Grant Agreement N° 215483

Title:	*Initial Set of Principles, Techniques and Methodologies for Assuring End-to-end Quality and Monitoring of SLAs*
Authors:	*UniDue, Tilburg, FBK, INRIA, Lero-UL, POLIMI, TUW, UPM, USTUTT*
Editor:	*Michael Parkin (Tilburg), Andreas Metzger (UniDue)*
Reviewers:	*Luciano Baresi (POLIMI), Attila Kertész (SZTAKI), Barbara Pernici (POLIMI)*
Identifier:	*CD-JRA-1.3.4*
Type:	*Contractual Deliverable (CD)*
Version:	*1.2*
Date:	*March 13, 2010*
Status:	*Final*
Class:	*External*

Management summary

The aim of this deliverable is twofold: (1) It provides an updated overview of the research challenges of WP-JRA-1.3 ("End-to-End Quality Provision & SLA Conformance"). (2) It reports on an initial set of principles and techniques for assuring the end-to-end quality and of monitoring SLAs. Work related to these principles and techniques, carried out by S-Cube NoE participants and published in books, journals and conference proceedings, is summarized and assessed with respect to the coverage of the research challenges for this workpackage.

Copyright 2010 by the S-Cube consortium – All rights reserved.

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

Members of the S-Cube consortium:

University of Duisburg-Essen (Coordinator) – UniDue	Germany
Tilburg University – Tilburg	Netherlands
City University London – CITY	U.K.
Consiglio Nazionale delle Ricerche – CNR	Italy
Center for Scientific and Technological Research – FBK	Italy
French Natl Institute for Research in Computer Science and Control – INRIA	France
The Irish Software Engineering Research Centre – Lero	Ireland
Politecnico di Milano – Polimi	Italy
MTA SZTAKI – Computer and Automation Research Institute – SZTAKI	Hungary
Vienna University of Technology – TUW	Austria
Universit Claude Bernard Lyon – UCBL	France
University of Crete – UOC	Greece
Technical University of Madrid – UPM	Spain
University of Stuttgart – USTUTT	Germany
University of Amsterdam – VUA	Netherlands
University of Hamburg – UniHH	Germany

Published S-Cube documents

These documents are all available from the S-Cube Web Portal at `http://www.s-cube-network.eu/`

Foreword

Workpackage JRA-1.3 of S-Cube ("End-to-End Quality Provision & SLA Conformance") has been designed to achieve four long-term objectives:

1. To define principles, techniques and methodologies for *specifying, negotiating and assuring end-to-end quality provision and SLA conformance* with respect to quality characteristics across the functional layers for service infrastructure, service composition and coordination, and business process management, and, across the chain of service providers and consumers. The quality characteristics considered will include, but will not be limited to, performance, dependability, reliability, availability, usability and accessibility.

2. To specify clearly defined interfaces and the interrelationships with respect to end-to-end quality aspects:

   - Between functional layers service infrastructure, service composition and coordination, business process management and the SBA engineering framework, and
   - Between the SBA engineering framework and the SBA monitoring and adaptation framework.

3. To shape the S-Cube convergence knowledge model by providing an integrated set of principles, techniques and methodologies for end-to-end quality assurance and SLA conformance.

4. To provide contributions to IA-3, where the results are integrated into the S-Cube Framework for Service-Based Applications.

As part of these long-term goals, this deliverable provides an updated overview of the research challenges being pursued within WP-JRA-1.3 and reports on an initial set of principles and techniques for assuring end-to-end quality and monitoring SLAs, thereby contributing to the research challenges of the WP.

Acknowledgments: The editors would like to thank the authors of the papers, technical reports, articles and book chapters described in this deliverable for allowing their work to be used in this document.

Contents

Chapter 1

Deliverable Overview

1.1 Introduction

Services are often provisioned within short-term, volatile and highly dynamic (business) processes. These processes are designed in an abstract manner and, when instantiated, can involve service providers which were not known of during the design time of the service-based application. Thus, different from traditional software systems, service-based applications require the composition and coordination of services within highly distributed environments, cutting across the administrative boundaries of various organizations.

To provide the desired end-to-end quality of such globally distributed service-based applications, the dynamic agreement and assurance of service quality becomes an important issue. This requires that not only quality aspects are negotiated and agreed, but also that these are checked during run-time in order to determine whether there is a need for adapting the service-based application or for re-negotiating the quality contracts.

To guarantee the desired end-to-end quality of those service-based applications, contracts between the service providers and the service requestors (also known as service consumers) on quality aspects of services must be established [6]. In general, a contract is a formal agreement between two or more parties to create mutual business relations or legal obligations. Contracts can have different parts, such as the definition of business partners, the specification of functional obligations, and quality, price, and penalties related with the object of the agreement.

Workpackage WP-JRA-1.3 in particular focuses on quality contracts, or more general on those parts of Service Level Agreements (SLAs) which deal with statements about the services quality levels on which the service requestor and the providers have reached an agreement. Other aspects of the contracts, such as parties' identification, legal obligations, or penalties for contract violation, which are also aspects covered in SLAs or general contracts, will not be in the focus of this workpackage.

Based on the general life-cycle of electronic contracts [27] [31], three main activities relevant for quality contracts within service-based applications can be identified:

- *Quality definition*: In electronic contracting, the contract definition activity concerns the establishment of a model or language for the definition of contract terms, which is understood and shared by the contracting parties. This model or language then is used to instantiate an actual contract (e.g., a SLA) that reflects the domain dependent interests of providers and consumers.

- *Quality negotiation*: Establishment of an electronic contract concerns the set of tasks required for defining an actual contract (e.g., SLA) based on the model or language for the definition of contract terms (see above). This may involve the selection of the service provider (the contract partner) among a set of potential providers, the negotiation of the contract terms between the selected provider and the service consumer, and the agreement to the contract terms.

- *Quality assurance*: Contract enactment in electronic contracting concerns tasks for assuring the satisfaction of the contracts. In the case of quality contracts, this implies assuring that the quality levels negotiated and agreed between the service provider and the service requestor are met.

This deliverable specifically focuses on novel contributions for the third activity, namely on quality assurance for service-based applications.

1.2 Deliverable Structure

As a paper-based deliverable, this deliverable contains two parts: (1) This document, which provides the overall motivation and overview of the key research outcomes of the workpackage; (2) The actual research publications that describe the workpackage outcomes in detail and which are summarized in this document.

The document, is structured as follows: In Section 2, this deliverable provides an updated overview of the research challenges being pursued within WP-JRA-1.3. In Section 3, the document – based on research publications of the workpackage members – reports on an initial set of principles and techniques for assuring end-to-end quality and monitoring SLAs, thereby contributing to the research challenges of the workpackage. Section 4 concludes the deliverable and provides an outlook to future work in the workpackage.

Chapter 2

Research Challenges and Contribution to Integrated Research Framework

As described in S-Cube's Description of Work (DoW), the general research goal of workpackage WP-JRA-1.3 is to define novel principles, techniques and methods for defining, negotiating and assuring end-to-end quality across the functional layers as well as across networks of service providers and consumers. This chapter provides an update of the research vision for this workpackage by defining and refining the research challenges addressed in WP-JRA-1.3.

2.1 Key Research Challenges

As part of the activities in IA-3 (Integrated Research Framework), key problems and key research objectives have been identified and refined for workpackage JRA-1.3. The remainder of this section provides the most recent version of those research challenges, which thus (a) allow relating the results presented in this deliverable to the research objectives of the WP (see Sections 2.2 and 3), and (b) provide goals for the future work in the WP (see Section 4).

Figure 2.1 provides and overview of the WP's research challenges in the context of the quality activities (as introduced in Section 1.1). Those research challenges are detailed below:

2.1.1 Challenges In Quality Definition

2.1.1.1 End-to-End Quality Reference Model

Motivation: Different kinds of quality attributes are important in an Service-Based Application (SBA). There is thus a strong need for methods that address quality attributes in a comprehensive and cross-cutting fashion across all layers of a service-based application. Due to the dynamism of the world in which service-based applications operate, techniques are needed to aggregate individual quality levels of the services involved in a service composition in order to determine and thus check the end-to-end quality during run-time. This aggregation will typically span different layers of a service-based application and thus a common understanding of what the different quality attributes mean within and across these layers is needed.

Challenge: To support end-to-end quality provision, S-Cube will aim at making the dependencies between different kinds of quality attributes explicit. For instance, the interrelation between the fulfillment of different QoS attributes across the various layers will be modeled. In addition, S-Cube aims at understanding the dependencies between QoI attributes on the infrastructure layer, the satisfaction of QoE on the service composition layer and the achievement of QoBiz (business value or business KPIs). One key means to achieve the above objective is to achieve a shared understanding of quality attributes between

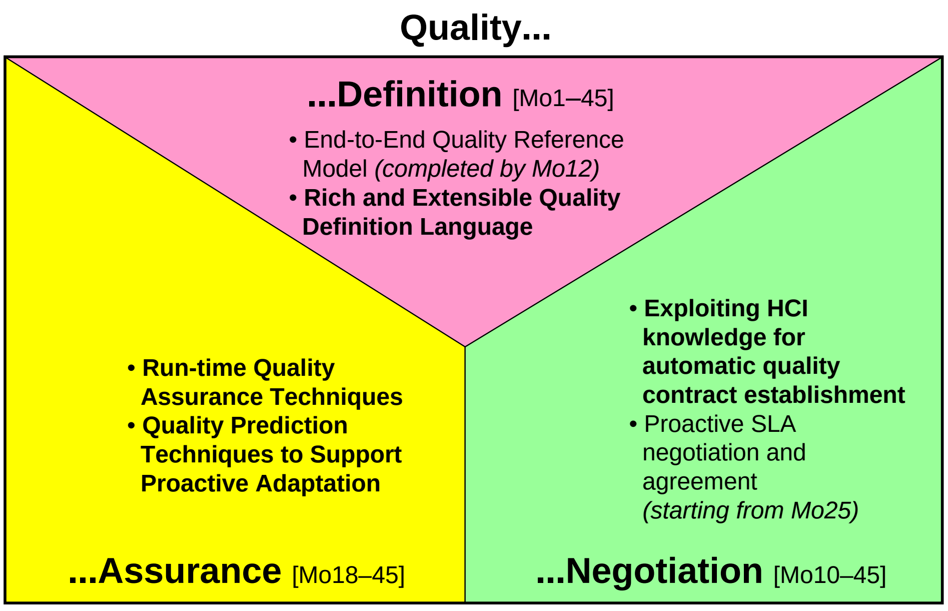
Quality...
...Definition [Mo1–45]
• End-to-End Quality Reference Model *(completed by Mo12)*
• Rich and Extensible Quality Definition Language
• Run-time Quality Assurance Techniques
• Quality Prediction Techniques to Support Proactive Adaptation
...Assurance [Mo18–45]
• Exploiting HCI knowledge for automatic quality contract establishment
• Proactive SLA negotiation and agreement *(starting from Mo25)*
...Negotiation [Mo10–45]

Figure 2.1: Activities relevant for quality of service-based applications and key research objectives of workpackage

the S-Cube layers and disciplines by defining the S-Cube Quality Reference Model. Based on the S-Cube Quality Reference Model and the quality definition language (see Challenge "Rich and Extensible Quality Definition Language" in Section 2.1.1.2 below), foundations for techniques will devised, which allow aggregating individual quality levels of the services involved in a service composition in order to determine and thus ultimately check end-to-end quality.

