Grant Agreement N° 215483

Title:	*Integrated adaptation and monitoring principles, techniques and methodologies across functional SBA layers*
Authors:	*CITY, FBK, POLIMI, SZTAKI, UNIDUE, UCBL*
Editor:	*Gabor Kecskemeti (SZTAKI)*
Reviewers:	*Jean-Louis Pazat (INRIA)* *Harald Psaier (TUW)* *Andreas Metzger (UNIDUE)*
Identifier:	*Deliverable # CD-JRA-1.2.4*
Type:	*Deliverable*
Version:	*1.0*
Date:	*December 14, 2009*
Status:	*Final*
Class:	*External*

Management summary

This deliverable aims to present the research progress of the project partners since the establishment of the baseline cross-layer adaptation and monitoring techniques and methodologies in deliverable PO-JRA-1.2.3. This progress was focusing on the integration of the different monitoring and adaptation approaches applied by the different layers of the service-based applications. The first integration results cover several aspects of the SBA life-cycle. These research results are presented through the summaries of joint papers of the project partners.

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

Members of the S-Cube consortium:

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

Published S-Cube documents

All public S-Cube deliverables are available from the S-Cube Web Portal at the following URL:
http://www.s-cube-network.eu/results/deliverables/

The S-Cube Deliverable Series

Vision and Objectives of S-Cube

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

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

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

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

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

Foreword

This deliverable, CD-JRA-1.2.4 "Integrated adaptation and monitoring principles, techniques and methodologies across functional SBA layers" aims to address the following goals:

- to continue, refine and consolidate the initial results of PO-JRA-1.2.3 towards the initial integration of cross-layer adaptation and monitoring techniques and methodologies.
- to present the initial integration results through the summaries of the joint papers that target the different aspects of the cross-layer adaptation and monitoring problem.
- to identify gaps in the current research domains of the different S-Cube partners and provide a common future direction that leads us towards the vision of the project.
- to provide the foundation for the following JRA-1.2 deliverables that would extend these integrated cross-layer adaptation and monitoring techniques and methodologies. This extension will focus on context awareness, human-computer interaction, predictive monitoring and finally proactive adaptation.

Contents

Chapter 1

Introduction

Monitoring and adaptation are among the most influential aspects of an SBA's life cycle (in the scope of the S-Cube project the life cycle model was defined in CD-IA-3.1.1.). These topics are cross cutting the SBA layers in their very essence. However, as it was identified by the previous deliverables (specifically, Deliverable PO-JRA-1.2.3 "Baseline of Adaptation and Monitoring Principles, Techniques, and Mechanisms across Functional SBA layers" [16]) basically no work has been done outside of any particular layer. Even though the three functional layers of the SBA (Service Infrastructure, Service Composition and Coordination, and Business Process Management) are closely related and are all affected during the execution of the SBAs, the adaptation and monitoring techniques still focus on a single layer. As demonstrated in PO-JRA-1.2.3, there could exist a variety of cases, where focusing on a particular layer of the SBA could cause problems. In particular, (i) the changes in one layer could negatively affect other layers and therefore should be planned carefully; (ii) taking into consideration the information available at multiple layers could lead to better adaptation decisions (or the opposite: not taking into account information from other layers could lead to problematic executions).

Understanding the phenomenon of cross-layer monitoring and adaptation is one of the key problems of this research area. In PO-JRA-1.2.3 we have already started with the collection of requirements, and with the definition of the baseline for cross-layer adaptation and monitoring principles, techniques, and mechanisms. The identified requirements have been classified and exemplified with one of the S-Cube case studies, and have been related to the overall Adaptation and Monitoring framework [11]. This framework has been further refined in order to reflect the elements and mechanisms specific to the problem of cross-layers integration. The presented baseline includes also some preliminary research results achieved by the partners towards cross-layer monitoring and adaptation of SBAs.

The presented document continues, refines and consolidates the initial cross-layer integration results, and present our current progress towards the integrated monitoring and adaptation principles through the research papers jointly written by the S-Cube partners. More specifically, the deliverable presents several results that specifically target different aspects of the cross-layer adaptation and monitoring problem, including novel approaches for the identification of adaptation requirements and strategies across layers driven by functional and non-functional changes, cross-layer models for monitoring and adaptation, infrastructures for engaging and coordinating adaptation actions at different layers. We then map these results on the baseline model of cross-layer adaptation and monitoring framework and identify the gaps and future research directions in this area.

