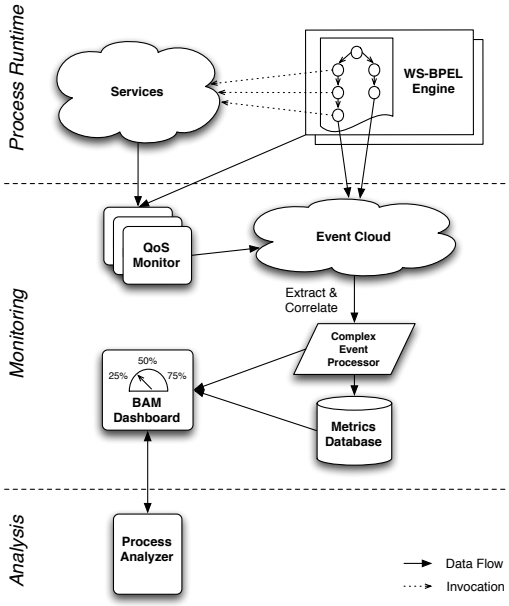


Fig. 2. Monitoring and Analysis Framework Overview

3.1 Monitoring of Influential Factors

In order to be able to analyze the influential factors of process performance in the analysis component, we first need to monitor the needed metrics. We distinguish in our approach between PPMs and QoS metrics, which are supported by different monitoring mechanisms.

PPM Monitoring PPMs are evaluated based on process events emitted by a WS-BPEL engine [4]. Most engines support such a mechanism, which in most cases is also to some extent configurable on which events to publish (via event filters). Based on the PPM metrics that have to be monitored, an event filter for the WS-BPEL engine is created, which specifies the events needed for the calculation. The monitoring tool subscribes to these events and evaluates them at process runtime as they are received. This is done by "querying" the event cloud. Note that the correlation of process events of the same process instance (needed for example for calculation of the duration of an activity) is done based on a unique process instance identifier, which is part of each event emited by the WS-BPEL engine. Note also that we do not yet support process data which comes from external data sources, in which case we would need additional correlation mechanisms.

QoS Metrics Monitoring QoS metrics are evaluated based on QoS events. These QoS events are emitted either by an external QoS monitor or by an instrumentation of the WS-BPEL engine itself. At the moment, we focus on monitoring of the QoS characteristics availability, response time, and accuracy (defined as 1 - # failed requests/# total requests). In our approach, we use an external QoS monitor for measuring the availability of the process engine and partner Web services of the WS-BPEL process. The QoS monitor polls the corresponding endpoints and emits events which contain information on their availability at a certain point in time. An approach to measuring response time and accuracy of partner Web services of a process is to instrument the Web service invocation module of the WS-BPEL engine. That means that as part of the execution of a WS-BPEL invoke activity, QoS events are emitted which contain information on the measured QoS data (response time and in case of accuracy whether the request was successful or failed).

Correlation of PPMs and QoS Metrics There is a technical difference between an external QoS monitor and a WS-BPEL engine internal instrumentation considering the correlation of process events and QoS events. As the WS-BPEL instrumentation is internal to the engine, it has access to the context of the process instance. Thus, it is possible to write the process instance identifier as an attribute into the QoS event which can then be used to correlate with process events of the same process instance. In case of an external QoS monitor this is not possible. Assume the KPI *order fulfilment lead time* and its evaluation for a specific process instance, i.e., a specific customer order. Assume also that we have defined *availability of process engine* as a potential influential factor of

this KPI. The QoS monitor collects availability data in a certain time period in which many process instances (customer orders) are processed. However, as a specific order is processed in a certain time frame, we have to calculate the *availability of the process engine* in the very same time frame, because outside of this time frame the availability of the engine has no impact on the KPI value of that process instance. This is why we need to correlate these metrics based on time frames (and not process instance identifier).

3.2 Analysis of Influential Factors

We monitor process metrics mainly with the goal of being able to later on perform dependency analysis on this data. The main idea of dependency analysis is to use historical process instances to determine the most important factors that dictate whether a process instance is going to violate its KPIs or not. The input of this analysis are stored process instances (i.e., the outcome of historical process instances, and the measured metrics associated with this instance), which are available in the metrics database. The output of dependency analysis is a decision tree that incorporates the most important factors of influence of process performance. We refer to this tree as *dependency tree*, because it represents the main dependencies of the business process on technical and process metrics, i.e., the metrics which contribute "most often" to the failure or success of a process instance.

Dependency Trees and Drilldown We give an example dependency tree in Figure 3. In this (simple) example, the most influential factor is the response time of the banking service, since a delay in this service generally leads to a violated KPI. However, even if the banking services response time is acceptable (below 210 time units in this example), the KPIs are still often violated if the order is placed by the customer with the ID '1234'. Business analysts can use the dependency tree to learn about the "hot spots" of the process, and inform themselves about possible corrective actions if a process underperforms. For example, considering the example in Figure 3, a business analyst can take the corrective action to replace the "Banking Service" against a service with better response time, if such a service is available.

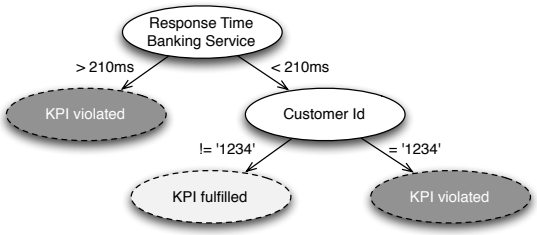


Fig. 3. Example Dependency Tree

If the metrics that have been identified as factors of influence have target values assigned, a further "drill down" analysis can be performed. For this, one of the factors of influence (e.g., the response time of the banking service) is selected, and another dependency analysis is launched. This identifies the more detailed dependencies that influence this specific factor of the overall process performance (e.g., one could find out that way that the response time of the banking service strongly depends on the type of the banking account).

Learning Dependency Trees The process analyzer uses decision tree classification, a well-known technique from the area of machine learning [8], to automatically learn dependency trees from existing process instances. Decision tree classifiers are standard machine learning technique for supervised learning (i.e., concepts are learned from historical classifications, in our case dependency information is learned from monitoring of previous process instances). Decision trees follow a "divide and conquer" approach to learning concepts – they iteratively construct a tree of decision nodes, each consisting of a test; leaf nodes typically represent a classification to a category. In our case, only two categories exist (KPI has been violated, or not). The process analyzer is flexible in the concrete algorithm that is to be used to construct the decision tree (e.g., C4.5 [9], or alternating decision tree [10]) and the parameterization (e.g., whether to use pruning [11] or not). However, data preprocessing (i.e., the process of reformatting raw process data in such a way that it can be fed into the machine learning algorithm) is done automatically by the dashboard. For learning decision trees we use 10-fold cross validation [8] to avoid having to split our historic process data into training, test and validation sets, and to estimate the classification error of the decision tree. The classification error allows the business analyst to measure the quality of the dependency tree, i.e., how exact the tree represents the actual structure of real-world dependencies.

4 Evaluation

In this section we describe our prototype implementation of the framework and experimental results based on the case study.

4.1 Case Study Implementation

We have implemented the case study as presented in Section 2 using a Java-based prototype. Many components of our solution are existing open source tools, which we describe further below. For experimentation, we have deployed all these components on a single desktop PC, mainly to prevent external influences such as network latency to influence our experimentation results.

Process Engine We use Apache ODE[3] as our business process execution engine. ODE is open source software, and implements the WS-BPEL standard for Web

[3] `http://ode.apache.org/`

service orchestration. One of the features of ODE is that it can trigger execution events (e.g., `ActivityExecStartEvent`), which we use as process events as described in Section 3. For hosting ODE we use the Apache Tomcat[4] application server. The eventing features of ODE demand for a JMS implementation to take care of the transportation of events to subscribers. We chose to use the open source message queue Apache ActiveMQ[5], however, any other JMS implementation could be used as well.

Purchase Order Process The purchase order process has been implemented as a WS-BPEL process which interacts with six Web services including the client of the process. These Web services have been implemented in Java using Apache CXF[6] and simulate certain influential factors. One can, for example, configure the response time, availability, and outputs of a service over time and dependant on business process data.

Database Server The metrics database is implemented as a standard MySQL[7] database. Because of the limited size of the case study we did not use advanced features such as clustering or load balancing.

Monitoring Tool and Dashboard The Monitoring Tool for evaluation of PPMs has been implemented in Java as described in [4]. We have additionally implemented support for correlation of PPMs and QoS metrics. The Dashboard component is implemented as an Eclipse-Plugin.

Process Analyzer The process analyzer is implemented as a standalone Web service, which is accessible over a RESTful interface [12]. The foundation of this component is the WEKA toolkit[8]. WEKA implements many high-quality machine learning schemes, including the decision tree based classifiers that we used in this paper. We have transparently integrated WEKA into our process analyzer component using the WEKA Java API.

QoS Monitor For experimentation we have implemented a naive Quality of Service monitor, which is able to non-intrusively check a limited number of typical QoS metrics.

4.2 Experimental Results

The procedure of experimentation is as follows. We create a configuration which simulates certain influential factors and define a set of potential influential metrics. We then execute the process a certain number of times (100, 400, and 1000

[4] `http://tomcat.apache.org/`
[5] `http://activemq.apache.org/`
[6] `http://cxf.apache.org/`
[7] `http://www.mysql.com/`
[8] `http://www.cs.waikato.ac.nz/ml/weka/`

times) by triggering the process using a simulated client. During execution, the process is monitored and metrics are saved in the metrics database. We then perform dependency analysis of the KPI and compare the result of the generated dependency tree with our configured influential metrics. In the following we present the results of two experimental runs. For both of them, we have used the same configuration consisting of the KPI *Order Fulfillment Lead Time* and a set of 31 potential influential factors (a subset is shown in Table 1).

For the first run, we have created a configuration which simulates the following factors: (i) the warehouse availability check (*order in stock*) returns a negative result for certain *product types* based on certain probabilities; *order in stock* is an important influential factor of the overall duration of the process as it decides whether products have to be requested from suppliers which increases the overall process duration substantially (ii) supplier 1 has in average a higher than expected *supplier delivery time*; (iii) average *shipment delivery time* is high in relation to the overall duration of the process instance. Based on this configuration, we expect the KPI to be mainly influenced by *order in stock*, *product type*, *supplier 1 delivery time*, and *shipment delivery time*. Other metrics (in particular response times of services) also influence the KPI value but in a marginal way.

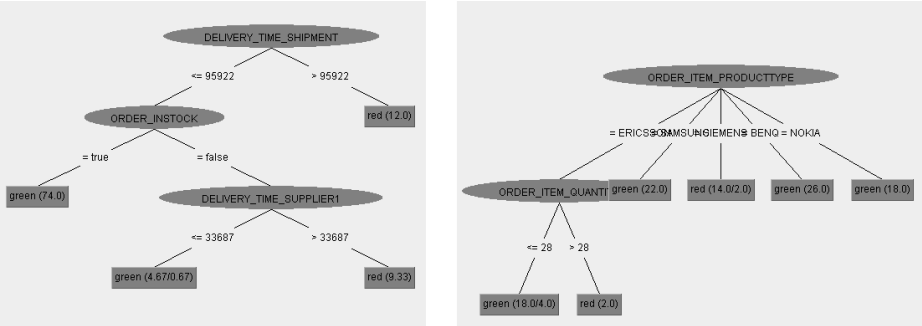


Fig. 4. Generated Trees for (a) *Order Fulfillment Time*, (b) *Order in Stock*

The generated decision tree is shown in Figure 4a. It has been generated by the J48 (the WEKA implementation of C4.5 [9]) based on 100 process instances. The most influential factor is the *shipment delivery time*; if it is above 95 time units all process instances lead to KPI violations ("red"), otherwise they depend further on the *order in stock* metric and *supplier 1 delivery time*. The leaves of the tree show the number of instances which are classified as "red" or "green".

The dependecy tree shows three of the four influential factors we have configured. Interestingly, the fourth factor, the *product type*, is not shown. One can explain this result as follows: *product type* directly influences *order in stock* which again influences the KPI value which is shown in the tree; as both metrics influence the KPI value in the same way, only one of them is shown in the tree.

This particular result is unsatisfactory, as it hides the root cause, namely *product type* in this case. The user can deal with this problem using two approaches: (i) he can drill down and request the analysis of the *order in stock* metric. A second tree is generated which explains when ordered products are not in stock as shown in Figure 4b. This tree now clearly shows how the unavailability depends on *product type* and *ordered product quantity*. (ii) He can remove the *order in stock* metric temporarily from the analyzed metric set. Now, the algorithm will search for alternative metrics which classify the instances in a similar way as *order in stock*. In that case, as shown in the experiments, the algorithm finds and displays *product type* in the tree (not shown).