Note: Work on this challenge constituted the foundations for the work on other challenges of the WP. Those activities have been finalized – as planned – by month 12 of the project.

2.1.1.2 Rich and Extensible Quality Definition Language

Motivation: Concerning quality modeling and definition, this project has observed there is a lack of an established, standardized, rich, extensible and semantically well-defined quality definition language [28]. As a result, quality capabilities and requirements, as well as service SLAs are described by many different formalisms and languages, such as the WSLA language [22], WSML [29], SLAng [19] and RBSLA [26] (amongst many others). Due to this fragmentation, there is still a requirement for a standardized and definition language — necessary for interoperable services.

Challenge: S-Cube is in the process of developing a quality definition language, which allows describing every relevant aspect of quality for services and SBAs, including metrics, units, measurement functions and directives, constraints, value types, etc. In addition, this quality definition language will encompass a rich set of domain-dependent and global quality attributes and will be extensible so as to allow the addition of new quality dimensions when it is needed (e.g., for a application domain which has currently not been considered). As a starting point, the set of quality attributes as defined in the S-Cube Quality Reference Model (see Challenge "End-to-End Quality Reference Model" above) will be exploited. Further, this standard quality definition language will be semantically enriched - where feasible - to be machine-processable or machine-interpretable. This quality definition language will be created to be applicable in complex service-based applications, in which services can be invoked and composed with variable quality profiles. Such a quality definition language should thus be capable of expressing quality capabilities and SLAs by using functions, operators and comparison predicates on quality metrics. It should also allow the description of composition rules for possible combinations of composition constructs and quality metrics.

2.1.2 Challenges In Quality Negotiation

2.1.2.1 Exploiting HCI knowledge for automatic quality contract establishment

Motivation: Service negotiation and agreement involves selecting one out of many service providers based on his quality offer so as to agree on and thus establish the contracts for the delivered service. To address dynamic adaptations of service-based applications, a growing need for automating the negotiation and agreement of quality attributes (e.g., as stipulated by SLAs) can be observed. However, this issue requires considering user interaction and experience (e.g., QoE) issues that may impact on the negotiation itself. This aspect requires a multi-disciplinary effort in which technology researchers will have to interact with researchers addressing user interaction issues.

Challenge: One key research objective regarding quality contract establishment is to exploit user and task models, which codify user preferences and characteristics (see JRA-1.1), in order to devise advanced automated negotiation techniques and protocols. Those advanced techniques could lead to service negotiators (e.g., autonomous components provided as core services) that perform the negotiation process on behalf of the service consumers (requestors) and providers.

2.1.2.2 Proactive SLA negotiation and agreement

Motivation: Similar to proactive (and possibly automated) adaptation (see Challenge "Quality Prediction Techniques to Support Proactive Adaptation" in Section 2.1.3.2), proactive SLA negotiation and agreement is a key prerequisite for effective run-time SLA negotiation since negotiation does not have a negligible computational cost and, therefore, undertaking it when there is an immediate need to use a new service can be unlikely or unfeasible at run-time.

Challenge: The challenge for quality contract negotiation and agreement is how to negotiate the terms and conditions under which a service can be offered before the need for deploying or invoking these services arises. Many of these challenges lie in the definition of negotiation models, and to make the envisioned advances in automated negotiation, we aim to address the limitations introduced above by starting negotiation when there is evidence that the need for deploying a new service and/or change the conditions of deploying a current service is likely to arise but has not arisen yet. Thus, our proactive negotiation approach is based on forecasting at run-time a number of factors related to the deployment of services. Those include, for example, the expected demand for a service, the expected levels of service provision, and the expected service terms and conditions that a service negotiator is likely to agree. The availability of accurate forecasts can lead to effective proactive run-time negotiation strategies for service clients. Prediction also plays a role in quality prediction for proactive adaptation (see Challenge "Quality Prediction Techniques to Support Proactive Adaptation" in Section 2.1.3.2). Although the factors which are relevant differ in both situations, we expect to be able to exploit synergies between the principles and techniques that are developed.

Note: It is expected that work on this challenge can build on the initial results as produced for Challenge "Quality Prediction Techniques to Support Proactive Adaptation" in Section 2.1.3.2. Work on this challenge will thus commence in month 25 of the project.

2.1.3 Challenges in Quality Assurance

2.1.3.1 Run-time Quality Assurance Techniques

Motivation: Given the need for adapting service-based applications at run-time, quality assurance techniques that can be applied at run-time are essential. The major type of run-time quality assurance techniques used today is monitoring, which is often classified as passive (when monitoring relies on actual inbound service consumer traffic to take measurements, so problems can only be discovered after they have occurred) or active (e.g., during run-time testing where the consumer traffic is generated by the testing agent). Monitoring observes the service-based application (or its constituent services) during their current execution, i.e., during their actual use or operation. However, monitoring only allows the assessment of the quality of 'representative' applications (in fact the application in operation) and thus key problems might only be discovered by coincidence. In contrast, standard and consolidated software quality assurance techniques employed during design time, can uncover problems that might only occur after many invocations of the SBA. As an example model analysis can examine classes of executions, thereby leading to more universal statements about the properties of the artifacts.

Challenge: S-Cube will investigate in how standard and consolidated offline software quality assurance techniques can be extended to be applicable while the application operates. For instance, we will investigate into run-time model analysis techniques and other online techniques such as online testing. In addition to extending the quality assurance techniques to the operation phase, synergies between the different classes of quality assurance techniques will be exploited. As an example, we will investigate how testing can be combined with monitoring in such a way that when a deviation is observed during monitoring, dedicated test cases are executed in order to determine - with high confidence - the cause for the deviation. In order to achieve feasible results from run-time quality assurance, it is essential that the artifacts exploited for run-time analysis or testing are a consistent and up-to-date representation (abstraction) of the running service-based application. For example, this leads to the challenge on how to 'synchronize' the model with the SBA in operation in order to achieve valid analysis results. Existing quality assurance techniques appear to be not yet fully incorporated into a comprehensive life-cycle. These aspects are particularly critical as the designers find that understanding what will happen as a result of some self-adaptation design choice quite difficult. Research, jointly with WP-JRA-1.1, will thus address the consistent and comprehensive integration of quality assurance into the service life-cycle (see JRA-1.1).

2.1.3.2 Quality Prediction Techniques to Support Proactive Adaptation

Motivation: To respond in a timely fashion to changes implied by the highly dynamic and flexible contexts of future SBAs and to promptly compensate for deviations in functionality or quality, SBAs have to be able to self-adapt. In current implementations of service-based applications, monitoring events trigger the adaptation of an application. Thus self-adaptation often happens after a change or a deviation has occurred. Yet, such reactive adaptations have several drawbacks, such as: (1) Executing faulty services can lead to unsatisfied users and typically requires the execution of additional activities (e.g., compensation or roll-back); (2) Execution of adaptation activities takes time and thereby can reduce the system performance; (3) It can take time before problems in the system lead to monitoring events (e.g., time needed for the propagation of events from the infrastructure to the business process level), thus events might arrive so late that an adaptation of the system is not possible anymore (e.g., because the system is in a deadlock situation).

Proactive adaptation presents a solution to address these drawbacks, because - ideally - the system will detect the need for adaptation and will self-adapt before a deviation will occur during the actual operation of the service-based application and before such a deviation can lead to the above problems. Key to proactive adaptation is to predict the future quality (and functionality) of a SBA and to proactively respond if the prediction uncovers deviations from expected quality (or functionality).

Challenge: To support the vision of proactive adaptation, S-Cube will work on devising novel quality prediction techniques need. Depending on the kind of quality attribute to be predicted, these can range from ones that built on traditional techniques (see Challenge "Run-time Quality Assurance Techniques" in Section 2.1.3.1) to ones that exploit modern technologies of the Future Internet. As an example for the first case, correctness or performance (QoS) could be predicted by building on techniques similar to online testing or run-time model analysis. As an example for the latter case, usability of services

(QoE) could be predicted by extending existing principles of reputation systems. In this context, one of the possible dimensions to explore is to analyze and predict the properties of networks arising from the interactions between various services. For instance if service A invokes service B, a link between these two services is established. The set of all services and their interactions constitutes a network, which can be represented as a graph structure that can be analyzed by means of traditional link analysis techniques. However, novel and more targeted analysis approaches are needed to support quality prediction.

2.2 Contribution of Deliverable to Key Research Challenges

Summarizing the above challenges, WP-JRA-1.3 will pursue integrative and innovative research to devise novel principles, techniques and methods for defining, negotiating and assuring end-to-end quality for service-based applications.

The aim of this deliverable is to present the first outcomes on the integrated and joint research on quality assurance. More specifically, this document summarizes the key research outcomes in addressing challenges ‘Run-time Quality Assurance Techniques’ (see Section 2.1.3.1) and ‘Quality Prediction Techniques to Support Proactive Adaptation’ (see Section 2.1.3.2). In the remainder of this document, those research outcomes will be presented and related to the WP challenges as well as to the other S-Cube WPs in more detail.

Chapter 3

Initial Principles, Techniques and Methodologies for Quality Assurance

The main research results in the context of this deliverable have been published or submitted as reports, research papers and articles. Thus, in this section we will present (in compact form) the contributions of those papers and how these results relate to the WP research challenges described earlier in Section 2.1.

3.1 Structured Presentation of Results

The following 8-part structure, inspired in parts by "How to Get Your Paper Accepted at OOPSLA" [25], is used to describe each of the reports, papers and articles that form the results of this deliverable.

- *Context and Background:* Initially, the context and background of the problem being addressed in the paper is provided.
- *Problem Statement:* Based on the background, the problem that is addressed (i.e., the research question which is answered) is motivated and explained.
- *Relevance of the Problem and Progress from State of the Art:* The explanation on why the problem is relevant is important to understand why the problem (i.e., research question) is worth pursuing. In addition the relation of the work to the state of the helps understanding the novelty of the contribution and its progress from existing work.
- *Relation to WP Challenges:* The contribution to the WP research challenges is described to understand the contribution of the paper to the overall aims of the deliverable (cf. Section 2.2) and the WP. *Note:* The problem could also (partially) be related to challenges beyond quality assurance, which is the focus of this deliverable.
- *Solution / Research Method:* Either the (innovative) solution (idea) to the problem is stated or or the research method employed (e.g., empirical study) is described.
- *Benefits and Evaluation:* The benefits and utility of the solution when applied to the problem is stated, and, if applicable, it is described how those benefits have been demonstrated by means of an evaluation (method of evaluation and results).
- *Relation to Research Framework:* The solution of the paper is related to the elements of the S-Cube Research Framework, thereby describing the integration achieved across JRA-1 and JRA-2: Monitoring and Adaptation, Engineering and Design, BPM, Service Composition and Coordination, Service Infrastructure.
- *Discussion and Future Work:* Critical discussion on what are the current gaps and shortcomings of the solution and which future research activities are planned. This will allow shaping the future research roadmap for the WP.

3.2 Summary of Research Results

Table 3.1 summarizes the research results (i.e., papers) by categorizing them in relation to their contribution to the six research challenges for this workpackage, which were described earlier in Sections 2.1.1.1–2.1.3.2. As the table shows, the majority of the work presented here concentrates on the quality assurance challenges, including run-time quality assurance techniques and quality prediction techniques to support proactive adaptation. This is in-line with the brief of this deliverable, to define an initial set of principles, techniques and methodologies for assuring the end-to-end quality and monitoring of SLAs.

