Grant Agreement N° 215483

Title:	*Derivation of QoS And SLA Specifications*
Authors:	*TUW, UniHH, UPM, USTUTT, UniDuE, FBK*
Editor:	*Dragan Ivanović (UPM)*
Reviewers:	*Harald Psaier (TUW)*
	Kristof Hamann (UniHH)
Identifier:	*CD-JRA-2.2.5*
Type:	*Contractual Deliverable*
Version:	*1.0*
Date:	*11 February 2011*
Status:	*Final*
Class:	*External*

Management Summary

This deliverable describes and presents contributions related to the mechanisms and algorithms for derivation of QoS/SLA specifications for services and service compositions.

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).

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

Deliverable Overview

1.1 Introduction

The goal of the S-Cube work package WP-JRA-2.2 is to establish the foundation for Quality-of-Service (QoS) aware adaptable service compositions. Such compositions are able to adapt themselves in response to changes in the QoS characteristics of the invoked services and systems within which they execute and with which they communicate. QoS characteristics of a services are specified by different Service-Oriented Architecture (SOA) functional layers, from the infrastructure, to individual "atomic" services, properties of other services, to high-level business processes. Therefore, service compositions and their adaptation are influenced by a combination of those factors. The Service Level Agreements (SLA) define constraints to which the actual QoS of an executing service instance is expected to conform. The work included in this work package thus relies on inputs, capabilities and QoS requirements from the Business Process Management layer and the Service Infrastructure layer, and uses them throughout service composition lifecycle, including modeling, verification, monitoring, and adaptation.

The goal of this deliverable is to present contributions that address the problem of automatic derivation of QoS/SLA specifications for Web services and service compositions. This includes different techniques that can be used in different stages of the service life cycle (e.g., at design time and/or run time), for either synthesis, decomposition, or verification of QoS/SLA specifications for service compositions.

There are several reasons why automating derivation of QoS/SLA specifications has significant practical importance in Service-Oriented Systems. One, of course, is the sheer size of globally accessible service ecosystems that frequently cross organizational boundaries, where QoS-awareness in service selection, matching, execution and adaptation would be practically unattainable without a degree of automation. But, more concrete scenarios for application of such an automated analysis are related to the research challenges identified within the JRA-2.2 work package (discussed in Section 1.4 on page 9). These include run-time monitoring of QoS/SLA conformance for service compositions, design-time and run-time analysis and prediction of their QoS properties, and QoS-aware adaptation.

1.2 Deliverable Structure

This is a paper-based deliverable that presents nine contributions developed within the framework of S-Cube. This first introductory chapter gives an overview of contributions, describes how they relate to the goal of this deliverable, places the contributions in the S-Cube Integrated Research Framework (IRF), and puts them in relationship with other S-Cube work packages.

Chapter 2 on page 12 gives a more detailed description of each contribution included in the deliverable. Besides the basic information on the contribution title, the participating S-Cube partners, keywords related to the content and the state of submission/publication, each description provides background, problem statement, contribution relevance to the deliverable, its summary, evaluation and conclusions.

The actual papers are in the appendix at the end of the deliverable.

Finally, Chapter 3 on page 30 gives a brief summary of the findings contained in the contributions, and provides an outline of future work. It is followed by the bibliography.

1.3 Overview of the Contributions

This deliverable includes the following contributions:

- *Integrating Perfective And Corrective Adaptation Of Service-Based Applications*
- *Building Dynamic Models of Service Compositions With Simulation of Provision Resources*
- *Privacy Preservation Within a Business Process Mashup for Web Service Oriented Applications*
- *Automatic Fragment Identification in Workflows Based on Sharing Analysis*
- *Runtime Decomposition of QoS Specifications For Distributed Processes*
- *Behavior Monitoring in Self-Healing Service-Oriented Systems*
- *Monitoring of SLAs in Service Choreographies*
- *Runtime Prediction of SLA Violations in Service Compositions*
- *Monitoring, Prevention and Prediction of SLA Violations in Service Compositions*

The contributions presented in this deliverable address the problem of automatic derivation of QoS specifications for service compositions from several different perspectives, i.e., taking into account different dimensions of Service Oriented Systems. The approaches presented tend to take into account specific features of application and its context, and try to meet specific needs of service designers, users and providers. They also naturally extend previous work within this workpackage on coordinated service compositions [25, 28] and fragmentation [27]. Their, sometimes close, relationship to topics in other S-Cube workpackages is discussed in section 1.5 on page 9. In the following, we give a brief overview of the key features of the contributions and how they contribute to automatic derivation of QoS/SLA specifications for service compositions, which is the subject of this deliverable. More detailed information on the particular contributions can be found in Chapter 2, and in the actual papers attached in the Appendix (from p. 35 onwards).

Design vs. Run-Time Approaches

The contributions can be classified depending on whether the derivation of QoS specification (primarily) takes place at design or run time. In the former case, the analysis is based on deducing the QoS properties based on the description of the composition using some adequate formalism, a set of assumptions about the environment of and inputs to the composition, as well as the properties of component services used in the composition (which, in turn, may be either assumed or a result of a similar analysis). In this category fall the following contributions: *Building Dynamic Models of Service Compositions With Simulation of Provision Resources* (p. 14), *Privacy Preservation Within a Business Process Mashup for Web Service Oriented Applications* (p. 17), and *Automatic Fragment Identification in Workflows Based on Sharing Analysis* (p. 19). In these cases, the analysis produces QoS specifications as estimates, approximations, or exact values, ahead of the actual execution. Consequently, designers can use these approaches to test, refine and reengineer their composition designs in order to meet specific QoS-related objectives.

Other contributions present approaches that are applied at run time, either during test (or training) runs, or in a production environment. These include *Runtime Decomposition of QoS Specifications For Distributed Processes* (p. 21), *Behavior Monitoring in Self-Healing Service-Oriented Systems* (p. 23)

Monitoring, Prevention and Prediction of SLA Violations in Service Compositions (p. 28), *Monitoring of SLAs in Service Choreographies* (p. 25), and *Runtime Prediction of SLA Violations in Service Compositions* (p. 26). These approaches use the empirical data from monitoring, supplied by the service execution infrastructure, both from the current execution of a composition and the historic records, to detect service misbehaviors, assess likelihoods of deviations from the nominal SLA, or trigger healing or adaptation mechanisms that may prevent such deviations. Here, the derived QoS specification is effectively an assessment of deviations from the nominal SLA at a given point of execution, in the remaining execution time, or within a fragment of the composition. An integrated framework for the application of those approaches for performing QoS-driven adaptation is presented in *Integrating Perfective And Corrective Adaptation Of Service-Based Applications* (p. 12).

It should be noted that the division between design time and run time approaches is not rigid. On the one hand, design time approaches may depend on historical data from monitoring to calibrate their models, and can, in principle, be re-applied at convenient points of execution at run time to obtain more precise predictions by taking into account the actual data and QoS of component services. On the other hand, runtime techniques usually depend on specific design time choices and models used to represent quality requirements, and are often meant to be used as an input for refining the composition design within the continuous requirements-design-deployment-adaptation lifecycle loop.

Synthesis vs. Decomposition of QoS Specifications

Of those contributions that are concerned with inferring the QoS levels within a service composition (rather than estimating the level of deviation from a given SLA model), either at design or run time, some aim at *synthesis* of QoS specification for the composition (and possibly its fragments), while others are concerned with *decomposition* of the composition level QoS specification, i.e., derivation of QoS specifications for composition fragments.

The contribution in the synthesis group are *Building Dynamic Models of Service Compositions With Simulation of Provision Resources* (p. 14) and *Automatic Fragment Identification in Workflows Based on Sharing Analysis* (p. 19), while those in the decomposition group include *Runtime Decomposition of QoS Specifications For Distributed Processes* (p. 21) and *Privacy Preservation Within a Business Process Mashup for Web Service Oriented Applications* (p. 17).

Support for fragmentation can be found in both cases. One motivation is the prospect of distributed enactment of business processes, which requires either design time or run time knowledge of QoS properties for potential fragments, including, for instance, security and privacy attributes. Another motivation is reuse of process fragments in service mashups, which may rely on fragment repositories to provide users with on demand, ad-hoc tailored tools for performing computations within a Service-oriented Architecture setting.

Prediction vs. Detection of Runtime Violations

Several approaches that deal with estimating or monitoring run-time QoS behavior focus on prediction of possible SLA violations ahead of time. The contributions *Monitoring, Prevention and Prediction of SLA Violations in Service Compositions* (p. 28), *Runtime Prediction of SLA Violations in Service Compositions* (p. 26) and *Building Dynamic Models of Service Compositions With Simulation of Provision Resources* (p. 14) rely on composition models and historically collected data to predict possible SLA violations, either on the level of the executing instance at run time, or on the level of a simulated set of executing instances at design time. This information can be used by service providers (who are a contractual party in an SLA) to tune their provision infrastructure so that it can meet the load under given SLA constraints, or to gauge the offering of SLA given the available resources and load scenarios. Alternatively, predictions can be used to prepare adaptation mechanisms on the instance level.