1.1 Relations with the Integrated Research Framework

1.1.1 Contribution to WP Challenges

The scope and the results of the deliverable directly contributes to the research challenges of the WP-JRA-1.2 and the S-Cube Integrated Research Framework. Specifically, this deliverable addresses the

challenge of "**Comprehensive and integrated adaptation and monitoring principles, techniques, and methodologies**". This challenge plans to overcome the isolation and fragmentation of existing adaptation and monitoring (A&M) solutions, the target holistic integrated A&M framework will aim to provide a uniform model of adaptation and monitoring that covers different domains, disciplines, and SBA elements. This framework will accommodate the integration of the existing solutions in different directions:

- Cross-layer adaptation and monitoring, where the problem is addressed for SBA as a whole propagating and exploiting specific actions, mechanisms, and tools at different functional SBA layers.
- Cross-boundary adaptation and monitoring, where the problem is considered across the boundaries of SBAs, addressing the issue of distribution of information, control, and effects to other applications, external systems, and services.
- Cross life-cycle adaptation and monitoring, where the knowledge and models available at different phases of SBA life-cycle (e.g., design-time or post-operational data) is exploited in order to devise new monitoring approaches (e.g., post-mortem analysis for prediction) and adaptation decisions (e.g., to learn from previous decisions and adaptations)

This deliverable aims to provide the initial integration for cross-layer SBA adaptation and monitoring, and, therefore, to present the first unified vision of the S-Cube project on the cross-layer integration of the principles and mechanisms. This vision builds on top of the set of requirements, set of necessary mechanisms and principles, that the previous deliverable defined (PO-JRA-1.2.3). The resulting integration however still shows gaps that are identified in this deliverable to provide input for the upcoming deliverables of the work-package. Building on top of this research, another round of integration approaches will take into account HCI, proactive adaptation and predictive monitoring techniques of the SBAs, this work will result a comprehensive adaptation and monitoring framework to be presented in deliverable CD-JRA-1.2.5.

1.1.2 Relations with other work-packages

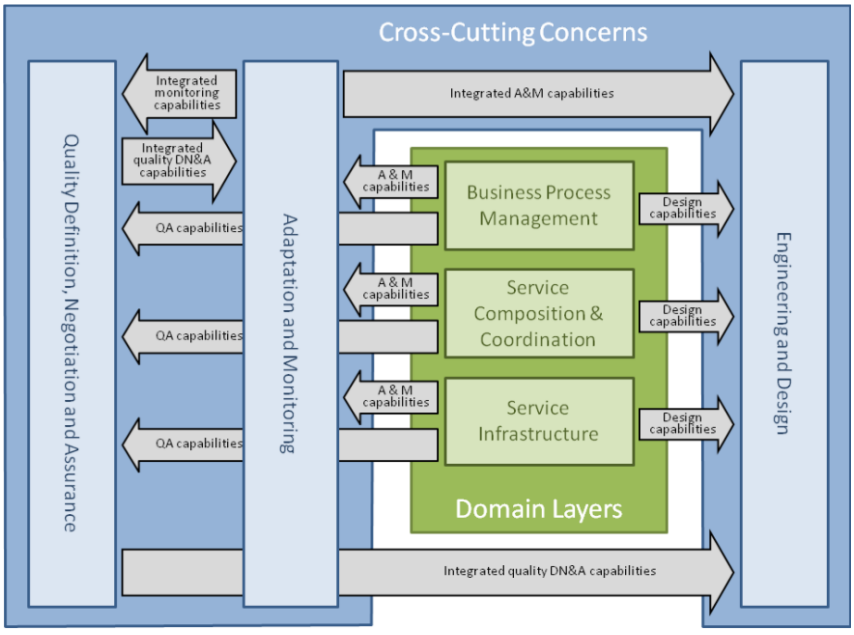

Figure 1.1: Conceptual research framework as defined in CD-IA-3.1.1

As it can be seen on Figure 1.1 the crosscutting nature of the monitoring and adaptation issues result the involvement of several domain layers of an SBA. The aim of this deliverable is the integration of the different monitoring and adaptation approaches already available in the domain layers. This deliverable is based on several joint research papers that already present the connections between the different work-packages shown in the conceptual research framework of CD-IA-3.1.1:

Connections with JRA-1.1 (i) We present the design time requirements through assumption-based verification techniques. With the help of these requirements in section 2.2.4 we identify the monitoring events that can be extracted from the design of the SBA. (ii) Then we propose a language to describe assertions that could raise monitoring events in case of misbehavior of a BPEL process. We also define a language to express the adaptation actions that should take place when the misbehavior is identified. These languages and their use are discussed in section 2.2.5. (iii) Finally in section 2.2.6 we propose that the supervision (including monitoring) of BPEL processes should be deployed within the processes itself.