The table also shows the integration of this work with the other workpackages of the S-Cube research framework. What is interesting is the number of research results also contributing to JRA-2 (the realization mechanisms for service-based systems) and to the research taking place in WP-JRA-2.2 into adaptable coordinated service compositions. Most of the results also contribute to workpackage WP-JRA-1.2, adaptation and monitoring principles, techniques and methodologies for service-based applications.

The principles, techniques and methodologies described in the remainder of this section will be the foundation for upcoming research in this workpackage on the dynamic testing of SBAs and static analysis of SBA properties with respect to end-to-end quality and SLA conformance.

Sections 3.2.6–3.2.4 now provide the descriptions of the research results according to the standard structure described above.

3.2.1 "Embedding Continuous Life-long Verification in Service Life-cycles" [3]

Context and Background: One of the defining properties of Software Service Engineering, as opposed to traditional Software Engineering, is the 'open world assumption', which demands techniques to allow software to react to changes by self-organizing its structure and self-adapting its behavior [1]. The implications of this assumption affect all aspects of Service Engineering, e.g., from the design of a service-oriented architecture to notions of correctness and quality that can be applied to define and check the integrity and the validity of a system design at all stages of the service life-cycle.

As services live in the 'open world', where change is often frequent and unexpected, it becomes important to have the capability to assert properties of a service that have lifelong validity — properties that can be verified as true throughout the design, deployment and execution of the service. This paper examines the consequences of this assumption on how verification is carried out during the service life-cycle and shows how different verification techniques can be applied at different stages of the service life-cycle.

Problem Statement: In the 'open world' context, it is no longer sufficient to apply verification to a service at design time nor to test a service at deployment time to ensure and guarantee it will continue to work as specified or was tested during the remainder of its lifetime. However, existing proposals for service life-cycles advocate that verification is carried out either during the design and implementation or deployment phases, with the possibility of performing verification at both stages (but this does not guarantee properties are conserved across phases). This narrow verification does not address the implications of the 'open world' assumption, such as continuous life-long verification.

Relevance of Problem and Progress from State of the Art: The motivation for this work is based on the observation that current approaches do not address the following:

- Design-time verification only gives limited guarantees;
- There are more phases in the service life-cycle than the design and execution phases;
- Execution-time verification can close the loop of iterative service life-cycles.

This paper attempts to progress the state of the art by addressing these issues with current approaches.

			Quality Definition		Quality Negotiation		Quality Assurance		
Section	Paper Title [reference]		End-to-End Quality Reference Model	Rich & Extensible Quality Definition Language	Exploiting HCI Knowledge for Automatic Contract Establishment	Proactive SLA Negotiation & Agreement	Run-time Quality Assurance Techniques	Quality Prediction Techniques to Support Proactive Adaptation	Also Contributes to Workpackage
3.2.1	Embedding Continuous Life-long Verification in Service Life-cycles	[3]					✓	✓	JRA-1.1, JRA-1.2
3.2.2	A Guided Tour Through SAVVY-WS: A Methodology for Specifying and Validating Web Service Compositions	[2]		✓			✓		JRA-1.1, JRA-1.2, JRA-2.2
3.2.3	Exploiting Assumption-Based Verification for the Adaptation of Service-Based Applications	[10]					✓	✓	JRA-1.2, JRA-2.2
3.2.4	Formal Analysis and Verification of Self-Healing Systems	[13]						✓	JRA-1.1, JRA-1.2
3.2.5	Towards Data-Aware Cost-Driven Adaptation for Service Orchestrations	[17]					✓	✓	JRA-1.2, JRA-2.2
3.2.6	Taming Dynamically Adaptive Systems with Models and Aspects	[24]					✓		JRA-1.1, JRA-2.2
3.2.7	A Framework for Proactive Self-Adaptation of Service-Based Applications Based on Online Testing	[15]					✓	✓	JRA-1.2, JRA-2.2
3.2.8	Online Testing for Proactive Adaptation with High Confidence	[23]					✓	✓	JRA-1.2, JRA-2.2
3.2.9	Runtime Prediction of Service Level Agreement Violations for Composite Services	[20]					✓	✓	JRA-1.2, JRA-2.2
3.2.10	Adaptation of Service-Based Systems based on Requirements Engineering and Online Testing	[11]					✓	✓	JRA-1.1, JRA-1.2, JRA-2.2
3.2.11	An Initial Proposal for Data-Aware Resource Analysis of Orchestrations with Applications to Predictive Monitoring	[16]		✓			✓	✓	JRA-1.2, JRA-2.2
3.2.12	Configuring End-to-End Business Processes using Multidimensional QoS Criteria	[21]				✓	✓	✓	JRA-1.2, JRA-2.2
3.2.13	A CMMI Based Configuration Management Framework to Manage the Quality of Service-Based Applications	[14]					✓		JRA-1.2, JRA-2.2

Table 3.1: Coverage of Research Challenges by Research Results

Relation to WP Challenges: The paper addresses the research challenge of "Quality Prediction Techniques to Support Proactive Adaptation", and partially "Run-time Quality Assurance Techniques" since the prediction occurs during run-time.

Solution / Research Method: This paper proposes that conventional life-cycle models are enhanced with a verification-oriented layer that sits above the conventional models to iteratively integrate and correlate different verification techniques. This layer makes it possible to guarantee the same properties of service contract specifications are still verified even when the service becomes a executable artifact and is embedded in a larger service-oriented architecture. In this manner, feedback provided by the verification activities of the continuous life-long verification can provide a fundamental building block for delivering self-adaptive systems.

Benefits and Evaluation: The enhanced life-cycle models described above give us a life-cycle model that incorporates continuous verification of service properties in order that they may live in the 'open world'. The benefits of adopting such an approach is that it is harmonious with and augments existing software engineering techniques, such as Test Driven Development, which emphasize the role of continuous testing during the development process.

The evaluation of this proposed lifecycle will be through the investigation of challenges of adapting the verification-oriented life-cycle to agile processes and how it can be adopted in the context of real service-based application (SBA) development.

Relation to Research Framework: The approach proposed in this paper provides a method of guaranteeing the behavior and quality of a service through verifying the services properties are constant throughout the its lifecycle — i.e., ensuring the service's advertised functional and non-functional properties are correct. Since this life-cycle also contributes to the other two workpackages in S-Cube's *Engineering & Adaptation Methodologies for Service-based Systems* activities (i.e., JRA-1), namely, Engineering and Monitoring Principles, Techniques & Methodologies for Service-based Systems (WP-JRA-1.1 and WP-JRA-1.2).

Discussion and Future Work: As the paper describes, new engineering methodologies are required to build service-oriented systems living in the 'open world'. This work concentrates on the verification of these systems, showing how existing service life-cycle models are inadequate to support the continuous verification of service behavior (i.e., functions) and quality (e.g., performance) demanded in a constantly changing environment. This paper's approach is to integrate various verification activities into a verification-oriented lifecycle. Future work will include an investigation into how this lifecycle model can be adapted to agile development processes (a likely good fit since both are based on iterations) and the impact on adopting such a model on the quality, duration and cost of engineering new services.

3.2.2 "A Guided Tour Through SAVVY-WS: A Methodology for Specifying and Validating Web Service Compositions" [2]

Context and Background: Software systems, under the form of Service-oriented systems, are evolving from static, closed, and centralized architectures to dynamically evolving distributed and decentralized architectures where components and connections may evolve dynamically. This means we must re-think the application's life-cycle, from development time to run time. Indeed, development time and run time are blurring.

Problem Statement: In open-world service-based systems traditional design-time validation techniques are loosing effectiveness. Applications are built composing services that are designed, deployed, and run by independent parties. Since the scenario is open, new services can appear and disappear, making it difficult to understand the application's true run-time configuration. Design-time validation cannot guarantee the desired levels of run-time quality of service and functionality. Validation must be extended to run time.

Relevance of Problem and Progress from State of the Art: The problem is very relevant to the general goal of providing end-to-end quality. The main progress over the state of the art consists in the proposal of having a holistic approach to the validation of service-based systems. We provide a validation methodology that should guide an application's entire life cycle. In practice we focus on the lifelong verification of service compositions, which means we cover both design-time and run-time verification coherently. The literature offers techniques for specific parts of the problem but, to our knowledge, a comprehensive methodology is still missing.

Relation to WP Challenges: The paper addresses some of the research challenges for WP JRA-1.3. In particular the work contributes to the definition of a "Rich and Extensible Quality Definition Language" and to the definition of a "Run-time Quality Assurance Technique".

Solution / Research Method: The paper presents the SAVVY-WS methodology and its tools. SAVVY-WS adopts ALBERT, an temporal assertion language used to formally specify the design-time and run-time properties that we desire for our BPEL processes. In accordance with our goal of having a holistic approach, it predicates over internal and external variables. The former are information pertaining to the execution space of the BPEL process, while the latter pertain to the surrounding environment and are collected through appropriate data sources. Using ALBERT it is possible to specify a service's required interface in terms of logical formulae called "assumed assertions". Based on the assumed assertions of al services invoked by the BPEL process, the composition may offer a service whose properties can be defined via ALBERT formulae called "guaranteed assertions". Therefore, at design time we perform model-checking to see whether the guaranteed assertions are met, under the assumption that the assumed guarantees are met. However, at run time the assumed guarantees cannot be assumed entirely. Therefore, we need to monitor both the assumed and the guaranteed assertions.

Benefits and Evaluation: The main benefits of this approach are:

1. the holistic nature of the solution,
2. the coherency that exists between design-time and run-time validation, and,
3. the availability of a complete set of tools for both design time and run time. The approach has been evaluated on a number of use cases that are documented in this paper, in which references to other works can also be found.

Relation to Research Framework: With respect to the research framework, the solutions presented in this paper covers elements: "Engineering and Design", "Adaptation and Monitoring", and "Quality Definition, Negotiation, and Assurance".

Discussion and Future Work: It is our intentions to generalize the work and the results achieved in SAVVY-WS, to support other forms of Service-oriented systems that are not based on BPEL. Our goal is to develop SAVVY, a complete methodology for the lifelong validation of dynamically evolving systems. SAVVY will integrate modeling, specification, analysis, and verification techniques, in a technology-agnostic manner, and will be supported by a wide array of tools.

3.2.3 "Exploiting Assumption-Based Verification for the Adaptation of Service-Based Applications" [10]

Context and Background: Service Based Applications (SBAs) are composed of services provided by service providers that are often different from the company that is operating the SBA [8]. In this way the core component of the SBA is not under the control of the SBA owner, as the SBA owner cannot control the provisioning, execution, management and evolution of externally provided services [8]. This means that the SBA designer must rely on the ability of the service providers to meet the expected functionality and quality of those services (encoded, for instance, as service-level agreements).

Once the SBA is put into operation, those expectations may — intentionally or unintentionally — be violated; for instance, a service might fail. The operator of the SBA must not only recognize these violations but also decide whether those violations mean that the overall SBA requirements will no longer be met. In such a case an adaptation of the SBA can become necessary.

Problem Statement: Our approach presented in [9] is able to detect run-time problems and violations of SBA's requirements and to identify specific root causes for those problems in order to determine appropriate adaptation actions.