The contribution *Behavior Monitoring in Self-Healing Service-Oriented Systems* (p. 23), on the other hand, illustrates how the detection of composition misbehaviors (from the point of view of the functional and non-functional specifications) can be used to trigger self-healing of composition instances at runtime.

1.4 Key Research Challenges and Results

The S-Cube Integrated Research Framework (IRF) defines the following *research challenges* for the workpackage JRA 2.2:

Formal Models and Languages for QoS-Aware Service Compositions: This challenge deals with formal models and Languages for QoS-aware service compositions. The challenge is substantiated by the facts, that firstly, there are no formal models for service compositions available that take into account the QoS and behavioral characteristics of these compositions and secondly, that the formal models are extremely important to guarantee that the final result of a composition services possesses the required characteristics.

Monitoring of Quality Characteristics of Service Orchestrations and Service Choreographies: In the context of QoS-aware service compositions, the focus of this challenge lies on monitoring of quality characteristics of service orchestrations and service choreographies. As service compositions implement business processes and at the same time run on IT infrastructure, their quality characteristics are influenced by both process-level and infrastructure-level metrics. A holistic monitoring approach for quality characteristics of service compositions involves monitoring of service orchestrations in terms of both process-level and infrastructure level factors and in addition monitoring of quality characteristics across participants in service choreographies.

Analysis and Prediction of Quality Characteristics of Service Compositions: When monitoring of quality characteristics of service compositions reveals that KPIs do not meet their target values, users are interested in finding out the causes and the most influential factors in order to be able to adapt the composition to prevent those violations in a future. Analysis and prediction mechanisms for quality characteristics will be devised, which are integrated with the monitoring mechanisms and provide input to the adaptation framework on which quality characteristics to adapt.

QoS Aware Adaptation of Service Compositions: Adaptations of Service Compositions driven by changes in the environment and in particular by changes in QoS characteristics still remains a major challenge in service-based applications. Mechanisms for enabling such adaptations will be developed, and the major drivers for adaptation will be defined. The influence of the BPM and SI layers of SBAs on the adaptation of SC must be taken into account to ensure consistency of the adaptation steps.

The relationship of the contributions to the challenges is given in Table 1.1 on page 11. The table cells briefly describe research questions addressed by the contributions, and the main features of the results obtained. Detailed descriptions of the challenges, research questions and research results can be found in the IRF.

1.5 Relationship to Other Workpackages

The subject of the deliverable is very closely related to the work package WP-JRA-1.3: *Quality Definition, Negotiation, and Assurance*. As described above, the motivation for many approaches is closely associated with quality assurance. In particular, the contribution *Runtime Prediction of SLA Violations in Service Compositions* (p. 26) has been presented in the deliverable CD-JRA-1.3.4: *Initial Set of Principles, Techniques and Methodologies for Assuring End-to-End Quality and Monitoring of SLAs*. The contribution *Integrating Perfective And Corrective Adaptation Of Service-Based Applications* (p. 12) is also related to the following research challenges from other work-packages:

- *Quality Prediction Techniques to Support Proactive Adaptation* (WP-JRA-1.3)

- *Run-time Quality Assurance Techniques* (WP-JRA-1.3)
- *Comprehensive and integrated adaptation and monitoring principles, techniques, and methodologies* (WP-JRA-1.2)
- *Proactive Adaptation and Predictive Monitoring* (WP-JRA-1.2)
- *Mixed initiative SBA adaptation* (WP-JRA-1.2)

Contribution	**Research Questions Addressed per Challenges in WP-JRA-2.2**			
	Formal Models and Languages for QoS-Aware Service Compositions	Monitoring of Quality Characteristics of Service Compositions	Analysis and Prediction of Quality Characteristics of Service Compositions	QoS Aware Adaptation of Service Compositions
Runtime Decomposition of QoS Specifications For Distributed Processes (UniHH)				Specification of Non-functional Parameters for Runtime Decomposition. Algorithm for Runtime Decomposition of Non-functional Requirements.
Behavior Monitoring in Self-Healing Service-Oriented Systems (TUW)		Foundations of Analysis for Service-Based Systems. Monitoring and Analyzing Influential Factors of Business Process Performance.	Variability Modeling and QoS Analysis of WS Orchestrations.	
Runtime Prediction of SLA Violations in Service Compositions (ustutt+tuw)			Runtime prediction of KPI and SLA violations based on machine learning techniques. Results focus on service compositions.	
Monitoring, Prevention and Prediction of SLA Violations in Service Compositions (tuw+ustutt)		Foundations of Analysis for Service-Based Systems. Monitoring and Analyzing Influential Factors of Business Process Performance.	Runtime prediction of KPI and SLA violations based on machine learning techniques. Results focus on service compositions.	Adaptation of QoS-aware Service Compositions based on Influential Factor Analysis and Prediction.
Integrating Perfective And Corrective Adaptation Of Service-Based Applications (UniDuE, USTUTT, FBK)				Adaptation of QoS-aware Service Compositions based on Influential Factor Analysis and Prediction.
Building Dynamic Models of Service Compositions With Simulation of Provision Resources (UPM, TUW)			Dynamic modeling of service orchestration and provision infrastructure. Using simulation models and monitoring data for calibration.	
Privacy Preservation Within a Business Process Mashup for Web Service Oriented Applications (Université Paris Descartes)			Analyzing and predicting privacy as a QoS attribute for reusable fragments and compositions assembled from them.	
Monitoring of SLAs in Service Choreographies (ustutt)		Process Monitoring in Service Choreographies. Results cover cross-organizational process monitoring.		
Automatic Fragment Identification in Workflows Based on Sharing Analysis (UPM)	Formally representing workflows in a Horn clause program form.		Applying sharing-based analysis to the the problem of composition fragmentation. Focuses on data-dependent QoS such as privacy and security.	

Table 1.1: Relationship of the contribution to the IRF challenges for WP-JRA-2.2.

Chapter 2

Contributions to Derivation of QoS and SLA Specifications

2.1 Integrating Perfective And Corrective Adaptation Of Service-Based Applications

Contributing Partners: UniDuE, USTUTT and FBK

Status: Book chapter, accepted and published in Service Engineering: European Research Results, S. Dustdar and F. Li, Eds. Springer, 2010, pp. 137-169.

2.1.1 Background

Organizations increasingly rely on the flexibility offered by service-based applications (SBAs). This flexibility allows those applications to operate in a highly dynamic world, in which the level and quality of service provisioning, (legal) regulations, as well as requirements keep changing and evolving. To respond to those changes, service-based applications need to modify their functionality and quality dynamically depending on the usage situation, context, and deployment platform.

In addition, those applications will need to react to failures of the constituent services to ensure that they maintain their expected functionality and quality. In such a dynamic setting, evolution and adaptation methods and tools become key to enable SBAs to respond to changing conditions.

The adaptation of an SBA can address various goals, such as (1) correcting faults contained in the SBA (corrective adaptation), and (2) adapting the SBA to new and yet unknown requirements (perfective adaptation).

2.1.2 Problem Statement

When building adaptive SBAs that address two or more adaptation goals, precautions must be taken to ensure that the interplay and the interactions between the different types of adaptations are considered, as otherwise this can lead to conflicting adaptations, which need to be avoided. As an example, to address the goal of corrective adaptation, the service-based application might aim at replacing a failed service A with a service B, while at the same time the SBA, in order to address the aim of perfective adaptation, aims to replace service A with a service C, which promises to provide a better quality of service than service A. In fact, coordinating various adaptation goals is considered one of the key challenges in self-adaptive software [29].

2.1.3 Contribution Relevance

In this contribution we present innovative solutions that specifically focus on understanding how the need for the corrective adaptation of a SBA must be aligned and synchronized with the opportunity for the perfective adaptation of a SBA.

To demonstrate how conflicts between adaptation goals can be avoided, we focus on the two adaptation goals introduced above: corrective and perfective adaptation. More specifically, we exploit the following techniques to determine the demand for an adaptation of the SBA: (1) Corrective adaptation based on online testing; (2) Perfective adaptation based on requirements engineering. Both of the above techniques share the characteristic that they are pro-active in nature, i.e., both techniques lead to "predictive" adaptation triggers. In the case of online testing, the failure of a service could point to a problem of the SBA (which involves this service) in the future. In the case of recommendations from Requirements Engineering (RE), this provides the possibility to improve the SBAs and to anticipate future requirements.

A central element of our approach is to use a central enterprise service registry. This registry contains references to in-house services, e.g. those services provided by the enterprise itself, and to external services, e.g., those services, that are provided by external service providers. Only these services are allowed to be used in the enterprise's service-based applications.

In [29], Salehie and Tahvildari stress that "coordinating [. . .] goals at different levels of granularity is one of the significant challenges in self-adaptive software." As a result of the literature survey carried out in their paper, the authors reach the conclusion that only very few approaches address more than one goal of adaptation (or, self-* property). In addition, they observe that most approaches that address more than one goal do not systematically coordinate those goals. One concrete, architecture-based approach that addresses multiple goals is introduced by Cheng et al. in [7]. However, rather than addressing high-level goals, such as perfective and corrective adaptation, which are addressed in our approach, the authors address conflicting situations between more fine-grained objectives, such as performance and other quality of service characteristics.