Instances	**Algorithm**	**Leaves/ Nodes**	**Displayed Metrics Expected/All**	**Correctly Classfied**
100	J48	4/7	3/4	95,0 %
100	ADTree	11/16	2/4	98,0 %
400	J48	6/11	3/4	97,8 %
400	ADTree	17/26	4/5	99,0 %
1000	J48	11/18	3/6	98,8 %
1000	J48 -R	6/11	3/4	97,9 %
1000	J48 -U	13/22	4/9	99,2 %
1000	ADTree	19/28	3/6	99,4 %

Table 2. Experimental Results

Table 2 shows the more detailed results of the first experimental run. We have experimented with two algorithms: J48 (based on C4.5 [9]) and ADTree (alternating decision tree [10]). Both of them show very similar results concerning the displayed influential metrics. Typically there is only one or at most two (marginal) metrics which differ. For the same precision (correctly classified instances in the training set, as shown in the last column), the algorithms also generate trees of about the same size. The usage of parameters has lead to only marginal changes in our experiments (for example, J48 -U with no pruning). The only parameter that turned out useful in our experimentation was the "reduced error pruning" (J48 -R) [8] as it reduced the size of the tree, loosing accuracy only marginally. This parameter is useful as the experiments show that the tree is getting bigger (column "Leaves/Nodes") with the number of process instances. For example, J48 generated for 400 instances a tree with 11 nodes, for 1000 instances a tree with 18 nodes, while the precision improved only by 1 %. In particular, when the tree gets bigger, factors are shown in the tree which have only marginal influence and thus make the tree less readable; column "Displayed Metrics" shows how many distinct metrics are displayed in the tree, the

first number thereby depiciting the number of expected metrics. In the case of too many undesirable (marginal) metrics, one can try to improve the result by simply removing those metrics from the analyzed metric set and repeating the analysis. Finally, concerning the analysis duration, in our setting on a standard laptop computer a decision tree generation based on 1000 instances takes about 30 seconds.

For the second run, we have created a configuration which, among others, simulates QoS influential factors: (i) the warehouse Web service and the shipment Web service are unavailable with the probability of 15%; the BPEL process contains fault handlers when trying to invoke partner services; in case of unavailability it waits for a certain time frame and retries; we have defined *response time* metrics (measured based on process events) for each invoke-activity which "include" the retries in case of unavailability (ii) the warehouse availability check (*order in stock*) returns now a negative result only with the probability of 5%; (iii) *shipment delivery time* is still very influential in relation to the duration of other activities of the process. Based on this configuration, we expect the KPI to be mainly influenced by the *availability* of warehouse Web service and shipment Web service, *order in stock*, and *shipment delivery time*.

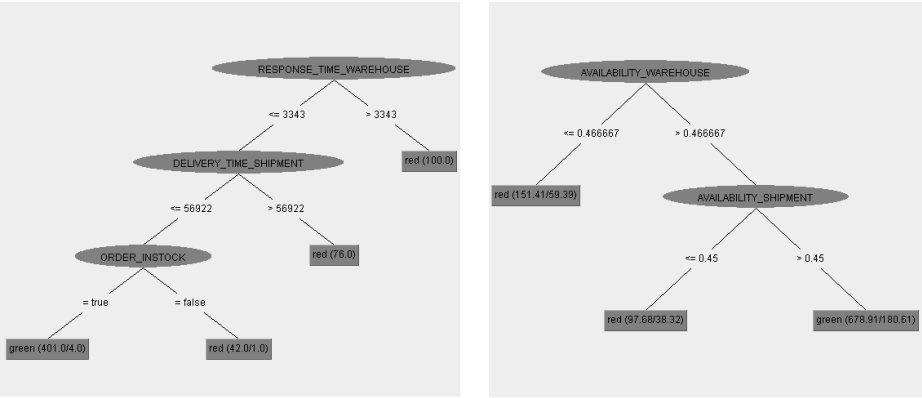


Fig. 5. Generated Trees for (a) All Factors, (b) Availability of IT Infrastructure

The generated decision tree is shown in Figure 5a. It is a J48 tree based on 1000 instances using reduced error pruning. The tree shows the *response time warehouse*, *delivery time shipment*, and *order in stock* as the main influential factors. Completely missing, however, are the expected dependencies on the availability of the warehouse Web service and the shipment Web service. We suspect that *availability of warehouse Web service* is hidden by *response time warehouse*, which could be analyzed by drilling down. However, we take now another approach and perform an analysis of the KPI only in relation to availability metrics of all services involved in the process, i.e. we remove all other metrics from the analyzed metric set. Effectively, we analyze the impact of avail-

ability on the KPI. The result is shown in Figure 5b. It clearly shows that (only) availability of the shipment and warehouse Web services have an impact on the KPI value, as expected.

Overall, we can draw following conclusions from the experiments. In general, the generated trees show the expected influential metrics in a satisfactory manner. Concerning the influential factors displayed in the tree, we have identified two problems: (i) as the tree gets bigger it contains often more metrics than expected, i.e. metrics which have only marginal influence and thus only "blur the picture"; in that case one can try to tune the algorithm by using, for example, reduced error pruning, or one can simply remove those metrics from the analyzed metric set and repeat the analysis; both techniques lead to more satisfactory results; (ii) the tree does not show some of the expected metrics: we have shown that this is often the case when there are "multi-level" dependencies between metrics; in that case further analysis (drill down) of lower-level metrics may help to find further influential factors. Note, however, that drill down functionality assumes that the metric which is to be analyzed has a target value assigned.

5 Related Work

There are several approaches that deal with monitoring of service compositions. They differ mostly in monitoring goals, i.e., what is monitored, and the monitoring mechanisms. IBM's approach integrates performance management tightly into the business process lifecycle and supports it through its WebSphere family of products [13]. Thereby, process metrics are modeled and monitored based on events published by the Process Server BPEL engine. Schiefer et al. [14] extend a WS-BPEL process definition with auditing activities in order to publish state changes by invoking operations on the monitoring tool. Our approach is similar to IBM's approach in that we use process events published by the process engine. We, however, also support monitoring of QoS metrics. Additionally, automated dependency analysis is not dealt with by the IBM solution. Baresi et al. [15] deal with monitoring of WS-BPEL processes focusing on runtime validation. The goal is thereby not to monitor process performance metrics, but to detect partner services which deliver unexpected results concerning functional expectations. Traverso et al. [16] describe a monitoring approach for WS-BPEL processes which supports run-time checking of assumptions under which the partner services are supposed to participate in the process and the conditions that the process is expected to satisfy. The approach supports also collecting statistical and timing information. All of these approaches have in common that they concentrate on monitoring of WS-BPEL processes only. In particular, they do not deal with QoS metrics integration and dependency analysis.

When it comes to the analysis aspect, most closely related to our work is iBOM, a platform for business operation management developed by HP [17, 18], as it supports both process monitoring, and analysis and prediction based on data mining. In [17] the authors give an overview and a classification of which data mining techniques are suitable for which analysis and prediction

techniques. Thereby, also decision trees are mentioned which we concentrate on in our approach. In [18], the iBOM platform is presented which allows users to define and monitor business metrics (not focused on WS-BPEL processes), perform intelligent analysis on them to understand causes of undesired metric values, and predict future values. Our approach is different in that we focus on SOA-based WS-BPEL processes, and explicitly integrate PPMs and QoS metrics for analysis purposes. We deal only with decision trees, but provide detailed experimental results. Another popular approach to process analysis is process mining. Process mining techniques operate on event logs provided by information systems and perform different kinds of analysis on them, in particular process discovery when there is no explicit process model a priori [19]. In our approach, we also operate on monitored "metric logs" during analysis phase, but focus on mining of metric dependencies using decision tree algorithms.

6 Conclusions and Future Work

In this paper we have presented a framework that performs monitoring of both PPMs and QoS metrics of business processes runnning on top of a Service-Oriented Architecture. Besides providing up-to-date dashboard information about the current process performance, the main goal of our framework is to enable what we refer to as dependency analysis, i.e., an analysis of the main factors that influence the business process and make it violate its performance targets. The result of this analysis is represented as a decision tree. We have presented experimental results which show that in general the generated decision trees provide explanations in a satisfactory manner, but in some cases further analysis has to be done. In that respect, we have shown how drill-down functionality and analyis based on different metric sets can influence the analysis result.

Our future work includes extending the framework presented here into various directions. Firstly, we plan to work on the runtime prediction of the outcome of process instances (i.e., whether the KPI is going to be violated or not) while they are still running. Basically, we can us the same techniques as for dependency analysis (however, we will in addition use regression trees). Secondly, we are working towards making use of the dependency analysis in the area of process adaptation – currently, dependencies are presented towards the human business analyst, who is then incorporating the gained knowledge back into the process, e.g., by exchanging service bindings. We currently think about a more automated feedback mechanism, which uses rule sets and predefined reactions to incorporate dependency knowledge back into the WS-BPEL process in a (semi-)automated way. One example would be service selection: if the dependency model of a process shows that the process outcome is sensitive to the response time of a service, then an expensive high-quality service is selected; if the response time is no important factor of influence a cheaper service is selected.

References

1. Weske, M.: Business Process Management: Concepts, Languages, Architectures. Springer-Verlag New York, Inc., Secaucus, NJ, USA (2007)
2. Papazoglou, M.P., Traverso, P., Dustdar, S., Leymann, F.: Service-Oriented Computing: State of the Art and Research Challenges. IEEE Computer **11** (2007)
3. Jeng, J.J., Schiefer, J., Chang, H.: An Agent-based Architecture for Analyzing Business Processes of Real-Time Enterprises. In: Proceedings of the 7th International Conference on Enterprise Distributed Object Computing (EDOC '03), Washington, DC, USA, IEEE Computer Society (2003) 86
4. Wetzstein, B., Strauch, S., Majdik, P., Leymann, F.: Modeling and Monitoring Process Performance Metrics of BPEL Processes. Technical Report 2008/05, University of Stuttgart, Germany (2008)
5. Council, S.: Supply Chain Operations Reference Model Version 7.0 (2005)
6. Rosenberg, F., Platzer, C., Dustdar, S.: Bootstrapping Performance and Dependability Attributes ofWeb Services. In: Proceedings of the IEEE International Conference on Web Services (ICWS '06), Washington, DC, USA, IEEE Computer Society (2006) 205–212
7. Luckham, D.: The Power of Events: An Introduction to Complex Event Processing in Distributed Enterprise Systems. Addison-Wesley Professional (2002)
8. Witten, I.H., Frank, E.: Data Mining: Practical Machine Learning Tools and Techniques. 2 edn. Morgan Kaufmann (2005)
9. J. R. Quinlan: C4.5: Programs for Machine Learning. Morgan-Kaufmann (1993)
10. Freund, Y., Mason, L.: The Alternating Decision Tree Learning Algorithm. In: Proceedings of the 16th International Conference on Machine Learning (ICML '99), San Francisco, CA, USA, Morgan Kaufmann Publishers Inc. (1999) 124–133
11. Mingers, J.: An Empirical Comparison of Pruning Methods for Decision Tree Induction. Machine Learning **4**(2) (1989) 227–243
12. Fielding, R.T.: Architectural Styles and the Design of Network-based Software Architectures. PhD thesis, University of California, Irvine, CA (2000)
13. Wahli, U., Avula, V., Macleod, H., Saeed, M., Vinther., A.: Business Process Management: Modeling Through Monitoring Using WebSphere V6.0.2 Products. IBM, International Technical Support Organization (2007)
14. Roth, H., Schiefer, J., Schatten., A.: Probing and Monitoring of WSBPEL Processes with Web Services. Proceedings of the The 8th IEEE International Conference on E-Commerce Technology (CEC-EEE'06) (2006)
15. Baresi, L., Guinea, S.: Towards Dynamic Monitoring of WS-BPEL Processes. In: Proceedings of the 3rd International Conference of Service-Oriented Computing (ICSOC'05), Springer (2005) 269–282
16. Barbon, F., Traverso, P., Pistore, M., Trainotti, M.: Run-Time Monitoring of Instances and Classes of Web Service Compositions. In: Proceedings of the IEEE International Conference on Web Services(ICWS'06). (2006) 63–71
17. Castellanos, M., Casati, F., Dayal, U., Shan, M.C.: A Comprehensive and Automated Approach to Intelligent Business Processes Execution Analysis. Distributed and Parallel Databases **16**(3) (2004) 239–273
18. Castellanos, M., Casati, F., Shan, M.C., Dayal, U.: iBOM: A Platform for Intelligent Business Operation Management. In: Proceedings of the 21st International Conference on Data Engineering (ICDE'05). (2005) 1084–1095
19. van der Aalst, W., Weijters, T., Maruster, L.: Workflow mining: Discovering process models from event logs. IEEE Trans. on Knowl. and Data Eng. **16**(9) (2004) 1128–1142