Connections with JRA-1.3. (i) First in section 2.2.3 we present how quality factor analysis can be embedded to adaptation strategies by associating quality metrics and adaptation actions. (iii) Later on in section 2.2.4 we propose the use of run-time quality assurance and verification in order to define new kind of monitoring events that are based on the results of JRA-1.3. (ii) Then in section 2.2.7 we provide an autonomous service infrastructure that monitors and adapts itself based on service level agreement requirements. (iii) Finally in section 2.2.8 we discuss an approach on monitoring service level agreement violations on the service infrastructure and business process management layers. This approach defines privacy requirements of the users in SLAs, and the compliance of the agreement is checked on different levels of the application.

Connections with JRA-2.1. (i) In section 2.2.1 we show service replacement policies that provide adaptation actions to change or create new requirements on the BPM layer. (ii) In section 2.2.2 We can also monitor behavioral properties of the service-based system, and in case of misbehavior we can execute service discovery to replace the improper service execution during runtime. (iii) Then in section 2.2.3 we also present novel ideas on using decision trees to determine necessary adaptation strategies in case the monitoring system reports KPI violations in business processes.

Connections with JRA-2.2. (i) First, in section 2.2.8 we discuss how monitoring functional and non-functional characteristics of participating services in a service-based system could lead replacing a particular service instance in the SBA. (ii) Second, section 2.2.4 identifies requirements of service replacement and re-composition that include the monitoring of deviations from conversational service protocols and temporal property violations.

Connections with JRA-2.3. (i) Service infrastructure dynamically changes and monitoring the unavailability or the appearance of a service instance could lead to replacing an already used one. During the execution of the SBA finding the new service instances and dropping defunct ones requires a service discovery framework that can be further elaborated jointly with this work-package. This framework is introduced in section 2.2.2. (ii) We have proposed an architecture in the service infrastructure layer that can autonomously adapt to new service environments by tracking the load of brokers and available service instances, this architecture is elaborated in section 2.2.7 (iii) Finally in section 2.2.8 we show that privacy requirements of service users can be met by monitoring and analyzing the possible vulnerabilities on a given service instance.

1.2 Deliverable Structure

This document, deliverable CD-JRA-1.2.4 aims to present our achievements towards the initial integration of the SBA monitoring and adaptation principles, techniques, and mechanisms across functional

SBA layers. Chapter 2 is built around the scientific papers that we wrote in collaboration to research the initial integration of cross-layer adaptation and monitoring. These papers are attached to the deliverable, and therefore this deliverable only provides a structured view on them. This chapter is structured as follows:

- First we summarize the integration baseline requirements identified in deliverable PO-JRA-1.2.3. Based on this summary we propose a template that can be used to analyze the different contributions and their alignment to the previously identified requirements.
- Next in section 2.2 we list the research results of the S-Cube partners. This listing not only provides the customized templates, but it also provides brief descriptions of the research ideas that initiate the integration of monitoring and adaptation principles across SBA layers. We remark that the results presented in this chapter are based on the materials presented in a set of papers that are referred from and attached to the current deliverable only.
- Finally using the proposed template and the research result summaries we provide an analysis of the results. This analysis presents the gaps and the research goals that are still not addressed. With the help of the analysis we also compare the different results and the approaches they took to accomplish the integration among the different SBA layers.

Chapter 2

Cross-layer Adaptation and Monitoring Approaches

The goal of this chapter is to provide an initial integration of the adaptation and monitoring principles, techniques, and mechanisms across functional SBA layers. To achieve this, in Section 2.1 we review the baseline cross-layer adaptation and monitoring framework presented in S-Cube deliverable PO-JRA-1.2.3. Based on the concept of this prospective framework, in Section 2.2 we will present and align the novel approaches developed by the S-Cube partners that aim to address and support the integrated cross-layer adaptation and monitoring. Finally, based on these initial results and solutions, in Section 2.3 chapter will analyze the gaps and missing elements, thus providing requirements and the roadmap for the final integration of adaptation and monitoring principles and mechanisms across functional SBA layers (to be presented in CD-JRA-1.2.5).