2.1.4 Contribution Summary

Service-based Applications (SBAs) can be dynamically adapted to address various goals, which include (1) aiming to better achieve the users' requirements (perfective adaptation), and (2) repairing and preventing failures (corrective adaptation). When building applications which aim at addressing more than of such goals, it is important to understand the interplay of these different adaptation goals. Otherwise this can lead to conflicting adaptations. Our contribution constitutes of a framework to integrate and align perfective and corrective adaptations, while addressing the problems that are due to the interactions between these two kinds of adaptation.

The framework uses requirements engineering techniques to trigger perfective adaptation and online testing techniques to trigger corrective adaptations. Based on the above techniques, this chapter investigates the interplay and interaction between the two types of adaptation. We demonstrate how perfective and corrective techniques can be integrated in a meaningful way to support the overall adaptation requirements of the service-based applications, while avoiding the above problems. As a solution, we propose exploiting an enterprise service registry, which restricts the ways in which a service-based application can be adapted.

2.1.5 Contribution Evaluation

The approach is demonstrated based on a concrete scenario within the telecommunication domain (which has been chosen as the overall application domain for the book in which the book chapter has been published). The scenario is used to demonstrate the application and usefulness of our approach. Specifically, we show how our scenarios support the business goals "Provide mobile phone number portability" and "Provide mobile services portability".

2.1.6 Conclusions

In our contribution we have shown how perfective adaptation (as enabled by requirements engineering) could be integrated and synchronized with corrective adaptation (as enabled by online testing) in a meaningful way. Thereby, our contribution has demonstrated a possible solution to the problem of how to handle multiple adaptation goals in service-based applications, which has been identified in the literature as one of the key challenges of adaptive software systems. Our solution has relied on state-of-the-art adaptation mechanisms for service compositions as well on the use of an enterprise service registry as a mechanism for achieving the synchronization.

In this context, we have observed that dynamic binding strategy driven by pre-described service requirements is beneficial over the static binding strategy. The dynamic binding strategy automatically respects the decisions of the requirements engineer to add new services and of the online tester to remove faulty services without human intervention. The static binding strategy, however, requires a modification of the workflow model, its re-deployment and ad-hoc adaptations for already running workflows. Since the services are restricted by the enterprise service registry, static binding does not prevent any unintended adaptation (its initial purpose) and should, therefore, not be used in this scenario.

We see the following, potential future research directions: (1) Integration of requirements engineering and workflow adaptability; (2) Integration of online testing and workflow adaptability; (3) Extension to other adaptation goals.

2.2 Building Dynamic Models of Service Compositions With Simulation of Provision Resources

Contributing partners: UPM, TUW

Status: Accepted and published in the proceedings of the 29th International Conference on Conceptual Modeling (ER 2010), Vancouver, Canada, 1-5 November 2010.

Keywords: Service Compositions, Business Process Modeling, Quality of Service, Simulation

2.2.1 Background

Service compositions have been studied thoroughly over recent years and different models to define service compositions have emerged [34]. Approaches like BPEL [32] or YAWL [35] define service compositions in a top down manner using specific notation. At the same time, abstract service composition models include different strands of Petri Nets [8] and process calculi [36].

Key to business usability of service compositions is conformance of their Quality of Service (QoS) attributes with Service-Level Agreements (SLA). Both are intimately related to monitoring and adaptation capabilities [21]. From the computational point of view, resource utilization management, especially the ability to scale computational resources to the expected level of demand, may have drastic impact on response time and failure rates.

2.2.2 Problem Statement

QoS behavior and the SLA constraints that the service provider may be willing to offer its users depend on two main groups of factors. The first group arises from the structure and logic of the given service composition, and from the QoS attributes of the component services it uses. The second group of factors depend on the capacity and up-/down-scaling policies of the provider's infrastructure on which the provided compositions (and possibly some of the component services execute). To adjust the level of QoS in line with SLA requirements, the provider can try adapt the composition, the capacity of the provision chain (the infrastructure), or both. While the first approach has been extensively studied, the problem of choosing and triggering adequate policies for adapting the behavior of the provision chain has been largely left open.

This paper tries to address the stated problem by proposing an approach for automatic derivation of dynamic, continuous-time models of behavior of service orchestrations. Those dynamic models are the core for developing simulation models that are used to qualitatively and quantitatively test the behavior of the composition provision chain under various interesting input regimes and to test potential resource management (e.g., up- and down-scaling) policies to assure that the specified QoS levels are met.

2.2.3 Contribution Relevance

Some aspects of dynamic modeling of service provision chains have already been studied [17]. Such modeling approaches in general apply the tools of system dynamics [30]. We extend the previous work on applying system dynamics to (atomic) service provision management [2], by automatically deriving the quantitative indicators for a service provision chain (number of executing instances, invocation and failure rates) from the structure of a particular composition being provided, as well as from a model of computational resources involved in provision. These quantitative indicators are, in fact, the expected values of QoS attributes for the service composition, derived from the structure of the orchestration, under the given input regime and provision resource model.

2.2.4 Contribution Summary

We propose an approach that utilizes a common (composition-independent) service provision framework that divides the modeling concern into the composition and the computational resource parts. The former has been studied in several interesting, but specific, cases [17, 24], and some examples of a more systematic approaches to building such dynamic composition models based on QoS constraints have been demonstrated [42]. Our intention is to propose a generic method of converting descriptions of orchestrations in the form of place-transition networks (PT-nets) [10] into dynamic models, in a manner that ensures composability of orchestrations within choreographies. We believe that such automatic dynamic model generation is a prerequisite for practical application.

The starting point is a conceptual model for dynamic representation of service compositions (orchestrations and choreographies). The goal of these models is to represent how the numbers of executing activites and rates of activity-to-activity transitions vary over time. For that, we use a conceptual framework for the continuous-time (CT) composition modeling. The fundamental building block of an *orchestration* CT (OCT) *model* is a *variable* that has a time-varying value. Compared to a PT-net model, *activity variables* are attached to transitions inside, and their value is the expected number of executing instances of a given activity at each point in time. In our example, each service in the data cleansing process would be represented with an activity variable, that shows the expected number of concurrently executing instances of the service at any moment.

The OCT model can be automatically built from the structure, and some behavioral parameters, of a service orchestration. One possibility, described in the paper, is to take a PT-net description of the orchestration, together with branching probabilities, and convert it automatically into a set of ordinary differential equations. Variables that appear in that equation set correspond to places and transitions of the PT-net. We assume an idealized execution environment where tokens, corresponding to process running instances, do not stay for a noticeable period of time in places before triggering a transition. On the other hand, transitions, which correspond to the activities within the orchestration, take some definite time to execute, modeled with an exponential time distribution around the mean. Thus, the transition (i.e., activity) variables are integrals (i.e., stocks), and the place variables are derivations (i.e., flows) that fill and empty the stocks.

The OCT dynamic model is fed by an input flow that shows the number of orchestration instances starting per unit of time. Therefore, different input regimes reflect on the time distribution of individual activities. In realistic service settings, some services can be long running because they involve complex computation or human-provided services, and indeed the orchestration itself may embody a long-running business process. Therefore, a dynamic model is the only possibility for studying transitional and oscillatory behaviors of the orchestration components under realistic, non-stationary input regimes. Besides, the dynamic OCT model can provide us with insights to the statistical behavior of individual instances and/or sets of instances in different stages of execution, when it comes to the execution time and the expected number of partner service invoked/executing at any moment of time. These can be obtained from the numerically calculated outflows from the OCT model given its in/outflows.

To complete the framework model of service composition provision, besides plugging the appropriate OCT model for the given composition, a computation resource model is added. In our example, we use a simple example of service threads, where new machines (hardware or virtual) can be added, and if necessary removed, to scale the computation resources up or down as necessary. The dynamic resource model derives the system load information from the OCT model, and provides the feedback in the form of extra delays and/or blocking. Some aggregate QoS quantities can be directly read from the framework model, given the input regime: the success rate, the failure rate, and the rejection rate (inverse of availability).

2.2.5 Contribution Evaluation

The empirical validation shows that the dynamic models automatically generated from orchestration descriptions correspond to the analytically derived expected values from stochastic Petri-Net simulation for median and the average running times of the orchestration. The objective was to establish whether the automatically derived dynamic (ordinary differential equations) model of a service orchestration, involving loops and parallel flows, we have compared the response of such dynamic system to a unit pulse (simulating arrival of a single request) with the empirical distribution of running times.

The goal of the validation was to show that the response of the system, given as the completed process outflow, and interpreted as a probability density curve over orchestration completion times, leads to descriptive parameters (namely mean and median) that correspond to the same parameters obtained from a set of measurements of running times from orchestration executions using dummy services.