Relevance of Problem and Progress from State of the Art: Currently, monitoring is used to trigger the adaptation of a service-based application. However, existing monitoring techniques — as detailed below — exhibit several limitations which impact on taking adaptation decisions. Failing to make those decisions may lead to unnecessary or harmful adaptations [18].

Monitoring individual services: It is possible to monitor specific events and elements of the SBA, such as monitoring the constituent services [12]. Such approaches recognize whether a service delivers the expected functionality or quality. However, it is unclear whether this violation of the contract eventually leads to a violation of the SBA's requirements. Without this information we cannot decide whether the SBA should be adapted or not. Assume, for instance, that a service takes $1s$ longer than expected. It may be the case that the service is part of a parallel control flow in the service composition and that such a delay does not have any impact on the performance of the parallel control flow and thus the overall quality of the SBA.

Monitoring service compositions / SBAs: The requirements to the whole service composition may be monitored [4]. In this way it is possible to check whether the composition behaves as required. However, in this case, the identification of the source of the requirements violation is not trivial. Assume, for instance that a service composition takes $30s$ longer than expected to terminate. "Debugging" as an additional step would then be needed to determine, which service(s) caused that delay. It is important to know the cause of the problem to compensate for it by adapting the system; e.g., one could replace the service that caused the delay.

Combined efforts: Even a combination of the above two techniques has limitations. Indeed, in case of complex SBAs, a variety of events and violations may occur. How to debug and identify a specific cause of the requirements violation in order to trigger proper adaptation actions remains an open problem in such a case. Additionally, even with the combined approach we can only identify a problem of the service composition when this is identified by monitoring. This especially means that it is not possible to "predict" whether a violation of a service contract (detected by monitoring individual services) may eventually lead to a violation of the requirements of the service composition.

Relation to WP Challenges: This paper concentrates on "Run-time Quality Assurance Techniques" and "Quality Prediction Techniques to Support Proactive Adaptation".

Solution / Research Method: To achieve this, our approach augments monitoring techniques (to detect service failures) with formal verification techniques (to determine requirements violations). The central idea of our approach is to observe specific properties — assumptions — that (1) are explicitly related to the requirements and (2) characterize the constituent services of the SBA. Thereby, our approach allows (a) verifying whether a problem can lead to a violation of requirements, and (b) tracing the violation to its root cause, which facilitates adaptation.

Benefits and Evaluation: The approach proposed comes with two prospective benefits:

1. The approach supports pro-active adaptation, e.g., it allows issuing an adaptation trigger before the SBA instance terminates.
2. The approach avoids unnecessary adaptations since an adaptation is only triggered if a violation will be violated.
3. Once an adaptation trigger is issued, the root-cause of the problem can be identified, leading to more effective adaptation strategies.

Relation to Research Framework: This work relates to the efforts in quality-definition negotiation and assurance, the operation and management of service-based applications and run-time quality assurance engines.

Discussion and Future Work: In the future we are going to implement our approach using the Astro/Dynamo monitoring framework and standard verification techniques. This implementation is used to validate the feasibility of our approach. Other research questions, which will be addressed in the future include:

1. The extension of the approach to contextual knowledge, e.g., to use the approach for triggering adaptation once the context changes.
2. The formulation of the assumptions in a way that a violation of the assumptions lead to a violation of the requirements and, thus, makes the re-verification step unnecessary.
3. Development of a process, which supports the definition of the assumptions, requirements and the SBA.

3.2.4 "Formal Analysis and Verification of Self-Healing Systems" [13]

Context and Background: The high degree of variability that characterizes Service-based Applications (SBAs) requires to design them with runtime evolution in mind. SBAs should provide mechanisms that autonomously decide how to adapt them at runtime to the internal reconfiguration and optimization requirements or to environment changes and threats. Systems with these characteristics are more general called self-healing or self-adaptive. Self-* systems deals with automatic discovery of their failures, and with techniques to recover from them. Typically the run-time behavior of these systems is monitored to determine whether a change is needed and some adaptation strategies are selected and executed to reconfigure the (part of) system in the desired way. Addition to monitoring, verification techniques could be used to execute online analysis to capture and even predict possible failures and problems.

Problem Statement: Adaptable Service-based systems change their behavior, reconfigure their structure and evolve over time reacting to changes in the operating conditions, so to always meet users' expectations. This is fundamental since those systems live in distributed and mobile devices, such as mobile phones, PDAs, laptops, etc., thus their environment may change frequently. Also, user goals and needs may change dynamically, and systems should adapt their functionalities accordingly, without intervention from technicians. To be able to ensure the expected functionality of these systems, quality analysis and verification techniques are necessary. Moreover, there is a need to define a general framework, abstracting from the runtime systems providing a high-level way to analyze and verify them. Using this framework we should be able to predict possible failures and problems of the running applications.

Relevance of Problem and Progress from State of the Art: Focusing on approaches proposed to model self-* systems, each of them tries to abstract from the runtime systems providing a high-level model useful to analyze, refine and verify them during the development. Other approaches have defined precise languages to provide structural reconfiguration aspects. In the community of Service Oriented Computing (SOC) various approaches supporting self-healing have been defined: triggering repairing strategies as a consequence of a requirement violation, and optimizing QoS of SBAs. Repairing strategies have been specified by means of policies to manage the dynamism of the execution environment or of the context of mobile SBAs. These different proposals are bound to particular language and models. The idea presented in this paper is aimed at understanding the main notions behind such proposals by abstracting away from particular languages and notions. The paper aims to present a uniform formal representation that is abstract enough to cover most of these features.

Relation to WP Challenges: The paper addresses the research challenge of “Quality Prediction Techniques to Support Proactive Adaptation”.

Solution / Research Method: The paper shows how to formally model self-healing systems by using algebraic graph transformations and to prove consistency and operational properties. The main idea is to model this systems with typed graph grammars where the systems rules are divided into three different kinds of rules, namely normal, environment, and repair rules. Normal rules define the normal and ideal behavior of the system. Environment rules model all possible predictable failures. Finally, for each failure a repair rule is defined. This formalization enables the specification, analysis and verification of consistency and operational properties of Self-healing systems. More precisely, the paper presents sufficient static conditions for two alternative self-healing properties, deadlock-freeness and liveness. The formalization of self-healing proposed in this paper enables the formal analysis and verification of modeled properties by using the different kinds of validation features of the AGG tool developed at TU Berlin.

Benefits and Evaluation: The paper in addition to propose a way on how model self-healing systems define a set of interesting properties that them should ensure during the execution. Consistency, deadlock-freeness and cyclic properties have been formalized together with a set of theorem useful to demonstrate that the running system is really self-healing.

The evaluation of the idea proposed in the paper has been done using an example of an automated traffic light system. The aim of the scenario proposed was to ensure suitable self-healing properties by applying repair actions that ensure safe system runs, independently of the unpredictable dynamism of the traffic flow.

Relation to Research Framework: The approach is relevant for the design principles and methods (JRA-1.1), Quality Assurance (JRA-1.3) and Adaptation techniques (JRA-1.2).

Discussion and Future Work: The idea proposed in the paper provides a way to model and analyze self-healing systems using algebraic graph transformation and graph constraints. It distinguishes between consistency properties. including system consistency and normal state consistency, and operational properties, including self-healing, strong self-healing, deadlock-freeness, and strong cyclicity. All properties are formalized and verified for a traffic light system. A future direction is to investigate how far the techniques int this paper for self-healing systems can be used and extended for more general self-adaptive systems as Service-based applications. Another direction should be to extend the way to define adaptation rules defining a way to discover them at run-time and relate them to precise adaptation needs.

3.2.5 "Towards Data-Aware Cost-Driven Adaptation for Service Orchestrations" [17]

Context and Background: Adaptation in SBAs happens at the moment in which a (serious) deviation w.r.t. the expected behavior is detected. A common type of adaptation is to select an alternative service provider which, while giving semantics compatible with the one required, behaves better in other respects (cost, speed, etc.). We tackle the problem of identifying from a set of available services, the most promising services are identified on the basis of the availability of *functions* which describe their expected behavior based on the characteristics of input data (the request that initiates their execution).

Problem Statement: We assume that services (and service compositions) have an observable behavior which depends on the input data: size, composition, etc. The complexity of such behavior (time, number of partner invocations, etc.) is represented with functions over (abstractions of) data in requests. We want to be able to decide dynamically the best partner for a given request (or set of requests) using these functions, and we also want to automatically synthesize and update the description of a composition based on its structure and on the functional description of the partners it interacts with.

Relevance of Problem and Progress from State of the Art: A more informed adaptation can produce compositions which are fitter in the sense that they incur smaller computational cost and communication overhead, and which therefore are able to give a better final QoS. In our case, this extra information comes from taking data characteristics into account and making an offline analysis which synthesizes abstract descriptions of compositions based on the input data, the composition structure, and the descriptions of the partner services. Tackling data as a first-level citizen has long been neglected in the field of SOC, and there are cases where taking it into account would be clearly advantageous.

Relation to WP Challenges: This work is related to:

- *Run-time Quality Assurance Techniques:* A combination of offline processing and run-time decision procedures are used to provide a better QoS. This is clearly shown in the simulations reported in the paper.
- *Quality Prediction Techniques to Support Proactive Adaptation:* although this paper does not address directly proactive adaptation, the functions and abstractions of behavior can be used (as shown in a companion paper) to predict misfits between the expected and the actual execution ahead of time. Also, the finer grain knowledge about the behavior of services can help in making a better decision which may lower the possibility of future mishaps.

Solution / Research Method: Data-aware static analysis is applied to orchestrations, whose abstraction is automatically generated. These abstract descriptions are used in a simulation where an agent looks for partners to meet a minimum QoS. The proposed method is compared with a random selection and a selection based on a static ranking based on average past behavior of services profiles.

Benefits and Evaluation: The simulations have shown large advantages of the data-aware adaptation approach using abstract behavior descriptions, when compared with any of the other two selection methods (fixed and random selection). Therefore, we can evaluate the proposed approach very positively.

Relation to Research Framework: The approach focuses on making predictions of characteristics which have a direct effect on quality. This is also relevant for monitoring and, as shown in this paper, for smarter adaptation, as a means to obtain better overall QoS. Therefore it is relevant for JRA-1.2. As the underlying techniques can also be used to make a more informed decision between different composition possibilities, it can also contribute to JRA-2.2.

Discussion and Future Work: A salient aspect to be worked on is to have more accuracy in the static analysis of data as commonly presented in SBAs (i.e., in XML format). Specific abstraction techniques would have to be devised for this. Tackling in full the complexity of existing orchestration languages (e.g., BPEL or other) remains as future work.

3.2.6 "Taming Dynamically Adaptive Systems with Models and Aspects" [24]

Context and Background: Since software systems need to be continuously available under varying conditions, their ability to evolve at runtime is increasingly seen as one key issue. Modern programming frameworks already provide support for dynamic adaptations. However the high-variability of features in Dynamic Adaptive Systems (DAS) introduces an explosion of possible runtime system configurations (often called modes) and mode transitions.

Designing these configurations and their transitions is tedious and error-prone, making the system feature evolution difficult.

Problem Statement: In this paper, we show how aspects can help designers determine interactions between dynamic variants and how runtime models can be used to validate new configurations on the fly, before committing them on the running system (making it easy to roll-back when a configuration is not valid). Indeed, once the system has been deployed, new variability dimensions and variants that have not been foreseen may appear while the system is running and cannot be stopped. In this case, it is very useful to validate configurations on the fly before actually adapting the running system.