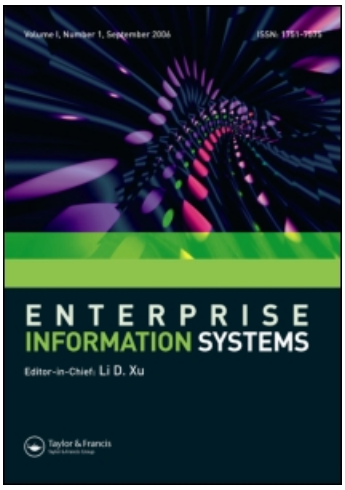
CD – JRA – 2.2.2

Paper [2]

Rosenberg F., Michlmayr A., Dustdar S.:

Top-down business process development and execution using quality of service aspects

In: *Enterprise Information Systems, Vol. 2(4):459-475, Taylor & Francis, 2008.*



Enterprise Information Systems
Publication details, including instructions for authors and subscription information:
http://www.informaworld.com/smpp/title~content=t748254467

Top-down business process development and execution using quality of service aspects

Florian Rosenberg [a]; Anton Michlmayr [a]; Schahram Dustdar [a]
[a] Distributed Systems Group, Technical University of Vienna, Vienna, Austria

Online Publication Date: 01 November 2008

To cite this Article Rosenberg, Florian, Michlmayr, Anton and Dustdar, Schahram(2008)'Top-down business process development and execution using quality of service aspects',Enterprise Information Systems,2:4,459 — 475
To link to this Article: DOI: 10.1080/17517570802395626
URL: http://dx.doi.org/10.1080/17517570802395626

Enterprise Information Systems
Vol. 2, No. 4, November 2008, 459–475

Taylor & Francis
Taylor & Francis Group

Top-down business process development and execution using quality of service aspects

Florian Rosenberg*, Anton Michlmayr and Schahram Dustdar

Distributed Systems Group, Technical University of Vienna, Vienna, Austria

(*Received 27 January 2008; final version received 7 August 2008*)

Developing cross-organisational business processes is a challenging task. The partners have to agree on a common data format and meaning as well as on the quality of service (QoS) requirements each partner has to fulfil. The QoS requirements are typically described using service level agreements (SLAs) among the partners. In this paper, a top-down modelling approach for Web service based business processes is proposed to capture the functional and non-functional aspects using a choreography language (WS-CDL) which describes the message interactions among the participants. The choreography is annotated with SLAs for the different partners. For each partner in the process, an orchestration (in WS-BPEL) and the necessary Web service templates are automatically generated. The service level objectives (SLOs) from the partner SLAs are automatically translated into policies that can then be enforced by a BPEL engine during execution. The deployment of the WSDL files, the monitoring of QoS attributes and the execution of the BPEL process itself are then handled by the VRESCo SOA runtime.

Keywords: quality of service; top-down modelling; business process development; service level agreements; WS-CDL; WS-BPEL

1. Introduction

Service-oriented architecture (SOA) represents an emerging paradigm to develop flexible and large-scale software systems using the Internet as the main infrastructure. Web services are one realisation of this paradigm by using well-established standards to describe and interact with other services.[1]

Many organisations are building their cross-organisational business processes based on Web services because of their platform-agnostic nature and the ease of integration. Currently available technologies such as composition engines using the Web Service Business Process Execution Language (WS-BPEL, or BPEL for short; see Alves *et al.* 2007) can be used to orchestrate business processes within an organisation.

An engineering method for Web service based business processes involving multiple partners requires agreement on the data exchanged that cannot be achieved using BPEL. For this purpose, the Web Service Choreography Description Language (WS-CDL; see Kavantzas *et al.* 2008) provides an XML-based language to describe the cross-organisational message exchanges from a global viewpoint. The different views (local *versus* global) are described by the terms *choreography* and *orchestration*. Choreography

*Corresponding author. Email: florian@infosys.tuwien.ac.at

ISSN 1751-7575 print/ISSN 1751-7583 online

DOI: 10.1080/17517570802395626
http://www.informaworld.com

can be defined as 'processes involving multiple services where the interactions between these services are seen from a global perspective' (Kavantzas *et al*. 2008). A choreography does not describe any internal actions that occur within a participating service, such as internal computation or data transformation, but rather focuses on the observable public exchange of messages. In contrast to that, Peltz (2003) defines *orchestration* as an 'executable business process that can interact with both internal and external Web services. The interactions occur at the message level. They include business logic and task execution order, and they can span applications and organisations to define a long-lived, transactional, multistep process model'.

These two concepts imply that such a choreographic description can be used to generate the orchestration behaviour (e.g. in the form of BPEL stubs) and the necessary WSDL templates automatically. Nowadays, service level guarantees and obligations among the service providers are becoming increasingly important as a means to capture runtime quality requirements and guarantees for a partner's service (such as response time, availability and security). Such quality of service (QoS) requirements are generally specified in a service level agreement (SLA) and need to be fulfilled during the service execution among the partners. Using existing Web service standards and proposals there is currently no integrated modelling method available to build such cross-organisational business processes considering SLA requirements as first-class citizens from the starting point of the development process. Additionally, the orchestration parts and the WSDL files for each partner in the choreography should be automatically generated.

In this paper, we propose a top-down modelling approach to build such QoS-aware, Web service based business processes using currently available technologies such as WS-CDL and BPEL. It leverages a design approach for efficient development of cross-organisational business processes similar to the idea of model-driven software development (Völter and Stahl 2006). The novelty of our approach is twofold: (1) a relevance mapping from WS-CDL to BPEL and (2) the consideration of SLA requirements from the beginning of the choreography development process. These SLA requirements are then automatically transformed and mapped to a WS-QoS policy and attached to the BPEL process of the affected partner allowing policy-aware middleware to check and enforce the SLA.

This paper is organised as follows. Section 2 describes the case study we implemented for evaluating the concepts of this work. Section 3 describes our main approach for realising QoS-aware business process development. The implementation of this approach is sketched in Section 4 followed by an evaluation in Section 5. Section 6 positions our approach among existing work and, finally, Section 7 concludes the paper.

2. Case study

In this case study we developed a *build-to-order* (BTO) scenario in the B2B area. The use case consists of a customer, a manufacturer, and suppliers for CPUs, main boards and hard disks. The manufacturer offers assembled IT hardware equipment to its customers. For this purpose the manufacturer has implemented a BTO business model. It holds a certain part of the individual hardware components in stock and orders missing components if necessary. In the implemented BTO scenario, the customer sends a quote request with details about the required hardware equipment to the manufacturer. The latter sends a quote response back to the customer. As long as customer and manufacturer do not agree on the quote, this process will be repeated. If a mutual agreement is achieved, the customer sends a purchase order to the manufacturer. Depending on its hardware stock the manufacturer has to order the required hardware components from its suppliers.

If the manufacturer needs to obtain hardware components to fulfil the purchase order, an appropriate hardware order is sent to the respective supplier. In turn the supplier sends a hardware order response to the manufacturer. Finally, the manufacturer sends a purchase order response back to the customer.

The interactions of the participants in our BTO scenario[2] are illustrated in the collaboration sequence diagram shown in Figure 1. The BTO scenario consists of six different Web service invocations which correspond to the following SOAP operations: `requestForQuote`, `updateQuote`, `sendPurchaseOrder`, `orderCPU`, `orderMB`, `orderHD`. Each SOAP operation depicts a synchronous message request-reply scenario which will be illustrated as an exemplar for the `requestForQuote` operation. The customer invokes the operation `requestForQuote` on the service interface of the manufacturer sending the `QuoteRequest` message. The manufacturer receives the message request and replies to the service invocation by returning the `QuoteResponse` message. Contrary to this, an asynchronous message scenario would require additional callback operations on the service requestor side. In this case the manufacturer invokes an operation `requestForQuoteCallback` on the service interface of the customer to send back the `QuoteResponse`.

The definition of SLA and QoS plays a crucial rule in cross-organisational business processes. Each participant offers services to other partners over the Internet which the latter need to run their businesses. Therefore, a certain degree of reliability concerning response time, throughput, uptime, etc. is desired and has to be specified and explicitly expressed from the beginning of the modelling phase. In our scenario we distinguish four different relationships between the choreography participants. The customer interacts with

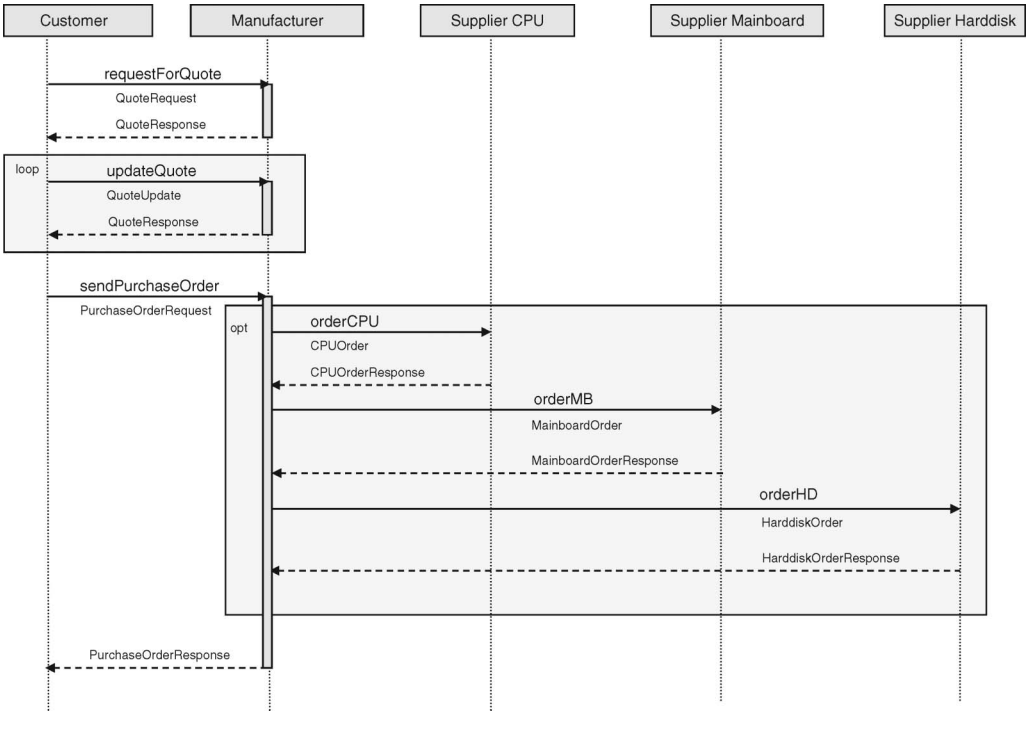


Figure 1. BTO case study.

the manufacturer; the manufacturer interacts with different suppliers. For each relationship an SLA is defined between the partners to regulate this degree the partners need for their business.

3. Top-down modelling approach

In this section, we describe our top-down modelling approach and the QoS integration. Due to space restrictions, we cannot describe WS-BPEL (Alves *et al.* 2007), WS-CDL (Kavantzas *et al.* 2008) and WS-Policy (Vedamuthu *et al.* 2008a) and therefore assume basic familiarity with the concepts and their syntax. For a short overview of these technologies, see Rosenberg *et al.* (2007).

3.1. Overview

As shown in Figure 2, the language constructs of WS-CDL can be mapped to BPEL allowing a choreographic description to be transformed into separate BPEL processes, one for each partner in the choreography, including corresponding WSDL descriptions.

Our approach consists of two main steps which are highlighted in this figure. In the first step, a number of models are defined at the highest level of abstraction (the choreography layer). On the one hand, this contains the WS-CDL document specifying the choreography in detail (*Step 1a*). The choreography is used to describe the partners in the process and the message exchanges. On the other hand, the SLAs between the participants are defined (*Step 1b*) which describe obligations and guarantees among the participants. The choreography is annotated with the SLA references to allow a pairwise agreement between two partners on a specific SLA. Furthermore, SLAs may also include QoS attributes as described in more detail below.

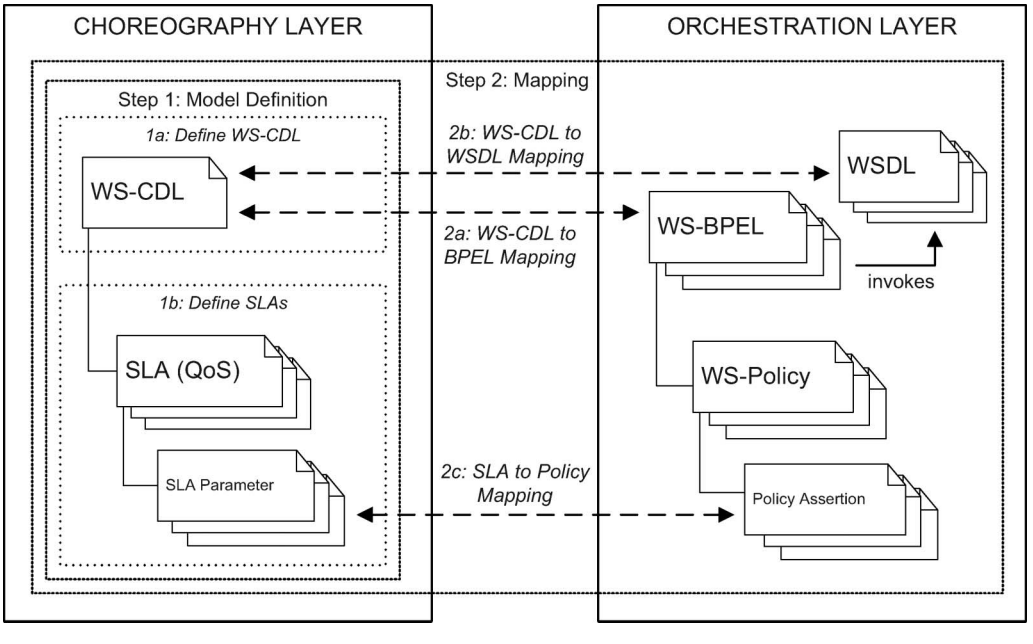


Figure 2. Modelling approach.