2.2.6 Conclusions

The approach proposed in this contribution can be used for developing dynamic models of service composition provision, based on an automatically derived continuous-time ordinary differential equation model of the target orchestration. The orchestration model is calibrated using empirical estimates of average activity execution times and branching probabilities, obtained from log analysis, event signaling, or other monitoring tools at the infrastructure level. Several dynamic models of orchestrations provided together can be composed in a modular way.

The resulting dynamic model of composition provision can be used for exploring how the provision system reacts to different input rates (requests per unit of time), testing and choosing different resource management strategies and their parameters, in the style of *management flight simulators*. The model output can be used for both quantitative prediction and qualitative assessment of reference modes (growth, oscillation, stagnation, etc.).

Our future work will concentrate on developing tools that allow simultaneous model calibration/simulation using live monitoring data, along with using these data for orchestration process discovery, when the design and its representation in Petri Net form is not given.

2.3 Privacy Preservation within a Business Process Mashup for Web Service Oriented Applications

Contributing partners: Laboratoire LIPADE Paris Descartes-France, Laboratoire OASIS ENSI-Tunisie

Status: Submitted to the 9th IEEE International Conference on Web Services (ICWS-2011), Washington DC, USA, July 4-9, 2011.

Keywords: Mashup, privacy, fragmentation, Business Process

2.3.1 Background

With the advent of Web Technologies, business organizations tend more and more to improve the quality of their services and to be efficient in time and productivity to cope with the competition which becomes tough. A key strategy is to organize their business processes in an agile manner. A business process of Web Services [18] (or shortly, process) is a structured set of activities linked up together with dependency links to produce a value-added task for a particular user. It models the manner in which Web Services should be organized to get a more complex functionality. Business Process Management is an approach consisting of modeling enterprise processes. Its objective is to achieve a better global view of all enterprise processes and their interactions in order to be able to optimize, extend and automate them up with business applications.

Reuse is another way to optimize ones productivity in a record time. Indeed, using past user experience is a Web 2.0 characteristic that gains currency nowadays. In addition to saving time, this also improves the quality of the new models through the reuse of established and optimized artifacts.

2.3.2 Problem Statement

The problem is that the traditional development tools for building processes are complex. Although they make it easy to draw a diagram of the process, when it comes to define, design user interfaces and link systems together, the tasks become tedious. Also, designing new processes or even incorporate new business requirements in an existing process is complex, time consuming, costly and error prone task. Ditto for organizations which want to change processes integrations with other organizations. This does not fit organizations main objectives.

To remedy such problems, processes could be reused to get new ones and then optimize productivity in a record time. Many domains have already treated reusing concept, as in modular programming. Indeed, reusing established and optimized artifacts improves the new models quality. Whole process reuse is not interesting for users because they are most of the time interested in only a fragment of it which represents a set of activities of the main process.

Furthermore, one should take into consideration the privacy of sensitive data that processes can use. With sensitive data, a process is specific to the process owner who uses it. The process owner is allowed to see these information. However, when publishing the process, he does not want to reveal them. Thus, sensitive correspondences between two data should be avoided.

Existing works [13, 5, 33, 11, 9] proposed to split processes into multiple clusters of activities. A cluster of activities is a group of activities that are semantically close. These approaches are either on enhancing the process execution or on protecting private data enclosed within the process. They aim to allocate sets of activities of the main process to the corresponding parties to perform them.

2.3.3 Contribution Relevance

We have provided a privacy preserving framework to break up processes into multiple fragments. These fragments are valid regarding functional and non functional concerns.

They are ready to reuse and do not disclose sensitive data. Our approach is based on Formal Concepts that we extend with non-functional dependencies to avoid sensitive data discloser. We also introduce functional dependencies and other conditions to generate coherent fragments. As perspectives, our work can be extended to comprise activities that are omitted in the assembling task. It is also interesting to work on non-functional dependencies that have already been fixed by past users to apply them on current processes.

2.3.4 Contribution Summary

First, we introduce constraints that are needed to make the fragmentation. We define two types of constraints:

- Functional constraints or functional dependencies: They are applied on activities dependencies to pre serve the structure semantic
- Non-functional constraints or non-functional dependencies: They are applied on data dependencies to provide privacy-aware fragments.

Second, we break a process into independent fragments which probably better fit the end-user's requirements or some of them. This results into clusters (groups) of activities to provide a first look on reusable and privacy-aware final fragments. Thus, in our work, a cluster represents a fragment of the main process. This step is based on Formal Concept Analysis (FCA) [7] as a clustering technique that we extend with functional and non-functional dependencies

Third, we propose conditions to assemble the resulted clusters to finalize the fragmentation and to get more coherent fragments (final fragments).

2.3.5 Contribution Evaluation

We have added a new dimension when applying the FCA technique. Indeed, the relation between the activities and the data depends on the sensitive data that activities are using. We have generated privacy-aware clusters (groups of activities w.r.t. properties and private data) and represented them within a lattice. We have represented these clusters into groups of privacy following the private data.

Having the different privacy groups means that the privacy is preserved. We have defined rules to provide final fragments that are well formed and privacy-aware.

2.3.6 Conclusions

We have provided a privacy preserving framework to split processes into multiple fragments. These fragments are valid regarding functional and non functional concerns.

They are ready for reuse and do not disclose sensitive data. Our approach is based on Formal Concept Analysis that we extend with non-functional dependencies to avoid sensitive data discloser. We also introduce functional dependencies and other conditions to generate coherent and well structured fragments. As perspectives, our work can be extended to comprise activities that are omitted in the assembling task. It is also interesting to work on non-functional dependencies that have already been fixed by past users to apply them on current processes.

2.4 Automatic Fragment Identification in Workflows Based on Sharing Analysis

Contributing partner: UPM

Status: Accepted and published in the proceedings of the 8th International Conference on Service-Oriented Computing - ICSOC 2010, San Francisco, CA, US, December 7-10, 2010.

Keywords: Workflow Fragmentation, Static Analysis, Abstract Interpretation, Sharing Analysis

2.4.1 Background

Service compositions are coarse-grained components that normally implement higher-level business logic, and allow streamlining and control over mission-critical business processes inside an organization and across organization boundaries. However, the centralized manner in which these processes are designed and engineered does not necessarily build in some properties which may be required in their run-time environment. In many cases defining subsets of activities (i.e., fragments inside the workflow) according to some policy can be beneficial in order to increase reusability (by locating meaningful sets of activities), make it possible to farm, delegate, or subcontract part of the activities (if, e.g., resources and privacy of necessary data make it possible or even advisable), optimize network traffic (by finding out bottlenecks and adequately allocating activities to hosts), ensure privacy (by making sure that computing agents are not granted access to data whose privacy level is higher than their security clearance level), and others. To this end, various fragmentation approaches have been proposed [38, 6, 31]. In the same line, mechanisms have been defined for refactoring existing monolithic processes into process fragments according to a given fragment definition while respecting the behavior and correctness aspects of the original process [15, 14].

2.4.2 Problem Statement

This contribution addresses the automatic identification of fragments given an input workflow, expressed in a rich notation. The kind of fragment identification policies we tackle is based on data accessibility / sharing, rather than, for example, mere structural properties. The latter simply try to deduce fragments or properties by matching parts of workflows with predefined patterns, as [4] does for deadlocks. In contrast, the design-time analysis we propose takes into account implicitly and automatically different workflow structures.

2.4.3 Contribution Relevance

A number of QoS attributes are derived from service composition design, rather than from lower-level metrics of individual activities (as in the cases of running time or availability). For instance, the way in which a service composition handles data, in terms of its relevance, accuracy, precision, accessibility or security, constitutes the “data quality” of the composition, which is usually described with a set of criteria or policies. The proposed approach associates workflow activities with policy levels expressed as points in a (complete) lattice. Such representation is commonly used in modeling privacy and security across organizational domains [41], but can also be used to describe other properties, e.g., those that correspond to subsets of features or combinations of logical propositions. The policy levels are derived from data flows that include different operations on data, as well as logically expressed dependencies between parallel flows and complex, structured activities such as branches and loops. This is an advance in comparison with the fragmentation techniques that rely on less sophisticated pipeline models of workflows. Besides, the derived levels directly represent the relevant QoS attributes for workflow activities, such as access policies, clearance levels, or possibly other QoS attributes derived from characterization of input data and the workflow logic, such as precision, reliability, etc.

2.4.4 Contribution Summary

At the technical level, our proposal is based on the notion of *sharing* between activities. This is done by considering how these activities handle resources (such as data) that represent the state of an executing composition (i.e., process variables), external participants (such as partner services and external participants), resource identifiers, and mutual dependencies. In order to do so, we need to ensure that the workflow is deadlock free in order to infer a partial order between activities. This is used to construct a Horn clause program [19] which captures the relevant information and which is then subject to sharing and groundness analysis [23, 12, 20, 22] to detect the sharing patterns among its variables. The way in which this program is engineered makes it possible to infer, after analysis, which activities must be in the same fragment and which activities need / should not be in the same fragment. Even more interestingly, it can automatically uncover a *lattice* relating fragments of increasing size and complexity to each other and to simpler fragments, while respecting the fragment policy initially set.