Relevance of Problem and Progress from State of the Art: Ensuring software correctness is an important issue and this is amplified when dealing with software variation. Correctness is even more important in Dynamically adaptive System where variation is handled at runtime. This issue has been addressed by the software product lines community. These contributions mainly introduce formal methods in order to exploit the commonalities of a software family in order to achieve these issues. They rely on *Boolean* or *propositional satisfiability* (or 'SAT') solvers or, more generally, on model-checking techniques in order to verify those tests. In our case, the main difference is the fact that verifications can only be done at runtime. New aspect can be designed and integrated, consequently unanticipated evolution can occur. Using Model Driven Engineering (MDE) [30] techniques allows software developer to apply his aspect on an abstraction of its runtime system to check its correctness.

Relation to WP Challenges: This research addresses the "Run-Time Quality Assurance Techniques" as the models used in adaptation mechanisms are validated at Run-Time using models.

Solution / Research Method: This approach combines aspect-oriented and model-driven techniques in order to limit the number of artifacts needed to realize dynamic variability. Our aspect model weaver allows us to construct configurations on-demand by selecting, by hand or according to pre-defined conditions, a set of aspects. Using the woven configuration, it is possible to validate this configuration before actually adapting the running system. Using aspects instead of low-level reconfiguration scripts allows us to detect some interactions that can provide assistance when selecting the set of aspects to be woven. Then, target configurations obtained after aspect weaving are checked with respect to the invariant we have defined into our meta-model. If a target configuration is not valid, the roll-back mechanism simply consists in not submitting this target configuration to the sub-sequent steps of the adaptation process. If the configuration is valid, we generate the adaptation logic using model comparison. This allows us to automatically determine a safe transition to make the system evolve from a its current configuration to the target configuration. A possible solution to validate target configuration before actually adapting the running system would be to simulate models. This can be done by describing the behavior of each configurations, for example using state machines or Petri nets. Then, it would be possible to use a metaprogramming environment optimized for metamodel engineering, such as Kermeta [5], to execute these models and perform the simulation and detect deadlocks, for example.

Benefits and Evaluation: When the number of configurations is limited, for example in the case of critical embedded systems, it is possible to validate at design time all the possible configurations. However, in larger-scale adaptive systems, this systematic validation may become too time and resource consuming to be realistic. Moreover, once the system has been deployed, new variation points that have not been foreseen may appear while the system is running and cannot be stopped. In this case, it is very useful to validate configurations on the fly before actually adapting the running system.

Relation to Research Framework: This approach tries to solve dynamic adaptations of (service oriented) systems using Models and Aspects to ensure the Quality of Service. This research is linked to the JRA-1.2 and JRA-2.2 WPs for respectively adaptation and monitoring aspects, and end-to-end quality.

Discussion and Future Work: We plan to extend our approach following different axis. Currently, we describe our systems according to their runtime architecture (components, bindings, etc). We will also consider the behavior of dynamically adaptive systems. This can be realized if we can modularize and compose the behavior of components. Thus, it would be possible to decompose the system behavior into aspects, as we do for the architecture. We plan to reuse the approach we have presented in the related work section to consider the behavior. Another axis is about the validation of target configurations. Currently, we ensure that the target configurations ensure the invariants defined in our meta-model. With the definition of the behavior, we would be able to perform simulation in order to detect some deadlocks, for example.

3.2.7 "A Framework for Proactive Self-Adaptation of Service-Based Applications Based on Online Testing" [15]

Context and Background: Service-based applications operate in highly dynamic and flexible contexts of continuously changing business relationships. The speed of adaptations is a key concern in such a dynamic context and thus there is no time for manual adaptations, which can be tedious and slow. Therefore, service-based applications need to be able to self-adapt in order to timely respond to changes in their context or their constituent services, as well as to compensate for deviations in functionality or quality of service. Such adaptations, for example, include changing the workflow (business process), the service composition or the service bindings.

Problem Statement: In current implementations of service-based applications, monitoring events trigger the adaptation of an application. Yet, monitoring only observes changes or deviations after they have occurred. Such a reactive adaptation has several important drawbacks. First, executing faulty services or process fragments may have undesirable consequences, such as loss of money and unsatisfied users. Second, the execution of adaptation activities on the running application instances can considerably increase execution time, and therefore reduce the overall performance of the running application. Third, it might take some time before problems in the service-based application lead to monitoring events that ultimately trigger the required adaptation. Thus, in some cases, the events might arrive so late that an adaptation of the application is not possible anymore, e.g., because the application has already terminated in an inconsistent state.

Proactive adaptation presents a solution to address the above drawbacks, because — ideally — the system will detect the need for adaptation and will self-adapt before a deviation will occur during the actual operation of the service-based application and before such a deviation can lead to the above problems.

Relevance of Problem and Progress from State of the Art: The majority of adaptation approaches resorts to exploiting monitoring techniques and facilities, as they provide a way to collect and report relevant information about the execution and evolution of the application. Depending on the goal of a particular adaptation approach, different kinds of events are monitored and different techniques are used for this purpose. The existing techniques follow the reactive approach to adaptation, i.e., the modification of the application takes place after the critical event happened or a problem occurred, and thus expose several key shortcomings as discussed above.

Furthermore, online testing and regression testing techniques could be exploited to detect changes and deviations before they can lead to undesired consequences and thus support proactive adaptation. Despite the fact that only several authors have highlighted the importance of online testing for service-based applications, there have been no concrete techniques to exploit testing results for (self-) adaptation.

Relation to WP Challenges: The paper addresses the research challenge of “Quality Prediction Techniques to Support Proactive Adaptation” and “Run-time Quality Assurance Techniques”, as the prediction occurs during run-time.

Solution / Research Method: The paper introduces the PROSA framework for PRO-active Self-Adaptation, which defines key activities for enabling the proactive self-adaptation of service-based applications. The novel contribution of PROSA is to exploit online testing techniques in order to anticipate future deviations or changes of a service-based application and thereby to trigger adaptation requests. Online testing means that testing activities are performed during the operation phase of service-based applications (in contrast to offline testing which is done during the design phase). In addition to the definition of those key activities, the paper discusses how those activities can be implemented by building on or extending existing testing and adaptation techniques.

Benefits and Evaluation: In contrast to the “traditional” form of reactive adaptation (e.g., based on monitoring), PROSA provides the following important benefits: First, changes or deviations from expected functionality or quality of service can be uncovered and addressed before they lead to undesirable consequences. Second, the execution of adaptation activities — if done proactively — does not interfere with the execution of the actual application instances, i.e., the users of the application wont be affected by the adaptation. Third, proactive adaptation can provide adaptation requests early enough such that an adaptation of the service-based application still is possible (in contrast to reactive adaptation, where the application can have already terminated in an inconsistent state, for instance).

The benefits and the applicability of the PROSA idea are demonstrated by means of scenarios from a Travel Planning service-based application.

Relation to Research Framework: The approach, from a mechanisms point of view, focuses on the Service Composition and Coordination layer (JRA-2.2), as individual services are monitored and tested to determine failures and deviations. In addition, the approach is relevant for Monitoring and Adaptation (JRA-1.2).

Discussion and Future Work: This paper has introduced the PROSA framework for PRO-active Self-Adaptation. PROSAs novel contribution is to exploit online testing to proactively trigger adaptations. Obviously, an online test can fail; e.g., because a faulty service instance has been invoked during the test. This points to a potential problem that the service-based application might face in the future of its operation; e.g., when the application invokes the faulty service instance. In such a case, PROSA will proactively trigger an adaptation to prevent undesired consequences.

Although exploiting only testing for proactive adaptation provides many benefits, we acknowledge at this stage that further work is required in order to demonstrate the applicability of the PROSA idea in practice. One aspect that, for example, has to be investigated, is the possible impact of the execution of test cases on the performance of the application. Thus, key issues that we will target in our future work are to create a proof-of-concept prototypes based on existing techniques and tools (as discussed in the paper) and to apply these prototypes to realistic cases.

3.2.8 "Online Testing for Proactive Adaptation with High Confidence" [23]

Context and Background: Service-based applications (SBAs) need to operate in a highly dynamic world, where services keep changing and evolving. Since SBAs are composed of individual services, SBAs have to react to failures of those constituent services to ensure that they maintain their expected functionality and quality of service. In such a setting, monitoring is typically used to identify failures of the running instance of a service-based application, which in turn triggers the *reactive adaptation* of the failed SBA instance to compensate for those failures.

Monitoring an SBA can only observe changes or deviations after they have occurred, such a reactive adaptation, has several important drawbacks, such as loss of money or increase of execution time (see Section 3.2.7 for more details).

In the case that more than one SBA instance is running, those drawbacks can be avoided. Assume that a failure has been observed in SBA instance A, then other, running SBA instances B and C could be *pro-actively adapted*. This means that instead of adapting B and C after they have failed, those SBA instances could be modified in such a way that they will be prevented from failing due to the same reason that lead to the failure of A.

Problem Statement: When taking pro-active adaptation decisions it is key there is confidence in the prediction of future failures (i.e., certainty that the prediction is correct). This means that pro-active adaptations should not be initiated based on uncertain predictions of failures for A but should be based on statistically sound evidence. Especially, in the case where a service-based application is built from external (third party) services and thus those constituent services are not under the control of the service composer, the observed quality of service and functionality of those constituent services can vary between different service invocations. As an example, a failure observed at one point in time can disappear at a later point in time, as for instance, a service provider could have repaired the service in the meantime. This means that even if a constituent service fails during the execution of SBA instance A, it might well be the case that the service works as expected when invoked for SBA instances B or C. Thus, if uncertain predictions of failures were used as a basis adaptation decisions, this could lead to unnecessary adaptations, such as a re-selection of service provider, for those SBA instances.

Relevance of Problem and Progress from State of the Art: Unnecessary adaptations can lead to severe consequences: Firstly, they can be costly in terms of time and money, even when pro-active actions are taken that are designed to minimize these factors. For instance, additional activities (such as SLA negotiation for alternate services) might have to be performed or the adaptation can lead to a more costly operation of the SBA (e.g., if a seemingly unreliable service is replaced by a more costly one). Secondly, an adaptation could lead to a fault, e.g., if the new service has significant bugs, leading to severe problems as a consequence. Thus, unnecessary adaptations should be avoided as best as possible.

It should be noted that in the case of reactive adaptations, a failed SBA instance is adapted as soon as a failure has been observed. This is also the case for sporadic failures, as in fact, even such a sporadic failure has lead to a problem for the SBA instance. This also means that reactive adaptations can be dealt with existing techniques and thus in this paper we only focus on novel techniques to enable pro-active adaptation.

Currently, there exist several approaches to predict the quality of software and service systems. Many of them consider statistical techniques during monitoring (e.g., log correlation) or during testing (e.g., statistical testing). However, those approaches face a significant shortcoming when applied to adaptive service-based systems. These techniques usually require a high number of data points (monitoring or test results) to produce statistically sound, and confident data, i.e., if not enough data points are available, the confidence for some of the observed failures might not be guaranteed. This poses significant problems for the applicability of those techniques in the SBA context. If monitoring approaches are applied and only few users have started to use the SBA, the collected monitoring data will only sparingly cover the SBA and thus will not suffice to produce confident results. If testing approaches are applied, achieving the required number of data points can lead to significant, additional costs. This is especially true if the SBA includes external services, and thus invoking those services will be associated with additional costs.