In the second step, the documents defined in the first step are transformed from the choreography layer into executable code in the orchestration layer. First, the WS-CDL choreography is mapped into a number of BPEL processes depending on the number of participants (*Step 2a*). Secondly, the WS-CDL document is used to generate WSDL descriptions of those Web services, which each participant has to implement and provide to its business partners (*Step 2b*). The details of mapping WS-CDL to BPEL and WSDL are described in Section 3.2.

The importance of QoS in cross-organisational business processes makes it necessary to consider these aspects from the beginning of the development process. Therefore, the SLAs are transformed to WS-QoSPolicy statements, which is our extension to WS-Policy, that are directly attached to the corresponding partner links in BPEL (*Step 2c*). The details of this transformation step, the integration of QoS and WS-QoSPolicy are given in Section 3.3.

3.2. *Mapping WS-CDL to BPEL and WSDL*

The main goal of transforming WS-CDL to BPEL is to allow the participants a rapid modelling and development process and generate relevant BPEL and WSDL documents which can then be used as a basis to implement the private (non-visible) business logic. The projection of such a global description to endpoint processes whose interactions precisely realise the global description is called *endpoint projection* (Carbone *et al*. 2007).

Mendling and Hafner (2007) define basic mapping rules from WS-CDL to BPEL. They use a recursive XSLT-based approach to generate the BPEL processes by iterating through each role type to check the relevance of the node. The authors consider a node as relevant if it contains activities with the attribute `toRoleTypeRef` and `fromRoleTypeRef`. However, this approach does not correspond with the endpoint projection definition given above, because more structured BPEL elements are generated than necessary. This is due to the fact that all parent nodes are considered during the mapping process even if they are not directly relevant (it can be considered as a simple 1:1 mapping). Listing 1 depicts an

```
1  <package>
2    <choreography>
3       <sequence>
4        <!-- ... -->
5        <sequence>
6          <!-- ... -->
7          <sequence>
8            <interaction operation="sendPO" ...>
9              <participate fromRoleTypeRef="Customer"
10                          toRoleTypeRef="Manufacturer" .../>
11             <exchange action="request" ...>
12               <!-- ... -->
13             </exchange>
14           </interaction>
15           <interaction operation="sendPO" ...>
16             <participate fromRoleTypeRef="Customer"
17                          toRoleTypeRef="Manufacturer" .../>
18             <exchange action="respond" ...>
19               <!-- ... -->
20             </exchange>
21           </interaction>
22         </sequence>
23       </sequence>
24     </sequence>
25   </choreography>
26 </package>
```

Listing 1. Choreography example.

example of this problem by using three nested `sequence` elements. In this example, their approach generates three nested BPEL `sequence` activities, although only the inner sequence (Lines 7–22) is relevant for the `Customer` and `Manufacturer` as they are referenced in the `fromRoleTypeRef` and `toRoleTypeRef`, respectively.

Therefore, we have adapted the rules from Mendling and Hafner (2007) and propose an extended endpoint projection mechanism based on a so-called *relevance mapping*. The basic idea is to map only those WS-CDL elements which are relevant in the BPEL process. To map the different ordering structures we need to distinguish between *child* and *descendant* relevance. The former describes that a relevant basic activity occurs as an immediate child of the respective ordering structure in the tree, whereas the latter describes that a relevant basic activity is nested at an arbitrary level. The relevance of a WS-CDL basic activity is determined by the occurrence of `interaction`, `assign` or `silentAction` where the `roleType` attribute is matching the `roleType` of the corresponding participant in the choreography.

If a node represents a relevant activity as described above, it is mapped to a BPEL activity according to Table 1, otherwise no mapping is generated. The basic algorithm for the relevance mapping is depicted in Listing 2. This algorithm takes a valid WS-CDL document as input and produces a number of BPEL documents, depending on the number of role types. For each role type at least one BPEL document is generated. We assume that a WS-CDL file is correct when it fulfils a number of properties, e.g. deadlock-free and livelock-free (Lohmann *et al*. 2007).

From Lines 2 to 7, we generate a BPEL process for each `roleType` found in the choreography. The algorithm inspects the `choreography` tag of the WS-CDL document by iterating each activity. If the activity type is an ordering structure or workunit a relevance mapping is performed (Lines 11–22). If the currently inspected activity is *descendant relevant* we must consider all child nodes of this activity (Line 12). If an activity is *child relevant* (Line 16), we generate the corresponding BPEL mapping according to Table 1 (Line 17). Otherwise, we recursively visit all child nodes (Line 19).

For our BPEL mapping we implemented an additional optimisation concerning the ordering structures. If a `parallel` or `sequence` ordering structure contains only one basic child activity, this ordering structure is ignored in the BPEL mapping (Lines 13–14). For instance, considering the example from Listing 1, only one BPEL `sequence` activity will be generated.

In Table 1 we have depicted a detailed overview of the WS-CDL to BPEL mapping rules. These rules are based on the mappings proposed by Mendling and Hafner (2007) and adapted where necessary. The adaption mainly includes `interaction` and `choice`. For the `interaction` activity and `choice` ordering structure, we also have to consider the role types to determine the sending and receiving party. Additionally, we address the synchronous and asynchronous message exchange patterns properly in the `interaction` activity.

Besides generating the BPEL artefacts, we also need to generate the WSDL artefacts from the WS-CDL description. The latter defines the static structure which can be extracted without analysing the choreography flow in detail. We generate a new WSDL document for each `roleType` of the choreography if the service interface is invoked somewhere in the choreography flow. The main idea is to check if the `roleType` is referenced within a `channelType` and a `variable` for this `channelType` exists that is used in an `interaction` with another partner. If this is the case, the `roleType` is in use and a WSDL needs to be generated. The WS-CDL to WSDL mapping is summarised in Table 2.

Table 1. WS-CDL to BPEL mapping.

WS-CDL	BPEL	Semantics
	Activities	
workunit (nested in choice)	case	Repeat and block attributes always false
workunit (block = true)	(receive)	Concept of blocking condition not defined in BPEL (Barros *et al*. 2005)
workunit (all other cases)	while	repeat = true and block = false
sequence	sequence	Sequential execution of activity units
parallel	flow	Parallel execution of activities
choice	switch	If inspected `roleType` is referenced in the guard condition of the inner workunit
	onMessage (nested in) pick	If inspected `roleType` is not referenced in the guard condition of the inner workunit but referenced in an `interaction` activity
interaction		
action = request	invoke	`fromRoleType` attribute corresponds to inspected role type
action = request	receive	`toRoleType` attribute corresponds to inspected role type. If `interaction` inside `workunit` which is defined inside a choice generate a BPEL `onMessage`
action = response	reply	`toRoleType` attribute corresponds to inspected role type
action = response	receive	`receive` only in the asynchronous case. For synchronous interaction append `outputVariable` to corresponding BPEL `invoke` which is defined in case 1
perform	*no mapping*	Perform separately defined choreography
assign	assign (for party in role type)	Variable assignment
silentAction	sequence with nested empty	To be refined in the BPEL process
noAction	empty (for party in role type)	Do nothing
finalise	compensationHandler	Finalising activities after completion

For a detailed description of the WS-CDL to WSDL mapping algorithm, refer to Rosenberg *et al*. (2007).

3.3. SLA/QoS integration

The integration of QoS attributes in business process development raises the need for appropriate techniques to consider QoS at the choreography and orchestration layer. Considering QoS at the choreography layer can be achieved by using SLAs which focus on performance and dependability aspects of the underlying QoS model. In contrast, the integration of QoS at the orchestration layer can be attained by the use of Web service policies. This section describes how WS-CDL and BPEL can be extended to support QoS attributes.

```
1  void transform(WscdlDoc doc) {
2    List<RoleType> roles = doc.getRoleTypes();
3    for (RoleType role : roles) {
4      for (activity : n.getActivites()) {
5        if (isOrderingStructure(activity) || isWorkUnit(activity))
6          relevanceMapping(activity, role);
7      }
8    }
9  }
10
11 void relevanceMapping(Node n, RoleType role) {
12   if (isDescendanceRelevant(n)) {
13     if (n.getRelevantChildCount() > 1)
14       createBPELMapping(n);
15     for (child : n.getChildNodes()) {
16       if (isChildRelevant(child))
17         createBPELMapping(child);
18       else
19         relevanceMapping(child, role);
20     }
21   }
22 }
```

Listing 2. Relevance mapping.

Table 2. WS-CDL to WSDL mapping.

WSDL		WS-CDL	
Element	*Attribute*	*Element*	*Attribute*
definitions	xmlns:tns targetNS name	package behaviour	xmlns:tns targetNS name
message	name	exchange	informationType

portType
  operation
  [input | | output]
	name
	name
	name
	message
		behaviour
		interaction
		exchange
			interface
			operation
			action
			informationType

binding
 	
  operation
  soap:operation
 	
  input soap:body
  output soap:body
	name
	type
	name
	soapAction
	
	namespace
	namespace
		behaviour
		
		interaction
		behaviour
		interaction
		behaviour
		behaviour
			name + 'Binding'
			'tns:' + interface+'Binding'
			operation
			interface namespace + operation
			
			interface namespace
			interface namespace

service port	name name binding	behaviour behaviour	interface + 'Service' interface + 'Port' 'tns:' + name + 'Binding'

3.3.1. *SLA integration*

As mentioned above, we use SLAs to integrate QoS at the choreography layer. For the definition of the SLAs we decided to use WSLA (Ludwig *et al.* 2008). For the actual integration, we extended WS-CDL with a construct which holds SLA references. We therefore leverage semantic annotations in WS-CDL constructs using the `description` element as shown in Listing 3 (Lines 3–7).

3.3.2. WS-QoS policy

In order to bring QoS aspects from the choreography to the orchestration layer, SLAs have to be mapped to the corresponding Web service policies to allow monitoring and enforcement of QoS attributes at the execution layer. However, the current WS-Policy specification focuses on security (WS-SecurityPolicy) and reliable messaging (WS-RMPolicy), whereas performance and dependability are not addressed. The WS-Policy framework provides a grammar for the definition of domain-specific policies. Hence, we use this feature to extend the WS-Policy framework by defining a WS-QoSPolicy. In our approach, the mapping from SLA to WS-QoSPolicy and its attachment to BPEL is done automatically, whereas other policy types (such as security and reliable messaging) are currently not supported by the mapping. Nevertheless, these can be manually attached to the BPEL documents as described in Charfi *et al*. (2007).

Before we go into the details of our mapping between SLAs and policies, we briefly sketch the underlying QoS model. Rosenberg *et al*. (2006) have defined a QoS model for Web services by identifying different QoS attributes. Since some attributes are either dependent on external factors or derived from empirical values, not all attributes are determinable in advance. In this work, we consider *Processing Time, Execution Time, Throughput* and *Availability* as possible QoS attributes in the WS-QoSPolicy. However, guarantees on the *Execution Time* will usually be defined in SLAs instead of *Processing Time* because the first also includes the network latency, which partly depends on the client's network connection.

The WS-QoSPolicy defines assertions for these QoS attributes. The normative outline of the assertions is shown in Listing 4. It defines `type`, `unit`, `predicate` and `value` of the assertion.

A concrete example for two such policy assertions is illustrated in Listing 5. Lines 3–4 define an assertion on the execution time, which has to be less than five seconds. Lines 5–6 state that the throughput has to be more than 100 requests per second.