We assume that any fragment definition is ultimately driven by a set of policies which determine whether two activities should / could belong to the same fragment. Fragment identification determines how to group activities so that the fragment policies are respected, while fragment checking ensures that predefined fragments abide by the policies.

We furthermore assume that the fragmentation policies can be described using a complete lattice $\langle L, \sqsubseteq, \top, \bot, \sqcup, \sqcap \rangle$, where $\sqsubseteq$ is a partial order relation over the non-empty set L of elements which *tag* the fragments, $\top$ and $\bot$ are the top and bottom elements in the lattice, and $\sqcup$ and $\sqcap$ are the *least upper bound* (LUB) and the *greatest lower bound* (GLB) operations, respectively.

The lattice formalism provides us with the necessary tools for identification of fragments. When a fragment is marked by some element $c \in L$, we can decide whether some activity a can be included in the fragment depending on whether its policy level $\hat{a}$ respects $\hat{a} \sqsubseteq c$ or not. In our approach policies which apply to data are reflected on the results of operations on data and on the activities that perform those operations. Thus, we assign policy levels to activities based on the policy levels of their input data flow.

2.4.5 Contribution Evaluation

The proposed approach has been validated by applying automated tools for sharing analysis of logic program to an appropriate representation of a sample workflow, and interpreting the results of the analysis to construct the resulting lattice of fragment levels.

The sample workflow was related to the health-care scenario, where several activities in patient handling workflow look at different pieces of information, such as insurance records, medication records, and medical history records. Assuming that the organization wants to fragment the workflow by sending its parts to be executed on partner sites, the fragmentation criteria is based on which parts of information can be seen by different fragments to be kept by the host organization or delegated to distributed execution at their partners.

Using "shadow" logic variables, different input data items to an orchestration can be organized into a (complete and finite) lattice of assigned levels. The sharing analysis applied to a representation of the workflow in the form of a logic program produces an abstract substitution that describes possible sharing settings between the variables that represent the input, the intermediate, and the resulting data items, as well as the activities themselves. By interpreting that resulting lattice using a minimal number of new "shadow" variables, a new lattice is constructed with all variables, which preserves ordering from the original lattice. The validation goal was to establish correspondence between the initial and the final lattice.

2.4.6 Conclusions

We have shown how sharing analysis, a powerful program analysis technique, can be effectively used to identify fragments in service workflows. These are in our case represented using a rich notation featuring control and data dependencies between workflow activities, as well as nested structured constructs (such as branches and loops) that include sub-workflows. The policies that are the basis for the fragmentation are represented as points in a complete lattice, and the fragments to which input data / activities belong are stated with initial sharing patterns. The key to this use of sharing analysis is how workflows are represented: in our case we have used Horn clauses designed to adequately enforce sharing between inputs and outputs of the workflow activities. The results of the sharing analysis lead to the construction of a lattice that preserves the ordering of items from the original policy lattice and which contains inferred information which can be used for both deciding the compliance of individual activities with given fragmentation constraints, and to infer characteristics of potential fragments.

As future work, we want to attain a closer correspondence between the abstract workflow descriptions and well-known workflow patterns, as well as provide better support for languages used for workflow specification. Another line of future work concerns aspects of data sharing in stateful service conversations (such as accesses to databases, updates of persistent objects, etc.), as well as on composability of the results of sharing analysis across services involved in cross-domain business processes.

2.5 Runtime Decomposition of QoS Specifications for Distributed Processes

Contributing partner: UniHH

Status: Accepted for the Workshop on Flexible Workflows in Distributed Systems (WiVS), co-located with the conference "Kommunikation in Verteilten Systemen (KiVS 2011)", March 10, 2011.

Keywords: Distributed business processes, quality of service, runtime fragmentation

2.5.1 Background

The specification and enforcement of Quality-of-Service (QoS) requirements are important mechanisms to ensure user-defined goals and preferences before, during or after the execution of services and service compositions. Especially in the context of distributed processes, where parts of such service compositions are outsourced and executed by different devices or organizations, QoS requirements have to be specified in an unambiguous way and have to be communicated to all participating parties in order to guarantee the intended functional and non-functional effects and results of the overall process even in a distributed execution environment.

2.5.2 Problem Statement

Especially in flexible business collaborations, service-based processes often have to be fragmented at runtime so that process parts are assigned and distributed to selected participants in a dynamic way which cannot be foreseen at design time, i.e. when specifying the relevant QoS parameters. It is therefore necessary to also dynamically decompose global QoS requirements which are specified for the execution of the entire process (e.g. the available budget or scheduled time interval) and attach them as respective local QoS requirements to the resulting process parts executed by different partners. As the main problem of this task, most existing approaches for such QoS decomposition and service selection are computationally complex and are thus not suitable to support the distribution of processes at runtime.

Furthermore, they are even less satisfactory for participating modern distributed systems including, e.g. mobile devices.

2.5.3 Contribution Relevance

Based on the observations as introduced above, this contribution presents an approach which:

1. supports process modelers in expressing their non-functional requirements in a way which facilitates runtime decomposition, and

2. derives local QoS specifications from remaining global requirements on process level at runtime by more simple heuristic approaches.

Thus, the presented contribution enhances existing approaches by considering the derivation of QoS specifications for business processes from the viewpoint of distributed process management. It contributes to the derivation of QoS specifications by facilitating the ad-hoc decomposition of non-functional requirements of an entire process into corresponding specifications for process fragments.

2.5.4 Contribution Summary

The presented contribution addresses the problem of QoS specifications for distributed business processes in order to allow the distribution of the process and thus the decomposition of QoS requirements at runtime. Therefore, the contributed research includes the identification and definition of relevant requirements based on an analysis of process execution in dynamic service networks (e.g. mobile ad-hoc networks). As an intermediary result, it was found that in such environments the set of available services may change even during the execution of single business process instances. Due to such dynamism and distribution, often neither the QoS specifications of all services are completely available prior runtime nor it can be guaranteed that discovered services are still available at execution time of single activities within the running process.

In order to nevertheless execute such processes in a user-defined way, an ad-hoc discovery of available services together with their non-functional characteristics (NFCs) and a respective service selection mechanism are proposed. Thus, in contrast to well-established service selection approaches for processes executed in static environments, here services are selected in a step-by-step manner and respective computations for service selection are performed at runtime of each process instance. In order to realize a preferably non-complex selection mechanism, we propose to decrease complexity by three strategies:

- Definition of simple thresholds instead of maximization respectively minimization rules in order to immediately allow for the selection of acceptable solutions.

- Definition of global and local requirements and respective decomposition mechanisms.

- Usage of heuristics instead of optimization techniques in order to further decrease computing time.

The presented language for describing non-functional characteristics and requirements is based on a model which has its foundation in the work of Aagedal [1] as well as of Ardagna and Pernici [3]. The main concepts are (non-functional) *characteristics*, *statements* and *profiles*. The definition of *characteristics* include properties and behavior of a specific non-functional characteristic, such as price. A *statement* restricts the possible range of values for a characteristic. Both service provider and service consumer can define multiple *profiles* in order to provide different service levels, each consisting of multiple statements. Service providers can specify which values they offer, whereas service consumers can provide their requirements. In order to fulfill the identified requirements for process execution in dynamic service networks, this work uses a combination of *WS-Policy* [37] and an enhanced approach based on the *Component QoS Modeling Language (CQML)* (cp. [1]). *Profiles* are realized with the capabilities of WS-Policy. Non-functional *characteristics* and *statements* are expressed in an XML-based language, which uses concepts and ideas of the non-XML language CQML replenished with the missing constructs for composition and priorities.

In order to enable service selection based on dynamic decomposition with respect to such pre-defined statements, the contribution proposes a respective low-complex service selection algorithm. The algorithm consists of three steps: In the first phase, services not satisfying the local requirements are removed. The second phase is a backtracking algorithm using a heuristic to find an efficient solution with respect to the global requirements. In a final step, the detected solution is optimized rudimentarily.

2.5.5 Contribution Evaluation

Both the description language and the algorithm have been successfully tested within the current prototype of the *DEMAC process execution engine* for mobile workflows (cp. [16]) which was formerly using an insufficient key-value-pair approach to specify offers and requirements of non-functional characteristics.

2.5.6 Conclusions

Complementing existing approaches on the specification of quality constraints and non-functional aspects as well as on service selection, this contribution presents an advanced description language to specify non-functional characteristics in a way which supports the usage of strategies in order to decrease complexity as well as a respective low-complex service selection algorithm. Ongoing work includes research on further description mechanisms which enable the integration of application-specific knowledge about the probable execution of the process in order to estimate the control flow with respect to branches and cycles.

2.6 Behavior Monitoring in Self-healing Service-Oriented Systems

Contributing partner: TUW

Status: Accepted as a full paper to the 34th Annual IEEE Computer Software and Applications Conference (COMPSAC), July 19-23, 2010, Seoul, South Korea. IEEE.