Relation to WP Challenges: The paper addresses the research challenge of “Quality Prediction Techniques to Support Proactive Adaptation” and partially “Run-time Quality Assurance Techniques”, as the prediction occurs during run-time.

Solution / Research Method: The paper proposes to augment monitoring with online testing in order to produce high confidence failure probabilities. With online testing we mean that the SBA is tested (i.e., fed it with dedicated test data / input) in parallel to its normal use and operation (see Section 3.2.7). The online test cases are determined in such a way, that the results provide additional data that complements the data collected through monitoring.

Benefits and Evaluation: The proposed approach is expected to achieve the required number of data points to reach the desired confidence in failure prediction. As we build on the monitoring data available, the testing effort can be kept smaller due to the synergy effects between monitoring and testing. Although the integration of monitoring and testing has been proposed in the literature (e.g., to exploit test cases for monitoring), that novel way of integrating monitoring and testing has not been addressed so far.

The application of our approach has the potential to ensure that all pro-active adaptation decisions can be taken with the desired level of confidence. The failure probabilities can be taken into account during the actual adaptation decisions to determine whether to adapt the SBA or to accept the known risk (i.e., probability) that a failure will occur during the actual operation of the system. In the latter case, the repair of the SBA would still be feasible by following a reactive approach.

The benefits and the applicability of the approach are demonstrated by means of a scenario from the E-Government domain (see WP-IA-2.2).

Relation to Research Framework: The approach, from a mechanisms point of view, focuses on the Service Composition and Coordination layer (JRA-2.2), as individual services are monitored and tested to determine failures and deviations. In addition, the approach is relevant for Monitoring and Adaptation (JRA-1.2).

Discussion and Future Work: The position we took in this paper is that augmenting monitoring data with dedicated test results from online testing promises to produce highly confident failure predictions, based on which pro-active adaptation decisions can be taken with high confidence. In order for this vision of synergetic monitoring and testing to become reality, several important research challenges remain to be addressed, which especially include how to exploit techniques from data mining and statistics to determine the confidence of an existing data set and to compute the additional test cases to reach a desired confidence.

3.2.9 "Runtime Prediction of Service Level Agreement Violations for Composite Services" [20]

Context and Background: In service-oriented computing, service providers and service consumers use service level agreements (SLAs) to agree on QoS properties of services. SLAs contain Service Level Objectives (SLOs), which are numerical target values on metrics such as response time or availability of a service (e.g., maximum response time is 30 seconds).

These metrics are monitored at runtime in order to check SLA conformance. Violations of SLOs negatively impact consumer satisfaction and often result in penalty payments. Therefore, it is very important for the service provider to be aware of SLA violations, in order to react to them accordingly and in a timely fashion.

Problem Statement: Typically, SLA monitoring is done ex post, i.e., metric values and violations of SLOs are reported after they happened. This approach is useful in that it alerts the provider to potential quality problems, however it does not help preventing violations. In that respect, an approach is needed which allows predicting possible SLA violations before they occur. In such a case the service provider could take measures in order to prevent the violation by performing certain adaptations.

Relevance of Problem and Progress from State of the Art: The work presented in this paper can be seen as part of SLA management [7]. SLA management includes the definition, deployment and monitoring of SLAs. In our work, we add another aspect to this, namely the prediction of SLA violations from the provider's point of view before they have actually occurred. We assume that the service has been implemented as a service composition. Thus, prediction demands for some insight into the internal factors that have an impact on service composition performance. The work we have presented in [32] is similar in that it uses machine learning techniques for an analysis of the impact that certain factors have on the performance of service compositions; however it focuses on post-mortem analysis and not prediction. There are similar approaches to SLA prediction as discussed in this paper, which are based on the idea of using prediction models based on machine learning techniques. However, they do not discuss important issues such as the integration of instance and QoS data including estimates of metrics not known at prediction time, strategies for updating prediction models, and architectural and implementation specific aspects such as design of check points.

Relation to WP Challenges: The paper addresses the research challenge of "Quality Prediction Techniques to Support Proactive Adaptation" and partially "Run-time Quality Assurance Techniques", as the prediction occurs during run-time.

Solution / Research Method: The paper presents an approach to runtime prediction of SLA violations for service compositions. Central to the approach are checkpoints, which are defined by the user at certain points in the process where he wants to predict certain SLA metric values (and be able to make adaptations if needed). At each checkpoint, a set of lower level metrics is evaluated which serve as input to the prediction model. Those metrics are:

1. Measured QoS and instance data gathered from process execution so far until the check point is reached,

2. Estimates, which represent predictions about data which is not yet available in the checkpoint, which however can be determined e.g., from history process instances. For creating prediction models, we use techniques from the area of machine learning to construct regression models which are trained using historical process instances. Retraining strategies govern at which times these regression models should be refreshed. Note that the actual run-time prevention of SLA violations is out of scope of this paper and part of future work.

Benefits and Evaluation: The paper presents an architecture of the approach and a prototype Java-based implementation which uses the WEKA Machine Learning framework to build regression models and which is integrated with the Apache ODE BPEL engine. Using an purchase order processing example run on our prototype, we have conducted an experimental evaluation, which shows that our approach is able to predict SLO values accurately, and does so in near-realtime (with an delay of well below 1 second).

Relation to Research Framework: The approach, from a mechanisms point of view, focuses on the Service Composition and Coordination layer (JRA-2.2), as service compositions are monitored. In addition, the approach is relevant for Monitoring and Adaptation (JRA-1.2).

Discussion and Future Work: In future work, the approach will be extended to not only report possible SLA violations to a human user, but to actively try to prevent them. This can be done by leveraging an adaptation framework for service compositions. In that case, additional models will be needed which define adaptation actions and their links to possible SLA violations. In addition, the usability of the prototype will be improved so that checkpoints and facts can be can be defined via a GUI. In particular also facts (metrics definitions) should be as far as possible generated automatically. Finally, the ideas presented in this paper will be generalized so that they are also applicable to aggregated SLOs, such as Average Response Time Per Day.

3.2.10 "Adaptation of Service-Based Systems based on Requirements Engineering and Online Testing" [11]

Context & 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 those applications 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).

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. In fact, coordinating those goals is considered one of the significant challenges in self-adaptive software. As an example, it is important to understand how the need for the corrective adaptation of the SBA must be aligned and synchronized with the opportunity for the perfective adaptation of the SBA.

Relevance of Problem and Progress from State of the Art: If the different adaptation goals are not aligned or synchronized, this can lead to conflicting adaptations, which need to be avoided. As an example, the corrective part could aim at replacing a failed service A for a service B, while at the same time the perfective part could aim at replacing service A with a service C.

This paper introduces a novel way of integrating and aligning perfective and corrective adaptations, while addressing the problems raised by the interactions between those two kinds of adaptation. The framework is based on two techniques for performing perfective and corrective adaptations respectively. The perfective adaptation technique is based on requirements engineering activities to identify new or improved services. The corrective technique is based on online testing, which enables the prediction of potential future failures of the SBA.

Relation to WP Challenges: The paper addresses the research challenge of "Quality Prediction Techniques to Support Proactive Adaptation" as it investigates the interplay of two techniques to support proactive adaptation.

Solution / Research Method: To demonstrate how conflicts between adaptation goals can be avoided, the paper focuses 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:

- *Corrective adaptation based on online testing:* Online tests of services are performed during runtime (operation) of the SBA to determine possible failures of the SBA's constituent services. A failure of such a test constitutes a corrective adaptation trigger, which could possibly be satisfied by replacing the failed service with an alternative service.
- *Perfective adaptation based on RE:* Typically, enterprises have contract relationships with other business partners. This fact is reflected in the set of services that may be used in SBAs. These partner services usually meet the requirements specified by the requirements engineers in the enterprise. In some cases however, due to the dynamic nature of the service market, new relationships are established with other (previously unknown) partners. If the newly introduced service is better and/or more appropriate (e.g., cheaper or faster), the requirements engineer could recommend the use of this new service and thereby issue a perfective adaptation trigger.

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 RE, this provides the possibility to improve the SBAs and to anticipate future requirements.

One key idea 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. Every reference to one of the service is accompanied by a service description. Since the enterprise service registry is a private registry, it can be administrated solely by the enterprise's administrators. On the one hand this enterprise service registry constraints possible adaptation and, therefore, reduces the flexibility of the SBA and, on the other hand, allows us to use techniques (e.g., testing techniques), which require a certain level of stability.

Benefits and Evaluation: The paper has demonstrated 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, it has introduced a possible solution to the problem of how to handle multiple adaptation goals in a service-based application, which has been identified in the literature as one of the key challenges of adaptive software systems.

The application of the integrated approach as well as its building blocks is demonstrated by using a scenario from the telecommunication domain.

Relation to Research Framework: The approach, from a mechanisms point of view, focuses on the Service Composition and Coordination layer (JRA-2.2), as individual services are tested to determine the need for adaptation. In addition, the approach relies on input from requirements engineering (and is thus related to JRA-1.1) and provides input for adaptation (relevant for JRA-1.2).

Discussion and Future Work: In addition to refining and realizing the presented approach for integrating perfective and corrective adaptation, a further, interesting way of extending the presented approach is possible. Instead of merely synchronizing the two adaptation goals, corrective adaptation could also be used to support perfective maintenance. In our setting, online testing could be used to determine the best possible alternative for an adaptation decision *before* the adaptation is executed. This means whenever an adaptation decision is imminent and different alternatives exist, those alternatives could be "pre-tested" and the best one could be chosen.

3.2.11 "An Initial Proposal for Data-Aware Resource Analysis of Orchestrations with Applications to Predictive Monitoring" [16]

Context and Background: Monitoring of service-based applications usually triggers reactive adaptations at the moment in which a (serious) deviation w.r.t. the expected behavior is detected. This has known drawbacks: acting late may bear additional costs, such as excessive resource consumption and the finalization of transactions which could otherwise been avoided or aborted. We tackle the problem of automatically identifying ahead-of-time an abstraction of the expected behavior of an SBA in order to have a finer-grain detection of deviations.

Problem Statement: We approach the problem of predicting deviations in orchestration behavior, based on observable evidence. Unlike other approaches, we do not start from a statistical estimate, but from safe approximations of the expected behavior. These are computed by means of static analysis and updated at run-time to accommodate dynamic variations. In our scheme, the resulting approximations take into account data items handled by some orchestration and their processing. This makes it possible to better predict future indicators such as number of messages sent or received, number of invocations of partner services, size of messages, etc.

The analysis is performed on a representation of the orchestration which takes into account its dataflow and interactions with partner services. This generates a series of equations for the resource at hand which, when solved, provide a *sticky (or program-point) semantics* which is used to infer safe approximations (e.g., guaranteed upper and lower bounds) for the behavior under study and for every point in the orchestration. For points which correspond to states that have not been reached during execution, these approximations give predictions of limits between which the future behavior of the orchestration can possibly evolve. These predictions are adjusted (in general, shifted upwards or downwards) based on the past history and the current execution state of the orchestration. If a prediction of future behavior falls outside the admissible ranges, then an alarm can be raised ahead of time.