3.3.3. SLA/QoS mapping

Our extension of the WSLA schema restricts the SLA parameters to the pre-defined QoS attributes introduced in the previous section. Therefore, the SLA can be directly mapped

```
1 <roleType name="ManRoleType">
2   <behaviour interface="b2o:manInterface" name="ManBehaviour"/>
3     <description type="semantics">
4       <qosp:slaReference name="SLA1" serviceconsumer="CustRoleType"
5         uri="http:// www.vitalab.tuwien.ac.at/slas/ManufacturerCustomerSLA.xml"
6       </qosp:slaReference>
7     </description>
8   </behaviour>
9 </roleType>
```

Listing 3. SLA integration in WS-CDL.

```
1 <qosp:[QoS]Assertion unit="xs:string"
2       predicate="tns:PredicateType"
3       value="xs:integer | xs:flow"/>
```

Listing 4. WS-QoSPolicy assertions.

to the WS-QoSPolicy, which consists of the following two steps: first, each SLA is mapped to a policy and, secondly, each SLA parameter is mapped to a policy assertion.

As each SLA may consist of one or more SLOs, we identified three patterns:

(1) One SLO is defined for each SLA parameter.
(2) One SLO consists of multiple SLA parameters.
(3) SLA parameters are defined in multiple SLOs.

Each of these patterns can be successfully mapped to an equivalent policy. In the first case, one `All` operator is used to contain all policy assertions. For each SLO, exactly one policy assertion will be generated. For example, an SLO `SLOServiceExecutionTime` defines an SLA parameter which corresponds to the type `ExecutionTime`. This parameter will be mapped to the corresponding policy assertion according to the WS-QoSPolicy.

In the second case, SLA parameters are grouped by an SLO using the logical operators `And, Or, Not, Implies`. Table 3 shows how these constructs can be mapped to equivalent WS-QoSPolicy operators. For example, such groupings of SLA parameters can be used to define an SLO `SLOServicePerformance` by combining `Throughput` and `ExecutionTime` in various ways using the provided logical operators.

In the third case, for each SLO a time period has to be specified. Therefore, it is possible to define multiple SLOs for different time periods. For instance, during peak-hours the execution time of a service has to be less than a specific value.

3.3.4. *WS-QoS policy integration*

The definition of a QoS policy and QoS/SLA mapping rules are the fundamental concepts for considering QoS in Web service based business process development. Yet the question remains of how to integrate the generated QoS policies in the orchestration layer. Regarding the top-down modelling approach of Web services, two integration approaches can be differentiated: policies can either be attached to service descriptions (WSDL) or be integrated in BPEL processes.

```
1 <wsp:Policy>
2   <wsp:All>
3     <qosp:ExecutionTimeAssertion unit="seconds"
4                                  predicate="Less" value="5"/>
5     <qosp:ThroughputAssertion unit="requests"
6                               predicate="GreaterEqual" value="100"/>
7   </wsp:All>
8 </wsp:Policy>
```

Listing 5. Assertion example.

Table 3. SLA operator mapping.

SLA operator		WS-QoSPolicy operator
`And`	→	`All`
`Or`	→	`ExactlyOne`
`Not`	→	Reverse predicate
`Implies`	→	`ExactlyOne` and reverse predicate

Attaching policies to WSDL descriptions following the WS-PolicyAttachment (Vedamuthu *et al.* 2008b) specification has two main drawbacks. First, service invocations are always subject to a policy, even if the service consumer has no corresponding SLA. Secondly, the service provider cannot differ between multiple policies for the same service since policies do not contain information about the participating parties. Therefore, following the second approach, the policies should be integrated in BPEL processes.

Extensibility in BPEL is achieved by allowing elements from other namespaces. The BPEL `partnerLink` element is the place to integrate the policy. For this integration, both synchronous (request-reply) and asynchronous (callback) message exchange patterns have to be considered. In contrast to the asynchronous case, in the synchronous case the service provider has no additional information about the service consumer, because the `partnerLink` has no service consumer-specific details. Therefore, the policy has to be integrated at the service consumer side as illustrated in Listing 6 (Lines 5–8).


```
1  <process>
2    <partnerLinks>
3      <partnerLink name="POService"
4                   partnerLinkType="ns1:POServiceLT" partnerRole="POServiceRole">
5        <wsp:Policy xmlns:qosp="..." xmlns:wsu="..."
6                    wsu:Id="xs:QName" qosp:operation="...">
7          <!-- ... -->
8        </wsp:Policy>
9      </partnerLink>
10     <!-- ... -->
11   </partnerLinks>
12   <!-- ... -->
13 </process>
```

Listing 6. Policy integration in BPEL.

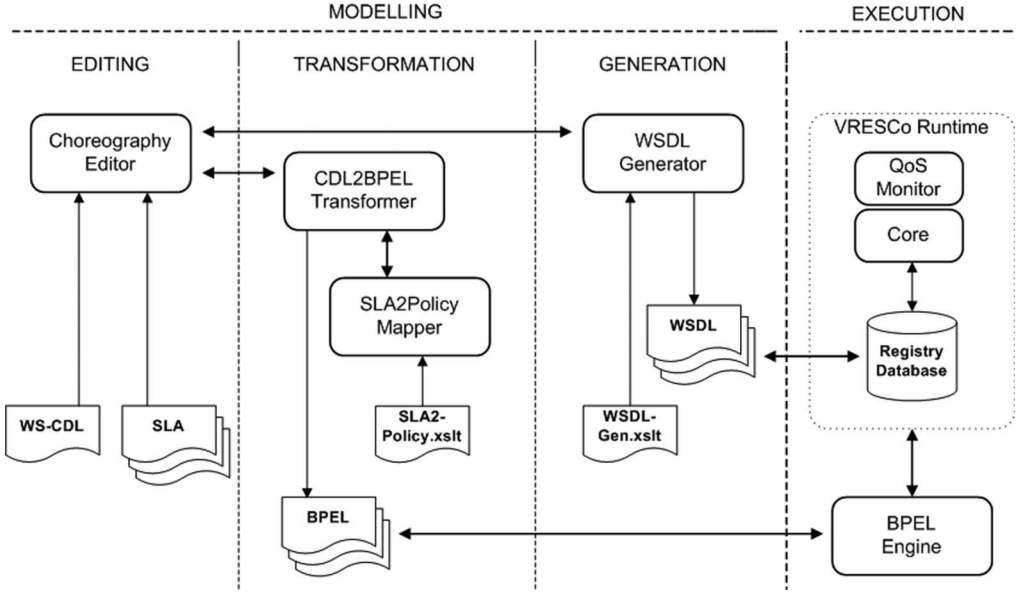


Figure 3. System architecture.

4. Architecture and implementation

The concepts and algorithms described in Section 3 have been implemented in Java using a simple Swing-based graphical user interface. The architecture of this system consists of four parts, which are depicted in Figure 3. We distinguish between the modelling and the execution phases. The modelling phase consists of editing choreographies and SLAs, transforming WS-CDL into BPEL and SLAs into policy assertions, and finally generating the WSDL artefacts. After implementing the private business logic in the BPEL artefacts and the corresponding services, they can be deployed in the VRESCo runtime as part of the execution phase.

4.1. Modelling phase

Editing choreographies in our approach is done in two steps: first, the choreography is modelled using the Pi4soa Eclipse plugin (pi4 Technologies Foundation 2008). This plugin provides a graphical tooling for creating complex choreographies. Secondly, our simple Swing-based SLA annotation tool is used to add SLA references to specific role types. Furthermore, the editor initiates the generation of WSDL and BPEL artefacts as described above.

The *Transformation* component implements the algorithms for transforming WS-CDL to BPEL, and SLA to policies. The WS-CDL to BPEL transformation is implemented using the DOM4J API whereas the SLA transformation is implemented using XSLT. During the transformation step one BPEL document is generated for each partner including the policy references which conform to the SLAs in the choreography layer.

The *Generation* component is responsible for generating the WSDL files from a choreography according to the algorithm described in Section 3. This component is also implemented using XSLT stylesheets.

4.2. Execution phase

The VRESCo runtime introduced in Michlmayr *et al*. (2007) aims at addressing some of the current challenges in service-oriented computing research and practice (Papazoglou *et al*. 2007). Among others, this includes topics related to service discovery and metadata, dynamic binding and invocation, service monitoring, QoS-aware service composition, service management and versioning (Leitner *et al*. 2008), and service notifications (Michlmayr *et al*. 2008). Besides this, another goal of the VRESCo project is to facilitate the engineering of service-oriented applications by reconciling some of these topics and abstracting from protocol related issues.

The VRESCo core services illustrated on the right-hand side of Figure 3 are responsible for publishing and querying services and associated metadata in the registry database. After generating the services in the modelling phase, the VRESCo publishing service is used to make these services available to the runtime.

The QoS monitor introduced in Rosenberg *et al*. (2006) is responsible for measuring the performance and dependability attributes of Web services. This monitor is integrated as part of the VRESCo runtime to measure the QoS attributes of the services published in the registry database. These QoS attributes are then used to monitor compliance with the SLA defined for the choreographies.

The BPEL processes generated in the modelling phase are finally executed using the Windows Workflow Foundation (WWF) that runs on top of the Microsoft .NET platform

and provides support for executing BPEL compliant processes. WWF is integrated into VRESCo as part of its composition support.

5. Validation and discussion

In this section, we validate our work using the case study described in Section 2 to demonstrate the feasibility of our approach. Additionally, we discuss the lessons learned during the validation and describe the limitations of our approach.

5.1. Validation

For modelling the choreography we have used the Pi4soa Eclipse Plugin. During the modelling phase, the main part is to identify the partners in the process and the messages that are exchanged among the partners. Most parts of the BTO scenario are implemented in the choreography itself. However, some non-observable implementation-specific details cannot be considered from a choreography point of view but have to be implemented internally by the choreography participants.

The choreography itself is then used to generate the BPEL and WSDL templates for each partner in the choreography. The SLAs are modelled pairwise and independently among the partners. The partners agree on a set of runtime constraints that need to hold during the message interactions. In general, the SLAs are independent of the choreography itself; nevertheless, the integration of an SLA in the development process can be achieved by adding an SLA reference to a specific `roleType` in the WS-CDL (compare Listing 3 for an integration example).

In our case study we have identified four different SLAs: one between the customer and the manufacturer, and one for every manufacturer–supplier pair. For example, an SLA between the manufacturer and the CPU supplier specifies the expected response time, throughput and execution time of a service including the periods where these obligations are valid. The transformation of SLAs to policy assertions does not generate new files. Instead, after the transformation of the WS-CDL to BPEL processes, the generated BPEL process contains a `partnerLink` annotated with WS-Policy statements to express the SLOs as enforceable policies, as can be seen in Listing 4 for execution time and throughput.

After the transformation steps, the generated BPEL and WSDL files are taken as a starting point for implementing the private business logic. This mainly deals with aspects that cannot be modelled from a global viewpoint in the choreography. These internal implementations are referred to as silent actions implemented during refinement of the BPEL code.

Finally, the services and BPEL processes are deployed to an orchestration engine such as ActiveBPEL that is responsible for executing the BPEL processes. Besides this, the services are published into the VRESCo runtime, which has a number of advantages. For instance, all services and associated metadata are published within the registry database and can be queried using the VRESCo core service. Furthermore, the QoS monitor integrated into the VRESCo runtime continuously measures the QoS attributes of the services within the runtime to monitor compliance with the SLAs defined in the choreography.

5.2. Discussion and limitations

An important point of discussion is the use of WS-CDL. Some may scrutinise why we use choreographies instead of following a bottom-up approach that builds on orchestration

languages such as BPEL. In fact, both modelling approaches are feasible and have their strengths and weaknesses. However, BPEL is intended for modelling business processes without knowledge of the global viewpoint. In contrast to this, we decided to stay close to the vision of cross-organisational choreography descriptions by using WS-CDL.

During the implementation of our case study we encountered several aspects which have to be considered when using such a top-down modelling approach. Some of these issues seem inherent to the domain of model-driven development in general. On the one hand, our approach is based on choreographies representing a global viewpoint of the business processes which raises the need for precise modelling of the global behaviour. To be more concrete, the business partners have to agree precisely on the message format used for their interaction. On the other hand, after the choreography has been initially defined, the underlying business model may evolve and lead to significant changes. Such changes clearly affect the partner processes, which causes the generation of new BPEL processes and corresponding WSDL files.

A similar problem of the top-down modelling approach is the compliance with existing choreographies. In such cases, an automated generation of the BPEL processes for each partner is not reasonable, as parts of the overall system already exist. Nevertheless, our approach can be used to generate the local BPEL code and WSDL files from an existing choreography. The QoS-specific policies then have to be attached manually because no global SLA annotations are available.

Enforcing SLA is an important aspect when considering cross-organisational business processes. So far we have focused on the modelling and integration of SLAs and QoS attributes in choreographies and orchestrations. As part of the VRESCo project, we are currently working on SLA enforcement. The goal is to monitor the QoS attributes from the orchestration layer, as described in Section 4, and trigger actions if these attributes do not comply with values defined in the SLA. Such actions may range from notifying the service providers to assigning penalties according to some penalty model.

6. Related work

Integrating QoS in Web service based business process development has not yet received much attention, whereas the modelling of choreographies is the subject of various research activities (e.g. Decker *et al*. 2007, Zaha *et al*. 2006). In this section, we discuss existing choreography modelling and transformation approaches and the integration of policies in BPEL.