Keywords: Self-healing model, monitoring, recovery, mixed service-oriented system, delegation behavior

2.6.1 Background

Web Services and Service-oriented Architecture (SOA) have become the de-facto standard for designing distributed and loosely coupled applications. Many service-based applications demand for a mix of interactions between humans and *Software-Based Services (SBS)*. An example is a process model comprising SBS and services provided by human actors also known as *Human-provided Services (HPS)*. Such applications are difficult to manage due to changing interaction patterns, behavior, and faults resulting from varying conditions in the environment. To address these complexities, we introduce a self-healing approach enabling recovery mechanisms to avoid degraded or stalled systems. The presented work extends the notion of self-healing by considering a mixture of human and service interactions observing their behavior patterns.

2.6.2 Problem Statement

Possible fault sources in Mixed Systems are manifold. In particular, the work focuses on unpredictable and faulty behavior of services in a Mixed System. For that purpose, we observe the behavior of the heterogeneous services and their interactions. We focus on task delegation behavior in a collaborative scenario. Services have a limited buffer for tasks and excessive delegations to single nodes in the network can cause buffer overloads, and furthermore, may lead to service degradation or ultimately to failure. It is thus essential that we identify misbehavior, analyze the cause, and heal the affected services.

2.6.3 Contribution Relevance

Our approach in this work is bottom-up. In this paper, we identify the fundamental delegation behavior models and their effects on the global health of a Mixed System. To our best knowledge this has not been attempted before and is of paramount importance to analyze the system and to settle agreements with guarantees that can be met. The work details the extensible model of the *VieCure* framework. The feedback loop design provided by this framework allows to observe the system's behavior and its boundaries.

Regarding QoS or SLA specifications this information is essential to set-up appropriate agreements with guarantees including reasonable triples of SLOs, including SLO monitoring metrics, the threshold, and the counter-actions for violations. Thus, a dedicated QoS/SLA enforcement module attached to the existing feedback-loop of *VieCure's* modular structure can be integrated. Additionally, the paper's evaluation setup illustrated in Figure 6 can be exploited to not only enforce the agreement's guarantees regarding, e.g., task performance of the HPSs in the system, but also, to simulate and evaluate possible side-effects of (penalty-)actions related to SLO violations in a real environment's copy first. This allows a service provider to better understand and estimate how to negotiate or re-negotiate future agreements before signing contracts and agreeing to enforce the guarantees in his/her real environment.

2.6.4 Contribution Summary

In our work we analyze misbehavior in Mixed Systems with our novel *VieCure* framework comprising an assemble of cooperating self-healing modules. We extract the monitored misbehaviors to models and diagnose them with our self-healing algorithms. The recovery actions of the algorithm heal the identified misbehaviors in non-intrusive manner. A behavior registry allows us to collect general behavior models that emerge during runtime. This supports the task of formulating high level agreements and in particular SLAs as the boundaries of the system become clearer.

2.6.5 Contribution Evaluation

The evaluations conducted include a simulated heterogeneous service environment. The simulated interaction network comprises a node actor framework implemented in JAVA language. The environment consists of a number of nodes with profiles of different behavior models. Each node has a task list with limited capacity. Depending on the deployed behavior model a node tends either to delegate, or process tasks, or exposes a balanced behavior. New tasks are constantly provided. A global timer initiates the simulation rounds. Depending on the behavior model, in each round a node decides to process tasks or delegate one task. However, each task is limited by a ten round expiry. If a task is not processed entirely in this period it is considered a failed task. In the experiments a different distributions of the profiles were shown. The results for the equilibrated environment demonstrate that in such an environment our recovery actions perform almost equal and complete each-other when combined. The success of environments with more notes that delegate or collect tasks depends more on the strategies.

2.6.6 Conclusions

In the work misbehavior of Mixed Systems was analyzed. With the novel VieCure framework comprising an assemble of cooperating self-healing modules we were able to compensate satisfactorily the misbehaviors in a Mixed System (about 30% higher success rate with equal distribution of behavior models). In all but one of the cases, deploying recovery actions supports the overloaded nodes resulting in a higher task processing rate. Important to note, that the failure rate increase near linearly even when recovery actions adjust the nodes' network structure. This observation emphasizes our attempt in implementing non-intrusive self-healing recovery strategies.

The results will help to take further steps towards an accurate policy model. Policies will attached the measured system limits to rules. Furthermore, these policies are the foundation for SLOs that will be negotiated in a future SLA structure.

2.7 Monitoring of SLAs in Service Choreographies

Contributing partner: USTUTT

Status: Submitted to Software Process: Improvement and Practice (SPIP) Journal: Special Issue on Business Process Life-Cycle: Design, Deployment, Operation & Evaluation.

Keywords: Service Level Agreement, SLA Monitoring, Service Choreography, WS-Policy, BPEL4Chor

2.7.1 Background

Service Level Agreements (SLAs) in the context of (Web) services are typically specified based on SLA parameters measured on service operation level (WSDL operation). Thereby, QoS metrics such as availability, response time, and invocation count are measured and agreed upon. There exist already specifications such as WSLA and WS-Agreement which allow modeling and monitoring of SLAs in the WS-* context.

2.7.2 Problem Statement

When services implement business processes via service compositions, it is important to be able to specify and monitor SLAs also on process level, i.e., based on a process model rather than just a (black-box) service interface. For this kind of monitoring, there are several use cases: (i) The service provider might offer *process tracking* information on his public process to the customer as a feature. For example, a shipper may provide shipment tracking information to the customer. In that case, typically only selected public information is exposed, while private process parts are not monitorable. (ii) Service level objectives (SLOs) can be defined based on *process metrics* which may be defined based on monitoring events from different partners (i.e., customer, provider, third parties). In that case, an agreement has to be made on which monitoring information has to be provided by each partner and how to calculate the SLOs. In both cases, there is a need for participants in a service choreography to model their monitoring capabilities and requirements based on public process models and agree on monitored properties.

2.7.3 Contribution Relevance

The presented approach is part of the KPI-related monitoring, analysis, and adaptation topics which are one of the main focuses of the WP-JRA-2.2 work package. The current approach extends the monitoring approaches described in previous deliverables (CD-JRA-2.2.2 and CD-JRA-2.2.4), which were focused on the intra-organizational perspective (service orchestrations), by enabling modeling and monitoring of KPIs and SLOs across business processes in service choreographies.

It also extends the cross-organizational process monitoring approach described in CD-JRA-2.2.4 and [40] which had focused on the monitoring part based on monitoring agreements between choreography participants. In this paper, we focus on how to *derive* such monitoring agreements (which are part of SLAs) from monitoring policies of the participants.

2.7.4 Contribution Summary

The presented approach allows modeling and monitoring of SLAs based on BPEL4Chor service choreography descriptions. Partners in a choreography specify their monitoring requirements and capabilities and corresponding service level objectives (SLOs) using WS-Policy. We provide a monitoring language for modeling of monitored properties on both the process layer (service choreography and orchestration) and the service layer (service interface). Monitored properties include basic properties, such as events which a partner requests or provides to other partners (e.g., a shipper providing location changes of the shipment for tracking purposes), or complex properties such as process metrics (e.g., order fulfillment time, shipment time, etc.). SLOs can be specified on both types of monitored properties. Complex properties are specified using complex event processing technology. The access to monitored properties can be provided both via a push model (events) and a pull model (Web service operations).

After policy creation, the policies of the choreography participants are intersected, thus creating an effective policy which is the basis of a monitoring agreement. The monitoring agreement is deployed on the infrastructure of the participants who then exchange events as specified in the agreement or alternatively use on-demand (pull mode) Web service interfaces to query monitoring information.

2.7.5 Contribution Evaluation

The approach has been implemented by extending the infrastructure already presented in [40] allowing the modeling, querying, intersection and deployment of monitoring policies. It has been evaluated based on a purchase order processing choreography scenario.

2.7.6 Conclusions

In this paper we have presented an approach to SLA monitoring in service choreographies. Each partner in a choreography can specify its monitoring capabilities and requirements using a WS-Policy based language based on an existing BPEL4Chor choreography description. The language supports specifying basic monitored properties for process tracking and composite properties such as metrics based on composite events. The partners create a monitoring agreement which defines the effective policy they have agreed on.

2.8 Runtime Prediction of SLA Violations in Service Compositions

Contributing Partners: TUW, USTUTT

Status: Presented in 3rd Workshop on Non-Functional Properties and SLA Management in Service-Oriented Computing, co-located with ICSOC 2009.

Keywords: Service Composition, SLA Compliance, Prediction, Machine Learning, Event-Based Monitoring

2.8.1 Background

The paper considers providers of composite Web services, which provide coarse-grained value added services on top of existing Web services. For these composite service providers the concept of Service Level Agreements (SLAs) is very important. SLAs are contracts between the provider and their most important customers. They govern the quality that customers can expect from the service. Quality requirements are formulated as a collection of Service Level Objectives (SLOs), which are numerical target values, and penalties for not fulfilling these objectives.