Relevance of Problem and Progress from State of the Art: It is clear that predicting service misbehaviors ahead of time is, for the reasons put forward above, always beneficial. In our case, and to the best of our knowledge, very little attention has been paid to predicting the actual behavior of SBAs when data is taken into account. While the universe of data values is in general infinite, safe descriptions of the expected behavior based data abstractions can in general be automatically synthesized. This additional information makes it possible to predict the actual behavior in a way which is closer to the actual one to be observed.

Relation to WP Challenges: This work is intimately related to:

- *Run-time Quality Assurance Techniques:* while the basis of the paper is a static analysis phase (which could, in principle, reveal that no execution of some SBA will possibly meet the expected requirements), we foresee the bulk of the application as a combination of static analysis (to derive descriptions of boundaries for expected behaviors) and run-time techniques, in which these limits are continuously updated and compared with the actual behavior of the system.
- *Quality Prediction Techniques to Support Proactive Adaptation:* Although the work in this paper does not address directly adaptation, but monitoring, the core techniques put forward in it are also applicable to proactive adaptation, as it is explained in a companion paper.
- *Rich and Extensible Quality Definition Language:* a minor, not stressed relationship, exists with the definition of extensible languages, as the properties we deal with are in principle arbitrarily complex (they are expressed in first-order logic) and are finitely expressed using an abstract domain.

Solution / Research Method: Data-aware static analysis is applied to orchestrations, and approximations for behavior are synthesized and used to compare actual and expected execution profiles.

Benefits and Evaluation: While thorough evaluation is pending, we expect to be able to flag future deviations in the execution before they happen, and trigger adaptations which may be able to correct them before it is too late.

Relation to Research Framework: The approach focuses on making predictions of characteristics which have a direct effect on quality, but which are relevant for monitoring (in this case) and adaptation (for a companion paper). Therefore it is relevant for JRA-1.2. The underlying techniques can also be used to make a more informed decision between different composition possibilities. It therefore also contributes to JRA-2.2.

Discussion and Future Work: An issue to be further developed is the accuracy in the static analysis of data as commonly presented in SBAs (i.e., in XML format). Specific abstraction techniques would have to be devised for this. Also, tackling the complexity of existing orchestration languages (e.g., BPEL) could not be done fully, and remains as future work.

3.2.12 "Configuring End-to-End Business Processes using Multidimensional QoS Criteria" [21]

Context and Background: The ability of a service end-user (or client application) to take a generic, abstract business process and configure it according to assured non-functional properties (i.e., with a guaranteed quality of service or QoS) is an important principle of service-based application (SBA) instantiation. The same techniques can be used to adapt badly-performing deployed SBAs to meet an expected performance benchmark.

This paper describes three algorithmic techniques for selecting services *differentiated* by their non-functional properties across multiple service *dimensions* to guarantee an end-to-end QoS for an SBA and their evaluation with respect to performance.

Problem Statement: The QoS parameters of a differentiated service may be for any logical level of the service-based application, from the infrastructure layer to business process. Each of these levels is considered a service dimension in the problem at hand. An abstract description of an SBA can be annotated across these dimensions and the whole process with user-specified values (or even with no value indicating no preference on how a particular step should be carried out). This paper presents an algorithmic solution for selecting the concrete services to meet the *constraints* the desired QoS values place on the abstract process.

Relevance of Problem and Progress from State of the Art: Much work exists in the description and matching the functional properties of services, such as through the use of semantic annotations and reasoning over them. Research into the non-functional properties of services has mainly concentrated on the description of those properties and models for considering the selection of services often only consider a single QoS dimension of a particular process, i.e., they only consider a limited search space. This work seeks to extend the state of the art by allowing QoS selection at different abstract levels of a generic business process and to assess the performance of carrying this out over a large search space.

Relation to WP Challenges: This work supports the work in “proactive SLA negotiation and agreement” challenge, as it can be used as a method of selecting different combinations of services as a negotiation progresses. When a mutually acceptable selection is found, an agreement can be formed. The same techniques can be used to re-negotiate an agreement or re-configure a SBA found not to be performing as desired and can be considered as contributing to the “quality prediction techniques to support proactive adaptation”. It is a general “run-time quality assurance technique”.

Solution / Research Method: The paper proposes the service selection for and configuration of service-based applications can be performed by considering an optimal choice of multi-dimensional QoS variables using the heuristics of simulated annealing and genetic algorithms. The methodology of the work is to use an example SBA and experiment with multiple desired end-to-end QoS and different algorithms to see how they performed, in terms of both accuracy (i.e., the selections difference in end-to-end QoS against what what requested) and time required to complete the selection.

Benefits and Evaluation: The evaluation of the algorithms described in this paper demonstrated that using a generic algorithm gave us a higher accuracy in selecting the services but took much longer than the algorithm based on simulated annealing. A hybrid algorithm that pre-selects services using the simulated annealing algorithm before refining the selection using the genetic algorithm was found to give a slightly higher accuracy than the genetic algorithm but required more time than the genetic algorithm to achieve that result.

Relation to Research Framework: This approach is generic in nature in that it applies equally well to different levels of process granularity, for example, business processes within end-to-end aggregations, component services within individual business processes, or even to activities within component services.

Discussion and Future Work: The results in this paper are preliminary in nature and refinements and extensions are needed in several directions. Initially, we wish to further validate our approach with additional experiments to further calibrate the initial parameters of and the weights attached to the simulated annealing and genetic algorithms in the hybrid algorithm. Following this, we intend to provide integrated support for all constituents of the operational framework we have outlined, and enable the end-user to configure their generic business processes to optimally suit the problem at hand.

3.2.13 “A CMMI Based Configuration Management Framework to Manage the Quality of Service-Based Applications” [14]

Context and Background: Today’s computer world consists of applications which are scattered across different networks and require special effort in terms of integration. Software quality assurance is about identifying the right things to implement and test, and allocating and managing resources in a way that minimizes risks when applications and services are deployed. Organizations have business processes in place in order to meet their objectives; e.g., sales, admin, and financial departments work together in a "Sales process". Each of the units involved in an organizational need one or more services (e.g., application software or utilities). These services run on IT infrastructure which includes both hardware and software, it must be managed accordingly to meet organizational objectives.

Problem Statement: Service-Based Applications (SBAs) have highlighted new challenges related to Configuration Management (CM) which is an important process for the assurance of end to end quality in software systems. As far as the quality of SBAs is concerned, configuration management remains an issue because of the loosely coupled and adaptive nature of the corresponding applications. A smart configuration management approach will allow organizations to make their IT resources more reliable and to utilize them to their maximum.

Relevance of Problem and Progress from State of the Art: In the context of enterprise systems to carry out business functions such as sales, administration and financial service, the departments within an organization work together to fulfill business processes. The inter-departmental nature of such processes requires services (e.g., application software or utilities) are shared between business units. These services run on IT infrastructure which includes both hardware and software, therefore it must be managed accordingly to meet the organization's objectives. Correct management of IT infrastructure will ensure that the required services by business processes are available. CM is part of this IT infrastructure which consists of procedures, policies, and documentation.

Relation to WP Challenges: In service-oriented environments the heterogeneity of resources is dealt by virtue of providing any kind of functionality or resource as a service with a stable interface. This however does not completely remove the need for configuration of the resources, which has to be performed by any service provider of an SBA. The configuration of service implementations and resources must therefore comply with the global requirements of the SBA and must be met at the site of each of the distributed pieces of software. Any kind of local configuration management has implications on the service quality and functional properties, which may not be desired.

Solution / Research Method: We propose a CM framework specifically designed with the adaptation of SBAs in mind. The starting point for this framework is the textbook description of the CM process as outlined by Galin. This process description was designed for traditional software development and has many deficiencies in relation to service-oriented development. Galin's CM process description is comprised of a set of configuration activities and their associated action items. Therefore, in order solve this problem, we use practices from CMMI-SVC to support the action items from CM process. As part of our research methodology, we identify the CM activities and all the corresponding action items in them. For each action item, we find corresponding CMMI-SVC practices within its process areas which can support the implementation of these actions items. This one by one mapping of CM supporting action items with relevant CMMI - SVC process areas' practices gives birth to a CM frame work for SBAs. It is worth considering whether or not to use the CMMI-SVC CM process area directly rather than using parts of it as an add-on to reference CM process. However, since CMMI-SVC is targeted at general services and not specifically at Service-Oriented Computing (SOC) or even software, all of its practices are not directly applicable to SOC. Therefore a suitable methodology was to use the skeleton of a Traditional Software Engineering (TSE) CM process and supplement it with software applicable practices from CMMI-SVC.

Benefits and Evaluation: This research proposes a service-based configuration management framework based on SEI-CMMI-SVC which contributes to the S-Cube project implementing this approach will allow organizations to effectively manage the configurations of their SBAs. The objective of this research is to create a CM framework that can contribute to the end-to-end quality assurance of SBAs. In terms of end to end quality, a CM process would allow developers more accurately develop and update the correct versions of services within SBAs. Service consumers would also benefit from CM as they would see reorganized services as they are updated correctly.

Relation to Research Framework: In the S-Cube life cycle, the Operations and Management phase belongs to both phases. Therefore, it must be efficient and precise enough to meet the transition needs of the entire life cycle. By improving the CM process within operations and Management, we aim to strengthen the entire life cycle of S-Cube. Existing quality assurance techniques appear to be not yet fully incorporated into a comprehensive life-cycle. These aspects are particularly critical as the designers find that understanding what will happen as a result of some self-adaptation design choice quite difficult. Configuration management is a quality assurance technique, and we are incorporating this into the S-Cube life-cycle under Operations & Management.

Discussion and Future Work: Services have made the world more connected; allowing producers, consumers, and other human resources to communicate frequently across the globe. The service industry is a significant driver for the growth of worldwide economy. So guidance on improving service management procedure can serve as a key contributor to the customer satisfaction, performance, and profitability of the business. In this research, we have proposed a CMMI-SVC based framework to manage the configuration of service-based applications. The hypothesis is supported by means of a case study to depict effectiveness of the approach that business can improve its CM process by means of our contribution. A special case with Service Oriented Architecture is that customers don't see the change of services as long as Service Level Agreements are met. Yet, this is not how it is done nowadays and remains a future research issue for us. Another issue is that sometimes the providers of services in an SBA do not coincide with the SBA provider and they may be discovered dynamically during execution. We intend to use configuration information for the purpose of audit and for ensuring compliance between them.