Mendling and Hafner (2007) define mapping rules for the derivation of BPEL processes from a WS-CDL choreography description. For each WS-CDL ordering structure and activity the corresponding BPEL construct respective activity is determined. These mapping rules define the basis for the mapping rules used throughout the top-down modelling process in this work. Whereas the mapping of WS-CDL to BPEL is referenced in detail, the generation of WSDL interfaces used in the BPEL processes is not addressed explicitly. In contrast to our work, the authors do not define explicit endpoint projection rules to determine which ordering structures are relevant for the participants of the choreography description. Additionally, we define mapping rules for the generation of WSDL descriptions which correspond to the service interface descriptions of the derived BPEL processes.

Diaz *et al*. (2006) use an intermediary model for the generation of BPEL processes from a WS-CDL choreography description concentrating on Web services where time constraints play a critical role. A choreography description is first transformed into a

timed automata model which is verified and validated for correctness using formal model checking techniques. This model is then further used to generate BPEL processes. In contrast to our work the focus is laid on the generation and verification of the timed automata model. Detailed mapping rules for the derivation of BPEL processes out of this model are not specified. We argue that in the context of top-down modelling it seems more appropriate to perform a direct mapping between WS-CDL and BPEL instead of using an intermediary model.

Decker *et al.* (2007) propose an new extension to BPEL called BPEL4Chor that allows modelling of choreographies within BPEL by leveraging an interconnected interface behaviour model, whereas WS-CDL represents an interaction model. As stated in their work, it has not been investigated yet which of these two approaches is more appropriate for human modellers. While we follow a top-down approach by transforming WS-CDL into BPEL, the authors propose a bottom-up approach by introducing a new choreography layer on top of BPEL. However, in contrast to our work, the integration of QoS into Web service choreographies is not addressed.

Pi4soa (pi4 Technologies Foundation 2008) is a toolset from π4 Technologies representing one of the first WS-CDL implementations. It provides a designer tool as an Eclipse plugin, which we used for modelling our choreographies, and a possibility to generate Java services from a WS-CDL. The support for generating BPEL processes is currently in progress. In contrast to pi4soa, our work considers QoS from the beginning of the development. However, it might be interesting to include the SLA/QoS related aspects into the pi4soa Eclipse plugin.

7. Conclusions

There have been some considerable debates as to the relationship between choreography and orchestration. Some people argue that there is no need for choreography and all business interactions can, and in fact should, be modelled in BPEL. Others advocate the use of modelling by using WS-CDL but then lament the lack of execution abilities. The prime motivation for the contribution of this paper is today's lack of modelling support for QoS-aware business processes. In particular, the need for QoS-aware processes is apparent in inter-organisational business processes.

The novelty of our approach lies within the fact that we consider SLAs as first-class entities while modelling service choreographies. Our approach allows for automatic generation of executable BPEL orchestrations and WSDL files for each partner in the choreography. A novel contribution is the mapping of QoS information specified in SLAs to WS-QoS policies which are attached to the BPEL process. As a consequence, a policy-aware middleware can verify, and possibly enforce, SLAs. The approach has been implemented and the feasibility is demonstrated using a simplified version of a built-to-order case study.

As future work, we plan to apply this approach to larger case studies in order to estimate the scalability of our approach. Moreover, we are currently adding SLA enforcement capabilities to the VRESCo runtime. Finally, we envision the implementation of our tool support within the Pi4soa Eclipse plugin to allow a better integration with existing modelling tools.

Acknowledgements

The research leading to these results has received funding from the European Community's Seventh Framework Programme [FP7/2007-2013] under Grant Agreement 215483 (S-Cube).

We also want to thank Christian Enzi for a first prototype implementation of these mappings and the tool support, and the anonymous reviewers for their helpful comments to improve the paper.

Notes

1. In this paper we use the term Web service and service interchangeably.
2. The case study can be downloaded from http://www.vitalab.tuwien.ac.at/~florian/qosintegrator/.

References

Alves, A., *et al*., 2007. *Web service business process execution language 2.0*. Billerica, MA, USA: OASIS. Available from: http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=wsbpel [Accessed 1 October 2008].

Barros, A., Dumas, M., and Oaks, P., 2005. A critical overview of the web services choreography description language (WS-CDL). *BPTrends Newsletter*, 3 (3).

Carbone, M., *et al*., 2007. Structured communication-centred programming for web serices. *In*: R. De Nicola, ed. *Proceedings of the 16th european symposium on programming (ESOP'07)*, 24 March–1 April 2007, Braga, Portugal. Berlin, Germany: Springer Verlag, 2–17.

Charfi, A., Khalaf, R., and Mukhi, N., 2007. QoS-aware web service compositions using non-intrusive policy attachment to BPEL. *In*: B.J. Krämer, K.J. Lin, and P. Narasimhan, eds. *Proceedings of the 5th international conference on service-oriented computing (ICSOC'07)*, 17–20 September 2007, Vienna, Austria. Berlin, Germany: Springer Verlag, 582–593.

Decker, G., *et al*., 2007. BPEL4chor: extending BPEL for modeling choreographies. *In*: L.J. Zhang, K.P. Birman, and J. Zhang, eds. *Proceedings of the IEEE international conference on web services (ICWS'07)*, 9–13 July 2007, Salt Lake City, UT, USA. Los Alamitos, CA, USA: IEEE Computer Society, 296–303.

Diaz, G., *et al*., 2006. Automatic generation of correct web services choreographies and orchestrations with model checking techniques. *In*: IEEE, ed. *Proceedings of the international conference on internet and web applications and services (ICIW'06)*, 19–25 February 2006, Guadeloupe, French Caribbean. Los Alamitos, CA, USA: IEEE Computer Society.

Kavantzas, N., *et al*., 2005. *Web services choreography description language (WS-CDL)*. Cambridge, MA, USA: W3C. Available from: http://www.w3.org/TR/ws-cdl-10/ [Accessed 1 October 2008].

Leitner, P., *et al*., 2008. End-to-end versioning support for web services. *In*: IEEE, ed. *Proceedings of the international conference on services computing (SCC 2008)*, 8–11 July 2008, Honolulu, HI, USA. Los Alamitos, CA, USA: IEEE Computer Society.

Lohmann, N., *et al*., 2007. Analyzing BPEL4Chor: verification and participant synthesis. *In*: M. Dumas and R. Heckel, eds. *Proceedings of the 4th international workshop on web services and formal methods (WS-FM'07)*, 28–29 September 2007, Brisbane, Australia. Berlin, Germany: Springer Verlag, 46–60.

Ludwig, H., *et al*., 2003. *Web service level agreement (WSLA) language specification*. Hawthorne, NY, USA: IBM Corporation. Available from: http://www.research.ibm.com/wsla/ [Accessed 1 October 2008].

Mendling, J. and Hafner, M., 2008. From WS-CDL choreography to BPEL process orchestration. *Journal of Enterprise Information Management*, 21 (5), 525–542.

Michlmayr, A., *et al*., 2008. Advanced event processing and notifications in service runtime environments. *In*: A.P. Buchmann and S.T. Piergiovanni, eds. *Proceedings of the 2nd international conference on distributed event-based systems (DEBS'08)*, 1–4 July 2008, Rome, Italy. New York, NY, USA: ACM Press.

Michlmayr, A., *et al*., 2007. Towards recovering the broken SOA triangle – a software engineering perspective. *In*: E.D. Nitto, A. Polini, and A. Zisman, eds. *Proceedings of the second international workshop on service oriented software engineering (IW-SOSWE'07)*, 3 September 2007, Dubrovnik, Croatia. New York, NY, USA: ACM Press, 22–28.

Papazoglou, M.P., *et al*., 2007. Service-oriented computing: state of the art and research challenges. *IEEE Computer*, 40 (11), 38–45.

Peltz, C., 2003. Web services orchestration and choreography. *IEEE Computer*, 36 (10), 46–52.

pi4 Technologies Foundation, 2006. *pi4soa*. Mountain View, CA, USA: SourceForge. Available from: http://sourceforge.net/projects/pi4soa [Accessed 25 January 2008].

Rosenberg, F., *et al*., 2007. Integrating quality of service aspects in top-down business process development using WS-CDL and WS-BPEL. *In*: IEEE, ed. *Proceedings of the 11th enterprise computing conference (EDOC'07)*, 15–19 October 2007, Annapolis, MD, USA. Los Alamitos, CA, USA: IEEE Computer Society, 15–26.

Rosenberg, F., Platzer, C., and Dustdar, S., 2006. Bootstrapping performance and dependability attributes of web services. *In*: IEEE, ed. *Proceedings of the IEEE international conference on web services (ICWS'06)*, 18–22 September 2006, Chicago, IL, USA. Los Alamitos, CA, USA: IEEE Computer Society.

Vedamuthu, A.S., *et al*., 2007a. *Web services policy 1.5 – attachment*. Cambridge, MA, USA: W3C. Available from: http://www.w3.org/TR/ws-policy-attach/ [Accessed 25 January 2008].

Vedamuthu, A.S., *et al*., 2007b. *Web services policy 1.5 – framework*. Cambridge, MA, USA: W3C. Available from: http://www.w3.org/TR/ws-policy/ [Accessed 25 January 2008].

Völter, M. and Stahl, T., 2006. *Model-driven software development: technology, engineering, management*. Hoboken, NJ, USA: John Wiley & Sons.

Zaha, J.M., *et al*., 2006. Let's dance: a language for service behavior modeling. *In*: R. Meersman and Z. Tari, eds. *Proceedings of the 14th international conference on cooperative information systems (CoopIS'06)*, 29 October–3 November 2006, Montpellier, France. Berlin, Germany: Springer Verlag.

CD – JRA – 2.2.2

Paper [3]:

Gehlert A., Hielscher J., Danylevych O., Karastoyanova D.:

Online Testing, Requirements Engineering and Service Faults as Drivers for Adapting Service Compositions

In: *Proceedings of the International Workshop on Service Monitoring, Adaptation and Beyond (MONA+ 2008), December 13, 2008, Madrid, Spain, Pages 39--50, 2008.*

Online Testing, Requirements Engineering and Service Faults as Drivers for Adapting Service Compositions

Andreas Gehlert[1], Julia Hielscher[1], Olha Danylevych[2], Dimka Karastoyanova[2]

[1]University of Duisburg-Essen, Schuetzenbahn 70, 45117 Essen, Germany
{andreas.gehlert | julia.hielscher}@sse-uni-due.de
[2] IAAS, University of Stuttgart, Universitaetsstr. 38, 70569 Stuttgart, Germany
{olha.danylevych | dimka.karastoyanova}@iaas.uni-stuttgart.de

Abstract. Adaptability is a key feature of service-based applications (SBAs). Multiple approaches for adaptability, including those borrowed from the traditional workflow technology, can be used to react to various types of changes in the SBA's environment. Unlike previous fragmented research, we aim at presenting a unified view reflecting the convergence of approaches from requirements engineering, online testing and adaptation mechanisms for service compositions. The main result of our approach is that a dynamic binding strategy known from service composition research leads to an interaction of the requirements engineering and online testing activities with an enterprise service registry only and, therefore, to a loose coupling between the three activities.

Keywords: Service Composition, Adaptability, Requirements Engineering, Online Testing, Self-optimization, Web Services

1 Introduction

The software development life-cycle for service-based applications (SBAs) usually has a very short design and testing phase [1, p. 46]. This modification of traditional software life cycles is due to two facts: First, the increased importance of software systems for the success of a company implies that software systems are developed more rapidly to shorten the time to market and, second, companies rely on the inherent flexibility of SBAs that facilitates their runtime adaptation according to the current service provision, changes in (legal) regulations, systems support, and according to existing and/or new requirements. This flexibility allows reducing the time needed for analysing, designing, deploying and testing SBAs since testing as well as eliciting new requirements can be postponed to the runtime phase of the SBA.

In this paper we show how the adaptability feature of SBAs changes the way in which SBAs are developed and tested. Unlike previous approaches, which focus on technical mechanisms needed to enable adaptation, we focus on the integration of three different research streams, namely the adaptability of service compositions,

online testing and requirements engineering. We aim at showing how techniques from these three areas may be intertwined.

Service-based applications are typically implementing a business process. The accepted approach for implementing business processes in a service-based environment is the workflow-based approach. Workflows using services are also called service compositions and are described in terms of control flow (tasks and their ordering), data flow (data exchanges between tasks and with services), exception handling and the service implementations that realise the individual tasks. Therefore, service composition descriptions are organized in two dimensions – control logic and functionality. For simplicity we assume that an SBA is a service composition and, hence, use both terms synonymously in this paper.

The mechanisms enabling adaptability of service compositions with respect to their functions dimension are currently restricted to anticipating service failures. Consequently, these mechanisms are not designed to react to online testing results and to newly available services, which are discovered by the requirements engineer in a continuous requirements engineering process.