2.8.2 Problem Statement

For service providers, it is essential to prevent SLA violations as much as possible to enhance customer satisfaction and avoid penalty payments. Currently, most research in this area focuses on finding the reason for SLA violations *ex post*, i.e., after the violation has happened. This is certainly useful for later analysis and improving the composition, but it does not directly help preventing violations. That is, the damage has already been done. Therefore, we argue that an *ex ante* approach would be preferable, which is able to *predict* violations at runtime before they happen, while it is still possible to set counteractive measures.

2.8.3 Contribution Relevance

The presented approach extends the monitoring and analysis approach described in the previous deliverable CD-JRA-2.2.2 and [39]. That approach had focused on ex post analysis, while here we enable an ex ante runtime prediction of SLA violations. Thereby, we combine event-based monitoring and machine learning techniques (in particular, artificial neural networks) in order to predict the SLO values of running service composition instances. After automatically predicting the values of SLO metrics at runtime, these predictions could be used in a subsequent step to adapt the service composition instances in order to possibly prevent the predicted SLO violations. This prevention step is presented in the subsequent work described in Section 2.9.

2.8.4 Contribution Summary

The paper proposes an approach for predicting SLA violations in service compositions at runtime. The user specifies check points in the service composition where the prediction should take place, typically before or after a specific activity has been executed. For each checkpoint, a prediction model is created using machine learning regression techniques. The prototype uses artificial neural networks but also other machine learning techniques could be used instead. The prediction model is trained using historical process instances. As input to the prediction model serve nominal and numeric facts which have been measured for the process instance (instance data and QoS measurements). The output is the numerical value of the SLO.

After the prediction model has been trained, it can be used at runtime for prediction of values for running instances. After the execution of an instance is stopped at a checkpoint, first all the measured facts for that instance are gathered. In a second step, estimators can be used to estimate facts, e.g. the response time of services which have yet to be invoked. Both measured and estimated facts are then provided as input to the prediction model which returns the predicted SLO value for that instance.

2.8.5 Contribution Evaluation

The approach has been implemented by extending the Apache ODE BPEL engine and validated experimentally based on an example BPEL process. Experiments show that the time necessary for building the prediction model depends linearly on the number of historical process instances available. The time needed for building the model for 1000 instances is about 30 seconds which seems acceptable for practice. The time necessary for actual prediction is constant and rather small (well below 1 second), which seems very acceptable for prediction at run-time.

2.8.6 Conclusions

The presented approach enables runtime prediction of SLA violations. Main concepts of the approach are predefined check points, which define concrete points in the execution of a service composition at which the prediction should take place, facts, which define the input of the prediction, and estimates, which represent predictions about data which is not yet available in the checkpoint. For the actual prediction, techniques from the area of machine learning are used. The approach is extended in the work described in Section 2.9 towards prevention of SLA violations via adaptation.

2.9 Monitoring, Prediction and Prevention of SLA Violations in Service Compositions

Contributing partners: TUW, USTUTT

Status: Presented in IEEE International Conference on Web Services (ICWS 2010) Industry and Applications Track.

Keywords: Service Composition, SLA Compliance, Prediction, Adaptation, Event-Based Monitoring

2.9.1 Background

The paper considers providers of composite Web services, which provide coarse-grained value added services on top of existing Web services. For these composite service providers the concept of Service Level Agreements (SLAs) is very important. SLAs are contracts between the provider and their most important customers. They govern the quality that customers can expect from the service. Quality requirements are formulated as a collection of Service Level Objectives (SLOs), which are numerical target values, and penalties for not fulfilling these objectives. For the service provider it is therefore vital to minimize cases of SLA violation (i.e., cases where one or more objective could not be fulfilled).

2.9.2 Problem Statement

Currently, most research in this area focuses on finding the reason for SLA violations *ex post* (after the violation has happened). This is certainly useful for later analysis and improving the composition, but it does not directly help preventing violations. That is, the damage has already been done. Therefore, we argue that an *ex ante* approach would be preferable, which is able to predict violations at runtime. Such an approach has been described in Section 2.8.

Besides just predicting the violation, it is desirable to be able to automatically trigger corresponding adaptation actions in order to prevent the violation. We refer to this preemptive approach to SLA management as runtime prevention. Runtime prevention is complementary to the more established *ex post* approaches, in that it helps minimizing SLA violations, even if it cannot substitute offline analysis and optimization of service compositions entirely.

2.9.3 Contribution Relevance

The presented approach extends the runtime prediction approach described in the Section 2.8. The main scientific contribution of this paper is a novel end-to-end approach to automated SLA violation prevention at runtime. We use predictions of SLA violations, which are generated using regression from monitored event data, to trigger adaptations in the service composition. These adaptations aim at improving the performance for the composition instance in such a way that violations are prevented. We argue that our research improves on existing works which consider prediction and prevention only independently. We base the discussions in this paper on service compositions implemented using Microsoft Windows Workflow Foundation technology however, our approach could as well be applied to compositions implemented in different languages, such as WS-BPEL.

Even though the main target of this paper is validating and ensuring compliance with SLAs at runtime, the results discussed here also have some impact on the formulation of SLAs. Specifically, knowledge about the frequency of violations and possible preventative adaptation actions helps derive robust SLAs, which can even be adhered to if things go (slightly) wrong. More concretely, a service provider who has tooling and methods at hand to inform himself about violations and actions which can be triggered to prevent those violations, including the impact that those actions have on SLAs, is better informed about the service level that he can reasonably provide, allowing him to not overcommit to service levels that he has no reasonable means of achieving.

2.9.4 Contribution Summary

In this paper the PREvent (prediction and prevention based on event monitoring) framework is presented. The overall idea of PREvent is to use event-based monitoring of composition runtime data (similar to business activity monitoring) to predict violations of service level objectives. This is implemented using machine learning techniques. We use historical composition data to train multilayer neural networks, which are used as regression classifier to estimate final service level objective values in advance. The prediction part of the framework has been presented in Section 2.8. If we estimate that a specific instance of the composition is likely to run into problems, we can then use the knowledge we have gained through this neural network classifier to trigger adaptations in this instance in a very targeted way (i.e., we can use the neural network to estimate the impact of certain adaptations in advance, and trigger these adaptations that are most helpful for a given problematic instance). PREvent supports a wide variety of different adaptation actions, such as data manipulation in the service composition (e.g., change the parameters passed to some services), service rebinding (use a different base service) or smaller structural adaptations in the composition (such as disabling of isolated activities). More complex structural adaptations have been added in a later publication (see Section 2.8). PREvent builds on the VRESCo service runtime environment to implement these adaptation actions, which is implemented as part of the WP-JRA-2.3 research activity in S-Cube.

Chapter 3

Conclusions

3.1 Summary

This deliverable is concerned with automatic derivation of QoS/SLA specifications for services and service compositions. The contributions presented in it address that problem from several perspectives, and using different approaches, including:

- Derivation by decomposition of QoS specifications for distributed execution of composition fragments based on appropriate expression of the overall QoS specification at design time and run-time heuristics.
- Monitoring misbehaviors in service compositions against nominal QoS/SLA specifications, in order to detect actual QoS bounds and inform correction or renegotiation of SLAs.
- Monitoring, prediction and prevention of SLA violations in service compositions, aimed at establishing reasonable levels of SLA that can be provided.
- Integrating QoS-driven corrective adaptation, based on testing, and perfective adaptation, based on requirements engineering, with reliance on registries of internal and external services.
- Building dynamic models of service orchestrations, coupled with models of service provision infrastructure, to estimate run-time QoS of a collection of running composition, and from there, the average QoS for individual compositions.
- Deriving privacy QoS properties of composition fragments from the overall specifications, and using those properties for assembling service mash-ups from reusable fragments on-demand.
- Creating, monitoring and managing models of Service-Level Objectives (SLOs) for service choreographies.
- Extending the service provision frameworks to allow run-time prediction of possible SLA violation points based on empirically collected data from monitoring and data mining techiques.
- Using sharing analysis as a program analysis technique to derive non-functional properties, such as QoS attributes, of individual activities and fragments inside an orchestration, based on lattices and dataflow inside the orchestration.

These approaches include both the design-time and the run-time aspects of QoS/SLA specifications for service compositions, and while some of them perform a priori synthesis of the specifications, other focus on detecting and predicting violations and lead to corrections in the existing SLAs. A number of contributions specifically address the problem of distributed enactment of service orchestration by means of fragmentation, and aim at deriving QoS properties for such fragments. The monitoring-based approaches include both detection and prediction of SLA violations, coupled with corrective adaptation mechanisms.

The results presented in this deliverable address the following research challenges in WP-JRA-2.2:

- Formal Models and Languages for QoS-Aware Service Compositions;
- Monitoring of Quality Characteristics of Service Orchestrations and Service Choreographies;
- Analysis and Prediction of Quality Characteristics of Service Compositions; and
- QoS Aware Adaptation of Service Compositions.

Several contributions are related to other work-packages, most notably WP-JRA-1.3 and WP-JRA-1.2, and address their corresponding research challenges.