Chapter 4

Conclusions

4.1 Summary

To allow service-based applications to be provisioned within short-term, volatile and highly dynamic (business) processes, their end-to-end non-functional properties, or qualities, should be assured in order that service consumers can be confident the service-based application will be reliable and meet their requirements. This deliverable presents thirteen publications that introduce novel principles, techniques and methods for assuring end-to-end quality for service-based applications, ranging from online testing to perform proactive adaptation to the use of formal models to verify successful adaptations:

- Using comprehensive, coherent design *and* run-time validation processes for service-based systems (see Sections 3.2.1 and 3.2.2).
- A prototype quality assurance system based on the prediction of service behavior from user-defined checkpoints and historic process instances (see Section 3.2.3).
- Providing a formal framework for the quality analysis and verification of services (Section 3.2.4).
- Representing the observable behavior of a service as a set of functions that can be solved to find the best partner for a given request (see Section 3.2.5).
- Allowing the validation of 'dynamic variants' of adaptive systems before adapting the running system (as described in Section 3.2.6);
- Determining the statistical probability of the success of a proactive adaptation (Section 3.2.8)
- Examining the consequences and interplays of more than one adaptation on an SBA (Section 3.2.10).
- Using safe approximations of expected service behavior to predict deviations in the behavior of orchestrations (see Sections 3.2.11);
- Using utility functions based on a users constraints and preferences to allow the direct comparison of services advertising different quality dimensions (Section 3.2.12).
- Treating proactive adaptation as a Configuration Management (CM) problem (see Section 3.2.13);

Together, these principles, techniques and methods represent a first-step at defining S-Cube's integrated view on key activities related to SBA quality. This integrated view will be produced for the upcoming deliverables of this workpackage, namely CD-JRA-1.3.5 ("Integrated principles, techniques and methodologies for specifying end-to-end quality and negotiating SLAs and for assuring end-to-end quality provision and SLA conformance") and CD-JRA-1.3.6 ("Validated principles, techniques and methodologies for specifying end-to-end quality and negotiating SLAs for assuring end-to-end quality provision and SLA conformance"), as well as through the IA-3 Integration Framework.

It should be noted that due to S-Cube being a Network of Excellence, and not an Integrated Project, the papers that constitute this deliverable present solutions to the WP challenges from different angles rather than different "views" on the very same technical solution. More important for S-Cube is the fact that those papers document a significant step towards integration of the different research communities that participate to S-Cube. As an example, the techniques exploit software engineering solutions (such as testing, verification or model-driven development) or techniques from BPM to address problems specific to service-based systems and thus addressed by the service-oriented computing community.

In summary, this deliverable has presented the key research outcomes of the two relevant research challenges of WP-JRA-1.3 concerning quality assurance, which are:

- "Run-time Quality Assurance Techniques" (see Section 2.1.3.1): Here, all but one of the pieces of S-Cube work presented in this deliverable (12 out of 13) address this challenge through the use of different strategies. For example, run-time verification, online testing and models analysis techniques have been published to enable *analytical* quality assurance. The configuration of service-based systems during run-time has also been investigated as a *constructive* quality assurance technique.
- "Quality Prediction Techniques to Support Proactive Adaptation" (see Section 2.1.3.2): This challenge is addressed in 10 of the 13 contributions to this deliverable. Note that most of this work overlaps with the previous challenge, as run-time quality assurance techniques often constitute the foundations for the prediction techniques. Those include online testing techniques to predict failures as well as techniques that predict SLA violations.

4.2 Future Research Activities on Quality Assurance

As can be seen from the summarized papers, in addition to more mature results (such as run-time verification) several of the techniques, despite being very promising, are in their initial stages. It will thus be important to refine and elaborate those techniques as well as to validate those techniques (specifically in collaboration with WP-IA-3.2 "Validation of Integration Framework") to assess their applicability and to demonstrate their benefits for assuring the quality of service-based systems. Also, possible synergies and interrelationships between the techniques will be further assessed (as a first example, the interrelationships between requirements engineering and online testing have been investigated).

Bibliography

[1] Luca Baresi, Elisabetta Di Nitto, and Carlo Ghezzi. Towards open-world software. *IEEE Computer*, 39(10):36–43, 2006.

[2] Domenico Bianculli, Carlo Ghezzi, Paola Spoletini, Luciano Baresi, and Sam Guinea. A Guided Tour through SAVVY-WS: A Methodology for Specifying and Validating Web Service Compositions. In *Advances in Software Engineering: Lipari Summer School 2007*, volume 5316/2008 of *Lecture Notes in Computer Science*, pages 131–160. Springer Berlin / Heidelberg, November 2007.

[3] Dominico Bianculli, Carlo Ghezzi, and Cesare Pautasso. Embedding continuous life-long verification in service life-cycles. In *Proceedings of the 2009 ICSE Workshop on Principles of Engineering Service Oriented Systems (PESOS'09)*, pages 99–102. IEEE Computer Society, 2009.

[4] Fabio Borbon, Paolo Traverso, Marco Pistore, and Michele Trainotti. Run-time monitoring of instances and classes of web service compositions. In *Proceedings of the 2006 IEEE International Conference on Web Services (ICWS 2006)*, pages 63–71. IEEE Computer Society, September 2006.

[5] CNRS, INRIA, ISRIA, and Université Rennes 1. Kermeta workbench: a powerful metaprogramming environment based on an object-oriented DSL (Domain Specific Language) optimized for metamodel engineering. `http://www.kermeta.org/`.

[6] F. Curbera. Components contracts in Service-Oriented architectures. *IEEE Computer*, 40(11):74–80, 2007.

[7] A. Dan, D. Davis, R. Kearney, A. Keller, R. King, D. Kuebler, H. Ludwig, M. Polan, M. Spreitzer, and A. Youssef. Web Services on Demand: WSLA-Driven Automated Management. *IBM Systems Journal*, 43(1):136–158, 2004.

[8] Elisabetta Di Nitto, Carlo Ghezzi, Andreas Metzger, Mike Papazoglou, and Klaus Pohl. A journey to highly dynamic, self-adaptive service-based applications. *Automated Software Engineering*, 15(3-4):313–341, 2008.

[9] Andreas Gehlert, Antonio Bucchiarone, Raman Kazhamiakin, Andreas Metzger, Marco Pistore, and Klaus Pohl. Exploiting assumption-based verification for the adaptation of service-based applications. In *Proceedings of the 25th Annual ACM Symposium on Applied Computing Track on Service Oriented Architectures and Programming*, March 2010.

[10] Andreas Gehlert, Antonio Bucchiarone, Andreas Metzger, and Raman Kazhamiakin. Improving the adaptation of service-based applications by exploiting assumption-based verification techniques. In *Proceedings of the 25th Annual ACM Symposium on Applied Computing Track on Service Oriented Architectures and Programming*, March 2010.

[11] Andreas Gehlert, Andreas Metzger, Dimka Karastoyanova, Raman Kazhamiakin, Klaus Pohl, Frank Leymann, and Marco Pistore. *Service Engineering: European Research Results*, chapter Adaptation of Service-Based Systems based on Requirements Engineering and Online Testing. Springer Berlin Heidelberg, June 2010.

[12] Carlo Ghezzi and Sam Guinea. *Run-Time Monitoring in Service-Oriented Architectures*, chapter Test and Analysis of Web Services, pages 237–264. Springer Berlin Heidelberg, September 2007.

[13] Ehrig Hartmut, Claudia Ermel, Olga Runge, Antonio Bucchiarone, and Patrizio Pelliccione. Formal analysis and verification of self-healing systems. In *Fundamental Approaches to Software Engineering, 13th International Conference, FASE 2010*, 2010.

[14] Sajid Ibrahim Hashmi, Stephen Lane, Dimka Karastoyanova, and Ita Richardson. A CMMI Based Configuration Management Framework to Manage the Quality of Service-Based Applications. Publication TBC, February 2010.

[15] Julia Hielscher, Raman Kazhamiakin, Andreas Metzger, and Marco Pistore. A framework for proactive self-adaptation of service-based applications based on online testing. In *Towards a Service-Based Internet. Proceedings ServiceWave 2008 Conference*. Springer, LNCS, 10-13 December 2008.

[16] Dragan Ivanovic, Manuel Carro, and Manuel Hermenegildo. An initial proposal for data-aware resource analysis of orchestrations with applications to predictive monitoring. In *Proceedings of the 2nd Workshop Workshop on Monitoring Adaptation and Beyond (MONA+ 2009)*, 2009.

[17] Dragan Ivanovic, Manuel Carro, Manuel Hermenegildo, and Pedro Lopez-Garcia Edison Mera. Towards data-aware cost-driven adaptation for service orchestrations. Technical Report CLIP X/2010.Y, Facultad de Informática, Universidad Politécnica de Madrid, November 2009.

[18] Raman Kazhamiakin, Andreas Metzger, and Marco Pistore. Towards correctness assurance in adaptive service-based applications. In *Towards a Service-Based Internet. Proceedings ServiceWave 2008 Conference*. Springer, LNCS, 10-13 December 2008.

[19] D. Davide Lamanna, James Skene, and Wolfgang Emmerich. SLAng: A Language for Defining Service Level Agreements. In *FTDCS 2003: Proceedings of the 9th IEEE International Workshop on Future Trends of Distributed Computing Systems*, pages 100–107. IEEE Computer Society, May 2003.

[20] Philipp Leitner, Branimir Wetzstein, Florian Rosenberg, Anton Michlmayr, Schahram Dustdar, and Frank Leymann. Runtime Prediction of Service Level Agreement Violations for Composite Services. In *3rd Workshop on Non-Functional Properties and SLA Management in Service-Oriented Computing, co- located with ICSOC 2009*, 2009.

[21] Qianhui Liang, Michael Parkin, Mike Papazoglou, and Willem-Jan van den Heuvel. Configuring end-to-end business processes using multi-dimensional qos criteria. Publication TBC, February 2010.

[22] Heiko Ludwig, Alexander Keller, Asit Dan, Richard P. King, and Richard Franck. Web Service Level Agreement (WSLA) Language Specification. Document wsla-2003/01/28, IBM Corporation, 2003.

[23] Andreas Metzger and Osama Sammodi. Online Testing for Pro-active Adaptation with High Confidence. Publication TBC, January 2010.

[24] Brice Morin, Olivier Barais, Grégory Nain, and Jean-Marc Jézéquel. Taming dynamically adaptive systems with models and aspects. In *Proceedings of the 2009 IEEE 31st International Conference on Software Engineering*, pages 122–132, 2009.

[25] OOPSLA Program Committee. How to Get Your Paper Accepted at OOPSLA. In *OOPSLA '91 Proceedings Appendix*, pages 359–363, 1991.

[26] Adrian Paschke. RBSLA: A declarative Rule-based Service Level Agreement Language based on RuleML. In *Proceedings of the International Conference on Computational Intelligence for Modelling, Control and Automation and International Conference on Intelligent Agents, Web Technologies and Internet Commerce Vol-2 (CIMCA-IAWTIC'06)*, volume 2, pages 308–314. IEEE Computer Society, November 2005.

[27] P. Radha Krishna, K. Karlapalem, and D.K.W. Chiu. An ER^{EC} framework for e-contract modeling, enactment, and monitoring. *Data Knowl. Eng.*, 51:31–58, 2004.

[28] S-Cube Partners. Quality Reference Model for SBAs. Deliverable CD-JRA-1.3.2, S-Cube Network of Excellence, March 2008.

[29] Akhil Sahai, Anna Durante, and Vijay Machiraju. Towards Automated SLA Management for Web Services. Report HPL-2001-310R1, HP Laboratories, Palo Alto, California, July 2002.

[30] Douglas C. Schmidt. Model-Driven Engineering. *IEEE Computer*, 39(2):25–31, February 2006.

[31] J. Vonk and P. Grefen. Cross-organizational transaction support for E-services in virtual enterprises. *Distrib. Parallel. Dat.*, 14:137–172, 2003.

[32] Branimir Wetzstein, Philipp Leitner, Florian Rosenberg, Ivona Brandic, Frank Leymann, and Schahram Dustdar. Monitoring and Analyzing Influential Factors of Business Process Performance. In *EDOC'09: Proceedings of the 13th IEEE International Enterprise Distributed Object Computing Conference*, 2009.