This paper aims to bridge this gap and to extend current adaptation mechanism of service compositions to three adaptation drivers: service failure (traditional adaptation technique), online testing results achieved during the runtime of the SBAs and new available services discovered in a continuous requirements engineering process.

The paper is structured as follows: In section 2 we present an overview of the approach to SBA adaptability that unifies the view of three fields of research, namely requirements engineering, online testing and adaptation of service compositions. In section 3 we identify the drivers for adaptability of SBAs, which are closely linked to our adaptation approach. This integration is then demonstrated with the help of an example in section 4 and the conclusions are summarised in section 5.

2 Integrating RE, Online Testing and Service Composition Adaptation Techniques

The aim of this section is twofold: First, we describe the rationale of our approach by introducing motivating scenarios and, second, we describe the adaptation of service compositions, requirements engineering and online testing in more detail. The approach is outlined in Figure 1.

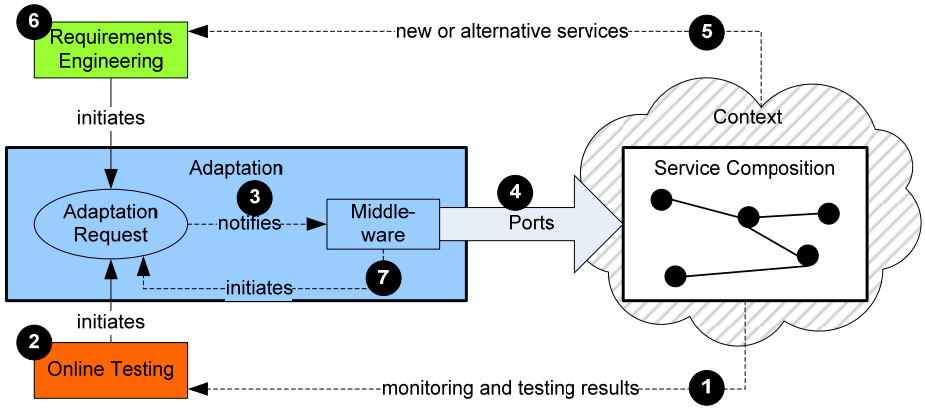


Figure 1: Integration of Online Testing, Requirements Engineering and Service Composition Adaptation Techniques

To illustrate our approach, we use the wine production example introduced in [2, p. 330] (cf. Figure 2 for the wine production process). The wine delivery process is described by six activities. The process starts with buying wine grapes (task "Buy Grapes"). After this task, the production (task "Produce wine") is carried out. Afterwards the process might proceed in two different ways. The first way is storing the wine in oak barrels to mature (task "Store"), which may be followed by the shipment task ("ship wine") or by finishing the production process (task "Wine lot production finished"). The alternative is to ship the wine immediately (task "Ship wine") followed by receiving a delivery note (task "Receive delivery note") and finished by the task "Wine lot production finished".

Assume now the following three scenarios which need three different adaptability drivers:

1. *Scenario 1 - Requirements Engineering*: Assume that the requirements engineer found a new shipping service, which is cheaper than the previously used service in the service composition (❺). If the requirements engineer decides to use this service in the service composition s/he needs to send an adaptation request (❻) to the process engine, which will eventually notify the service middleware (❸). This in turn leads to the usage of the new service in the service composition (❹).
2. *Scenario 2 - Online Testing Scenario*: Assume that online testing finds a failure in the "Buy grapes" service, which would lead to buying red instead of white grapes under some circumstances (❶) before invoking the service for a particular SBA instance. In this case the online testing activity (❷) initiates an adaptation request. This adaptation request leads to a notification of the middleware (❸), which needs to find a new service and to use this service in the service composition (❹).
3. *Scenario 3 - Service Failure Scenario*: Assume that the middleware wants to invoke the "shipment" service, which is currently not available. As a consequence, the middleware sends and adaptation request, which is either handled

by the middleware itself (❹) or by the service compositions (❼). In any case the service composition is adapted to use a new service (❹).

To understand the interplay between requirements engineering, online testing and adaptation, these techniques are described in the following subsections in more detail.

2.1 Requirements Engineering

Requirements engineering (RE) in traditional software engineering process models is carried out as separate activity before the design of the software system. It aims to elicit, document and agree upon the goals, assumptions and requirements of the software system to be [3]. Although RE activities are also an essential part of the engineering process for SBAs, which determine the goals and purposes of the SBA, we argue that requirements engineering should be a continuous process, which covers the entire life cycle of the SBA. The rationale of this argument is the adaptability of the SBA, which allows the requirements engineer to trigger an adaptation of the SBA, e. g. in case of newly available services (scenario 1).

If a new service becomes available the requirements engineer needs to check for two conditions: First, the new service must provide the functionality, which is actually needed by the SBA, i. e. the service must "fit" with the purpose of the SBA. Second, the service should fulfil the requirements better than the previously used service. The second requirement ensures that the new service is superior to existing services. This continuous requirements engineering approach rests on the assumption that new services become available over time.

The algorithms needed for this continuous requirements engineering process are described in [4, 5]. The main idea of these approaches can be described as follows: The requirements of a SBA are described as Tropos goal models [6, 7]. The rationale for using Tropos is threefold: First, Tropos is a comprehensive approach to develop software systems from early requirements to their implementation. Second, Tropos was already applied to the service discipline, e. g., it was already shown that it is applicable to SBAs [e. g. 8, 9-12]. Third, Tropos comes with a formalisation which allows analysing the influence of the satisfaction of one goal on the entire goal model [13].

Based on the assumption that services are described with goal models, the requirements engineer has to carry out the following two steps once a new service becomes available: First, the goal model of the new service should be contained in the goal model of the SBA. This step ensures that the new service provides a functionality needed by the SBA. Second, once this matchmaking is done, the goal satisfaction for all hard- and soft-goals can be calculated based on TROPOS's goal propagation algorithm. The result of this calculation is threefold: First, the goal satisfaction rates of some goals are higher while the goal satisfaction rates for the remaining goals remain unchanged. In this case the new service is superior to the existing one and it should be used (pareto principle). Second the goal satisfaction rates of some goals are lower then before and remain unchanged for all other goals. In this case the service is inferior to the previous one and it should not be used. Third, the goal satisfaction rates are higher for some goals and lower for some other goals. In this case the requirements

engineer needs to decide whether to use the new service based on the priority of the different goals.

This goal driven approach to SBA leads to perfective maintenance known from software engineering [14, p. 493] or to so called self optimization in the service world [1, p. 43]. The interaction of this approach with the adaptation of the service composition is described in section 3.

2.2 Online Testing

In traditional software development processes quality assurance including testing is almost finished when the runtime of a system starts. The loose coupling feature of SBAs, however, allows to easily adapting the SBA during runtime so that the testing phase and the runtime phase of an SBA may overlap. This means that some testing activities are purposely postponed until the SBA is up and running [15, p. 40]. The consequence of this overlap is that the SBA may be modified once an error is detected during the runtime quality assurance – in particular online testing – activities.

Similar to traditional software systems, two different inputs can be considered to assure the quality of SBAs: monitoring information and testing outputs. In contrast to traditional software development the comparison of expected and actual behaviour and possible following adaptations has to be done at runtime. Consequently, not only monitoring techniques but also online testing techniques are relevant to assure the quality of the entire SBA. However, due to space limitation, we only concentrate on online testing in this paper.

Hielscher et al. present the *PROSA* (*PRO*-active *S*elf-*A*daptation) framework in [16], which enables online testing in addition to traditional runtime quality assurance methods such as monitoring. The results of the online testing activities are compared with expected results to detect problems, which may be resolved by adaptations. Online testing aims at finding possible problems before they occur in a running instance of the SBA. The underlying assumption of PROSA is that online testing activities do not interfere with the running SBA. This requires that each service, which should be tested, offers a test mode. This test mode prevents the “normal” execution of all activities of an instance of the SBA, which trigger more than just returning a computational result – e. g. it prevents the delivery of physical goods.

Testing service compositions by testing their constituent services requires that the services are known at test time. This means that either the services are statically connected to the tasks of the service composition (see below) or that the registry from which the services are discovered contains a constant set of services during test time. If none of the two previous conditions holds, it follows that online testing must be executed in parallel to each service composition instance since the services used in different service composition instances may differ. This approach, however, requires considerable computational resources and is, therefore, discarded here.

2.3 Adaptation

Adaptations in service compositions can be carried out during design time or during runtime. During design time the process model, which describes control and data flow can be modified in any possible way, including the assignment of service implementa-

tions to tasks or the modification of the control and data flow of the workflow definition ([17]).

During runtime the process model may be altered so as to assign new service implementations to process tasks [e. g. 18] or to modify the control and data flow completely. After these modifications it must be decided whether all running instances should benefit from these modifications (instance migration), whether some of the running instances should benefit from these modifications or whether future instances should use the modifications only. If the modifications target only some instances of a process model, the changes are called ad-hoc changes and are not reflected in the process model.

According to the previous classification of adaptation, we need to distinguish two modification dimensions – e. g. modification of the process model and modification of the assignment of concrete services to process model tasks – and two phases in the SBA lifecycle – e. g. design time and runtime. In this paper, however, we only focus on modifying the assignment of service implementations to process tasks during runtime.

This assignment is based on so called *binding strategies*. According to [20] and [19, p. 536] we can distinguish between two major binding strategies: First, the *static binding strategy* requires that the service implementations are assigned to the activities during design or deployment time. Second, *dynamic binding* requires a declarative description of the requirements of the service at design or deployment time. The actual service implementations are then discovered by the underlying middleware during runtime based on the declaratively specified criteria.

The dynamic binding strategy is the most flexible approach for service binding because it enables the discovery of service implementations during runtime based on requirements for the service selection specified declaratively prior to the process execution. In case of a service implementation failure, the middleware is then able to discover a new service implementation according to the specified requirements. The functional part of the requirements is typically provided by the service composition (e. g. operation and port type) and the non-functional requirements are specified in separate artefacts associated with the composition (e. g. WS-Policy definitions). It is important to note that these requirements for service selection are specified at the latest during the deployment phase of SBA development. This means, that the selection *criteria* even in the most flexible strategy for service binding are fixed during runtime.

3 Adaptability Drivers

Service-based applications that are implemented using the process-based approach can be adapted as a reaction to changes in the environment or according to the occurrence of exceptional situations using the approaches for process adaptability described in the previous section. Adaptation of the SBAs can be thus triggered by the following drivers:

1. *Recommendations from the requirements engineer*: Typically, enterprises have contract relationships with other business partners. This fact is reflected in the set of service implementations that may be used in service compositions. These partner service implementations usually meet the requirements specified by the requirements engineers in the enterprise. In some cases however, due to the dynamic nature of the service market, new relationships are established with other (previously unknown) partners. If the newly introduced service is better and/or more appropriate (e. g. cheaper, faster etc.), the requirements engineer is entitled to recommend the use of this new service (adaptation trigger).
2. *Results from online tests*: Online tests of service implementations are performed during the runtime of the SBA. If a test of a service fails the online testing expert may recommend to discontinue using the service (adaptation trigger).
3. *Service failures*: During the execution of process instances the discovery, selection and invocation of a service is delegated to the service middleware (Note that in the case of static binding strategy the discovery and selection steps are not needed). The middleware is responsible for tackling the situation in which selected services exhibit failures when invoked. In case the middleware cannot discover any service compliant with the requirements provided by the process model and the process execution environment, and as specified by the requirements engineer, the service selection criteria may be adapted. The new alternative criteria are then used to discover and select another service that can perform on behalf of the process (adaptation trigger).

All three mechanisms are explained in the next section along with our wine example.

4 Example

The presentation of the mechanisms enabling the reaction to the adaptability drivers presented in the previous section will be demonstrated with the help of the example introduced in section 1. Figure 2 depicts the process described by the example. The figure also shows the available service implementations assigned to the tasks either by static or dynamic binding – e. g. the “Grapes retailer” service is bound to the “Buy grapes” task while three different shipment services are available to support the “Ship wine” and “Receive delivery note” tasks.