3.2 An Outline of Future Work

The work presented in this deliverable extends the previous work within this workpackage on algorithms and techniques for splitting and merging service compositions (deliverable CD-JRA-2.2.3) and on models, mechanisms and protocols for coordinated service compositions (deliverable CD-JRA-2.2.4). The future work, as outlined in the contributions, can be expected to aim at more robust implementations coupled with more precise detection and prediction techniques, as well as towards development of formalisms for QoS derivation that provide closer correspondence to the actual languages in which service compositions are expressed and their semantics. These results are expected to be presented in the next deliverable CD-JRA-2.2.6, on QoS-aware, coordinated service compositions – mechanisms and techniques.

Bibliography

[1] Jan Aagedal. *Quality of Service Support in Development of Distributed Systems*. PhD thesis, University of Oslo, 2001.

[2] Lianjun An and Jun-Jang Jeng. Web service management using system dynamics. In *ICWS*, pages 347–354. IEEE Computer Society, 2005.

[3] Danilo Ardagna and Barbara Pernici. Global and local qos guarantee in web service selection. In *Business Process Management Workshops*, pages 32–46. Springer, 2006.

[4] Ahmed Awad and Frank Puhlmann. Structural Detection of Deadlocks in Business Process Models. In Witold Abramowicz and Dieter Fensel, editors, *International Conference on Business Information Systems*, volume 7 of *LNBIP*, pages 239–250. Springer, May 2008.

[5] Luciano Baresi, Andrea Maurino, and Stefano Modafferi. Towards distributed bpel orchestrations. *ECEASST*, 3, 2006.

[6] Luciano Baresi, Andrea Maurino, and Stefano Modafferi. Towards Distributed BPEL Orchestrations. *ECEASST*, 3, 2006.

[7] Shang-Wen Cheng, David Garlan, and Bradley Schmerl. Architecture based self adaptation in the presence of multiple objectives. In *SEAMS '06: Proceedings of the 2006 international workshop on Self-adaptation and self-managing systems*, pages 2–8, New York, NY, USA, 2006. ACM.

[8] J. Esparza and C. Lakos, editors. *Applications and Theory of Petri Nets 2002*, volume 2360 of *Lecture Notes in Computer Science*. Springer Verlag, 2002.

[9] B. Ganter and R.Wille. *Formal Concept Analysis*. Springer, 1999.

[10] Rachid Hamadi and Boualem Benatallah. A petri net-based model for web service composition. In *ADC '03: Proceedings of the 14th Australasian database conference*, pages 191–200, Darlinghurst, Australia, Australia, 2003. Australian Computer Society, Inc.

[11] Dragan Ivanovic, Manuel Carro, and Manuel V. Hermenegildo. Automatic fragment identification in workflows based on sharing analysis. In Paul P. Maglio, Mathias Weske, Jian Yang, and Marcelo Fantinato, editors, *ICSOC*, volume 6470 of *Lecture Notes in Computer Science*, pages 350–364, 2010.

[12] D. Jacobs and A. Langen. Static Analysis of Logic Programs for Independent And-Parallelism. *Journal of Logic Programming*, 13(2 and 3):291–314, July 1992.

[13] R. Khalaf and F. Leymann. E role-based decomposition of business processes using bpel. In *Web Services, 2006. ICWS '06. International Conference on*, pages 770 –780, 2006.

[14] R. Khalaf and F. Leymann. E Role-based Decomposition of Business Processes using BPEL. In *IEEE International Conference on Web Services (ICWS'06)*, 2006.

[15] Rania Khalaf. Note on Syntactic Details of Split BPEL-D Business Processes. Technical Report 2007/2, Institut für Architektur von Anwendungssystemen, Universität Stuttgart, Universitätsstrasse 38, 70569 Stuttgart,Germany, July 2007.

[16] Christian P. Kunze, Sonja Zaplata, and Winfried Lamersdorf. Mobile process description and execution. In *Proc. of the 6th Int. Conf. on Distributed Applications and Interoperable Systems (DAIS 2006)*, pages 32–47. Springer, 2006.

[17] Jung Hoon Lee, Young Soon Han, and Chan Hoon Kim. It service management case based simulation analysis and design: Systems dynamics approach. In *Convergence Information Technology, 2007. International Conference on*, pages 1559 –1566, 21-23 2007.

[18] F. Leymann and D. Roller. Business processes in a web services world: A quick overview of bpel4ws, 2005.

[19] J.W. Lloyd. *Foundations of Logic Programming*. Springer, 2nd Ext. Ed., 1987.

[20] K. Marriott and H. Søndergaard. Precise and efficient groundness analysis for logic programs. Technical report 93/7, Univ. of Melbourne, 1993.

[21] Anton Michlmayr, Florian Rosenberg, Philipp Leitner, and Schahram Dustdar. Comprehensive qos monitoring of web services and event-based sla violation detection. In *MWSOC '09: Proceedings of the 4th International Workshop on Middleware for Service Oriented Computing*, pages 1–6, New York, NY, USA, 2009. ACM.

[22] K. Muthukumar and M. Hermenegildo. Combined Determination of Sharing and Freeness of Program Variables Through Abstract Interpretation. In *International Conference on Logic Programming (ICLP 1991)*, pages 49–63. MIT Press, June 1991.

[23] K. Muthukumar and M. Hermenegildo. Compile-time Derivation of Variable Dependency Using Abstract Interpretation. *Journal of Logic Programming*, 13(2/3):315–347, July 1992.

[24] Elena Orta, Mercedes Ruiz, and Miguel Toro. A system dynamics approach to web service capacity management. *Web Services, European Conference on*, 0:109–117, 2009.

[25] S-Cube Partners. Models and mechanisms for coordinated service compositions. Technical report, S-Cube Network of Excellence, December 2008.

[26] S-Cube Partners. State-of-the-art in composition and coordination of service. Technical report, S-Cube Network of Excellence, June 2008.

[27] S-Cube Partners. Algorithms and techniques for splitting and merging service compositions. Technical report, S-Cube Network of Excellence, December 2009.

[28] S-Cube Partners. Models, mechanisms and protocols for coordinated service compositions. Technical report, S-Cube Network of Excellence, August 2010.

[29] Mazeiar Salehie and Ladan Tahvildari. Self adaptive software: Landscape and research challenges. *ACM Transactions on Autonomous and Adaptive Systems*, 4(2):14:1–14:42, 2009.

[30] John D. Sterman. *Business Dynamics: Systems Thinking and Modeling for a Complex World*. Irwin McGraw-Hill, 2000.

[31] Wei Tan and Yushun Fan. Dynamic Workflow Model Fragmentation for Distributed Execution. *Comput. Ind.*, 58(5):381–391, 2007.

[32] Tony Andrews and Francisco Curbera and Hitesh Dholakia and Yaron Goland and Johannes Klein and Frank Leymann and Kevin Liu and Dieter Roller and Doug Smith and Satish Thatte and Ivana Trickovic and Sanjiva Weerawarana. Business Process Execution Language for Web Services, 2003.

[33] I. Trickovic. Modularization and reuse in ws-bpel, 2005. SAP Development Network.

[34] W. van der Aalst. Don't Go With the Flow: Web Services Composition Standards Exposed. *IEEE Intelligent Systems*, Jan/Feb 2003.

[35] W. M. P. van der Aalst and A. H. M. ter Hofstede. Yawl: yet another workflow language. *Inf. Syst.*, 30(4):245–275, 2005.

[36] W.M.P. van der Aalst. Pi calculus versus petri nets: Let us eat "humble pie" rather than further inflate the "pi hype", 2003.

[37] Asir S Vedamuthu, David Orchard, Maryann Hondo, Toufic Boubez, and Prasad Yendluri. Web services policy 1.5 – primer. Technical report, W3C, 2006.

[38] Barbara Weber, Manfred Reichert, and Stefanie Rinderle-Ma. Change Patterns and Change Support Features - Enhancing Flexibility in Process-Aware Information Systems. *Data Knowl. Eng.*, 66(3):438–466, 2008.

[39] B. Wetzstein, P. Leitner, F. Rosenberg, I. Brandic, S. Dustdar, and F. Leymann. Monitoring and Analyzing Influential Factors of Business Process Performance. In *Proceedings of the IEEE International Enterprise Distributed Object Computing Conference (EDOC-09)*, pages 141–150, 2009.

[40] Branimir Wetzstein, Dimka Karastoyanova, Oliver Kopp, Frank Leymann, and Daniel Zwink. Cross-Organizational Process Monitoring based on Service Choreographies. In *Proceedings of the 25th Annual ACM Symposium on Applied Computing (SAC 2010); Sierre, Switzerland, 21-26 March, 2010*. ACM, March 2010.

[41] Ustun Yildiz and Claude Godart. Information Flow Control with Decentralized Service Compositions. In *ICWS*, pages 9–17, 2007.

[42] Haiyan Zhao and Hongxia Tong. A dynamic service composition model based on constraints. In *GCC*, pages 659–662, 2007.