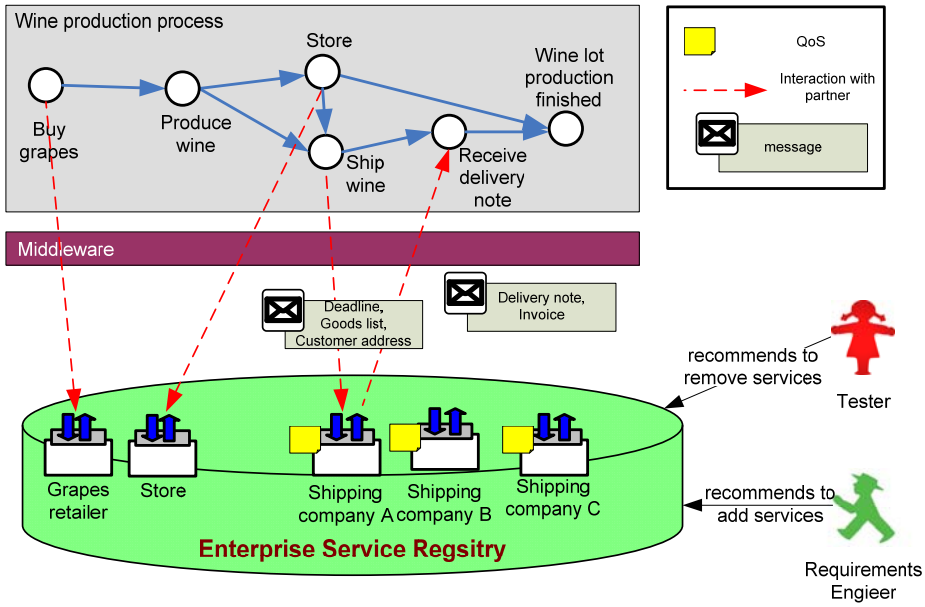


Figure 2. Example of a Wine Production Process.

The process is executed by a process engine which relies on a service middleware [20]. The middleware supports among others the discovery and invocation of services that the process composes (infrastructure services). If the binding strategy for a task is static, the middleware can only invoke the concrete service implementation as defined in the process definition; the middleware receives the input data for the service and its endpoint reference (EPR [20]). After the invocation of the service implementation the middleware returns the result from the service invocation to the process engine and hence to the process instance. The middleware performs an additional step if the binding strategy is dynamic. Before a service invocation can be carried out a discovery of a set of service implementations compliant with the requirements (provided in the process definition) is triggered and the most appropriate service implementation is selected.

All the adaptation mechanisms listed in the previous section require service invocation. Depending on the binding strategy (static vs. dynamic) service discovery and selection may (dynamic binding) or may not (static binding) be needed. In the rest of this section we present the adaptation approaches that can be triggered as a possible reaction to the adaptation drivers we cover here. All these approaches make use of the infrastructure services supported by the service middleware.

4.1 Adaptation Triggered by the Requirements Engineer

As new services become available outside the enterprise service registry, the requirements engineer needs to decide whether the newly available service should be used. This decision is made according to the mechanism described in subsection 2.3. Once a

decision to use the new service is made, the requirements engineer has two choices to make use of the service in the service composition: Using the static binding strategy, the requirements engineer may exchange the static service reference of one or more tasks directly in the process model and to either migrate all instances, selected instances or no instances to this new process model. If the dynamic binding strategy is used, it is sufficient to register the new service in the service registry. Due to the enhanced service characteristics, the new service implementation will be given preference by the middleware's service discovery component.

Assume that the shipment service of company D becomes available and the requirements engineer decides to use it for the above service composition. In a static binding scenario the requirements engineer initiates the exchange of the static reference for the task "Ship wine" from the shipment service of company A to the shipment service of company D. In a dynamic binding scenario, the shipment service of company D is registered in the enterprise service registry and the middleware will automatically prefer it over the shipment service of companies A–C because of its better suitability to the requirements.

Requirements engineering for service based applications is a relatively new field. Existing approaches cover the description of requirements for SBAs [e. g. 8], the discovery of services using goal models [e. g. 4], the consistency check between goal models and workflows [e. g. 12] and the continuous goal-driven optimization of SBAs [e. g. 5].

4.2 Adaptation Triggered by Online Testing

As described above, online testing may find failures, which require an adaptation. The adaptation action required depends on the binding strategy. If the binding strategy prescribes static binding, the faulty service must be replaced in the process model (usually in the deployment information). After this replacement the process model must be re-deployed and it must be decided whether all running instances (instance migration), selected running instances or all new instances should be migrated to the new process model.

Assume that the shipment service of company A should be tested in our example above. The tester generates test cases with deadlines, lists of goods to be shipped and customer address data as input. Then the output is checked against the expected output. In our scenario, the expected output is a delivery note and an invoice. If the service of shipping company A cannot deliver a delivery note, this failure is reported and an adaptation is triggered. The service of shipping company A is then removed from the process model (static binding) or from the registry (dynamic binding) and replaced by another service, e. g. the shipping service of company B.

Numerous testing techniques exist in the literature. An overview of testing techniques for services is given in [21, pp. 48]. Especially test case generation from existing Business Process Execution Language (BPEL) specifications is described in [22, 23]. These existing approaches have to be adapted for using them in parallel to process instance execution.

4.3 Adaptation Triggered by Service Faults

The third driver for adaptation of service compositions are failures of the composed services. A mechanism for tackling such failures for both statically and dynamically bound services is needed to ensure the completion of the process instance. This mechanism involves discovery and selection of a new service capable of performing the task in the process with similar requirements.

In the dynamic binding strategy it is up to the middleware to discover and invoke a service compliant to both the functional and quality requirements specified in the process definition. If the middleware in not able to discover such a service implementation, an alternative set of (quality) requirements may be specified in the process model and hence used by the middleware for service discovery and selection. In case of a static binding strategy, the middleware does not have to discover a new service and, hence, the service composition must be repaired manually or automatically using an alternative set of requirements to the service.

In the context of the wine production example, consider for instance that shipment company A ships goods for less than 100,000.00 € and within one week. Assume now that the shipment service of company A is temporarily not available. As an alternative service implementation the wine producer might be willing to pay more than originally specified to a shipping company that can comply with the time constraint (e. g. shipment company B). The discovery and invocation of such a service is performed by the middleware provided that the alternative quality requirements are made available to it by the service composition definition.

Adaptation mechanisms for BPEL processes are presented in [18, 24], where the quality requirements are specified as Web Ontology Language for Web Services (OWL-S) service descriptions and encapsulated in WS-Policies. Prototypical implementations of an infrastructure including a BPEL engine and a service bus are also available for both the OWL-S and Web Service Modelling Ontology (WSMO) based approaches.

5 Conclusion

In this paper we integrated requirements engineering, online testing and state-of-the-art adaptation mechanisms for service compositions. The paper shows clearly that the dynamic binding strategy driven by pre-described service requirements is beneficial over the static binding strategy. The dynamic binding strategy resolves automatically all service faults, e. g. due to unavailable service implementations. In addition, requirements engineering and online testing only need to interact with the enterprise service registry and have no other influence on the existing service middleware. While the online testing aims to remove faulty services from the registry, the requirements engineering activity aims to add new and innovative services to it, which lead to a better fulfilment of the SBA’s requirements. The static binding strategy, however, requires a modification of the process model and its redeployment.

We are also able to identify future research for the integration of online testing and workflow modelling as well as the integration of requirements engineering with work-

flow modelling. Online testing techniques require a certain sequence of tests, e. g. the tester needs to know, which service implementation to test first. This sequence depends heavily on the workflow(s) in which the service implementation is used. In addition, this paper and the cited online testing techniques cover only service testing. However, it is also important to test the whole service composition, e. g. the workflow itself (integration test). Techniques need to be developed and integrated with adaptation techniques.

For the requirements engineering field, the most predominant problem is the mapping of the requirements to service descriptions, e. g. to OWL-S. This mapping ensures that services identified as superior in the requirements engineering activity, are actually given preference during service discovery carried out by the middleware. In addition, workflow instances may be tailored to the so-called context-factors, e. g. workflows may be adapted to certain users or a certain device. These context-aware workflow adaptations require a tight integration of workflow and requirements engineering research.

Acknowledgements

The research leading to these results has received funding from the European Community's Seventh Framework Programme FP7/2007-2013 under grant agreement 215483 (S-Cube).

References

1. Kephart, J.O., Chess, D.M.: The Vision of Autonomic Computing. IEEE Computer **36** (2003) 41-50
2. Nitto, E.D., Ghezzi, C., Metzger, A., Papazoglou, M., Pohl, K.: A Journey to Highly Dynamic, Self-Adaptive Service-Based Applications. Automated Software Engineering (2008) 257-402
3. Pohl, K.: Requirements Engineering - Grundlagen, Prinzipien, Techniken. dpunkt (2008)
4. Gehlert, A., Bramsiepe, N., Pohl, K.: Goal-Driven Alignment of Services and Business Requirements. 4th International Workshop on Service-Oriented Computing Consequences for Engineering Requirements (SOCCER 2008), Barcelona, Spain (2008)
5. Gehlert, A., Heuer, A.: Towards Goal-Driven Self Optimisation of Service Based Applications. 1st International Conference of the Future of the Internet of Services (ServiceWave 2008). Springer, Madrid, Spain (2008), 13–24
6. Bresciani, P., Perini, A., Giorgini, P., Giunchiglia, F., Mylopoulos, J.: Tropos: An Agent-Oriented Software Development Methodology. Autonomous Agents and Multi-Agent Systems **8** (2004) 203–236
7. Castro, J., Kolp, M., Mylopoulos, J.: Towards Requirements-Driven Information Systems Engineering: The Tropos Project. Information Systems **27** (2002) 365–389
8. Aiello, M., Giorgini, P.: Applying the Tropos Methodology for Analysing Web Services Requirements and Reasoning about Qualities. UPGRADE: The European Journal for the Informatics Professional **5** (2004) 20–26

9. Lau, D., Mylopoulos, J.: Designing Web Services with Tropos. International Conference on Web Services (ICWS 2004), San Diego, CA, USA (2004) 306–313
10. Misra, S.C., Misra, S., Woungang, I., Mahanti, P.: Using Tropos to Model Quality of Service for Designing Distributed Systems. International Conference on Advanced Communication Technology (ICACT 2006), Phoenix Park, Gangwon-Do, Republic of Korea (2006) 541–546
11. Penserini, L., Perini, A., Susi, A., Mylopoulos, J.: From Stakeholder Needs to Service Requirements. 2nd International Workshop on Service-Oriented Computing: Consequences for Engineering Requirements (SOCCER 2006), Minneapolis, Minnesota, USA (2006) 8–17
12. Pistore, M., Roveri, M., Busetta, P.: Requirements-Driven Verification of Web Services. Electronic Notes in Theoretical Computer Science **105** (2004) 95–108
13. Giorgini, P., Mylopoulos, J., Nicchiarelli, E., Sebastiani, R.: Formal Reasoning Techniques for Goal Models. Journal on Data Semantics. Springer, Berlin, Heidelberg (2003) 1–20
14. Swanson, E.B.: The Dimensions of Maintenance. International Conference on Software Engineering (ICSE 1976), San Francisco, CA, USA (1976) 492–497
15. Baresi, L., Nitto, E.D., Ghezzi, C.: Toward Open-World Software: Issue and Challenges. IEEE Computer **39** (2006) 36–43
16. Hielscher, J., Kazhamiakin, R., Metzger, A., Pistore, M.: A Framework for Proactive Self-Adaptation of Service-based Applications Based on Online Testing. 1st International Conference of the Future of the Internet of Services (ServiceWave 2008), Madrid, Spain (2008)
17. van der Aalst, W.M.P., Jablonski, S.: Dealing with Workflow Change: Identification of Issues and Solutions. International Journal of Computer Systems Science and Engineering **15** (2000)
18. Karastoyanova, D., Houspanossian, A., Cilia, M., Leymann, F., Buchmann, A.: Extending BPEL for Run Time Adaptability. 9th International Conference on Enterprise Distributed Object Computing (EDOC 2005), Enschede, The Netherlands (2005) 15–26
19. Karastoyanova, D., Leymann, F., Buchmann, A.P.: An Approach to Parameterizing Web Service Flows. In: Benatallah, B., Casati, F., Traverso, P. (eds.): 3rd International Conference on Service Oriented Computing (ICSOC 2005). Springer, Amsterdam, The Netherlands (2005) 533–538
20. Weerawarana, S., Curbera, F., Leymann, F., Ferguson, D.F., Storey, T.: Web Services Platform Architecture: Soap, WSDL, WS-Policy, WS-Addressing, WS-BPEL, WS-Reliable Messaging and More. Prentice Hall (2005)
21. Pernici, B., Metzger, A. (eds.): S-Cube Project Deliverable PO-JRA-1.3.1 - Survey of quality related aspects relevant for SBAs (2008)
22. Lübke, D.: Unit Testing BPEL Compositions. In: Baresi, L., Nitto, E.D. (eds.): Test and Analysis of Web Services. Springer (2007) 149–171
23. Dong, W.-L., Yu, H., Zhang, Y.-B.: Testing BPEL-based Web Service Composition Using High-level Petri Nets. 10th IEEE International Enterprise Distributed Object Computing Conference (2006) 441–444
24. Karastoyanova, D., van Lessen, T., Nitzsche, J., Wetzstein, B., Wutke, D., Leymann, F.: Semantic Service Bus: Architecture and Implementation of a Next Generation Middleware. Proceedings of the 23rd International Conference on Data Engineering Workshops (ICDE 2007), Istanbul, Turkey (2007)