2.1 Baseline for Cross-layer Adaptation and Monitoring Principles, Techniques, and Mechanisms

One of the key research challenges in adaptation and monitoring of SBAs is to overcome the limitations of the fragmented and isolated approaches and solutions, providing a comprehensive holistic framework where these approaches are integrated and aligned. In particular, this integration should provide a way to properly locate and evaluate the monitored events across functional SBA layers, as well as to properly identify and propagate adaptation actions across those layers.

The state of art techniques and approaches are not able to deal with this problem. They normally address specific problems peculiar to a particular functional SBA layer, i.e., business process management layer, service composition and coordination layer, and service infrastructure layer. In complex real-scale applications, however, the realization of adaptation and monitoring solutions of the different SBA layers may be highly interleaved. Without appropriate engineering and management these relations may be hidden, which in turn may lead to wrong diagnosis of problems at different layers, incorrect adaptation decisions, and useless or even dangerous modifications (see section 2.1.1 for more details).

To approach this challenge, in S-Cube Deliverable PO-JRA-1.2.3 [16] a baseline for adaptation and monitoring principles, techniques, and methodologies across Functional SBA Layers has been defined. This baseline has described the relevant adaptation and monitoring characteristics of service-based applications, identified a set of key problems that require cross-layer principles and mechanisms, and, finally, defines a prospective framework aiming to accommodate those principles and mechanisms. The results of this work have been also presented in [14]. In the following we will briefly summarize these aspects to provide a basis for the initial integration and alignment of the novel principles and approaches developed by the S-Cube partners.

2.1.1 Requirements

When the problem of SBA monitoring and adaptation is addressed in isolation at different functional layers, a set of problems may arise. In [14, 16] the following problems have been identified:

- *Lack of alignment of monitored events*. If the monitoring is done by isolated mechanisms at different layers, it is possible that the same situation or an event is seen by those mechanisms independently and in different way, potentially triggering independent and even contradictory adaptations. There is a need to properly align, and correlate information among different layers in order to achieve a consistent picture and enable correct and coordinated adaptation activities.
- *Lack of adaptation effectiveness*. The adaptation activities initiated at one functional layer may fail to achieve the expected effect, since they do not take into account the properties of other layers. For example, at the SCC layer in order to reduce execution time the adaptation aims to execute independent tasks in parallel by delegating them to different services. In order to achieve this effect it is necessary that those services are independent also at the SI layer.
- *Lack of compatibility*. This problem refers to a situation, where the adaptation performed at one functional SBA layer is not compatible with the requirements and constraints posed by the application design at other layers.
- *Lack of integrity*. A problem of adaptation integrity happens in a situation, where the adaptation performed in one layer is insufficient; the activities should also be propagated to other layers as well. For instance, when the adaptation is performed at the BPM layer (e.g., a business process is changed), it is necessary to propagate the changes also to the other layers (e.g., change the compositions that manage corresponding process instances and/or perform some compensation actions; negotiate, bind, and adjust to the new formats and protocols at the SI layer, etc.).

To address these problems, the cross-layer adaptation and monitoring approaches should satisfy the following requirements:

- provide means to propagate the monitored information across the layers in order to properly diagnose the actual source of the problem;
- align and correlate the monitored events across functional layer in order to avoid spontaneous and uncoordinated adaptation activities at different layers;
- take into account features and relations across the whole SBA stack when the adaptation requirements are defined and the adaptation strategy is identified;
- define certain "boundaries" for each of the layers and check that the adaptation activities of other layers do not cross those boundaries;
- identify, aggregate, and enact wide range of adaptation actions available at different functional layers in a coordinated manner.

2.1.2 Prospective Cross-layer Adaptation and Monitoring Framework

The prospective cross-layer adaptation and monitoring framework defined in [16] aims to accommodate and address the requirements identified in 2.1.1. We remark that this framework does not provide a concrete solution to the problem of cross-layer integration, but suggests a general vision on the adaptation and monitoring problem from the cross-layer integration perspective. In particular, it shows how the requirements are positioned in the general A&M framework (described in [11]) and what kind of mechanisms and principles are needed to achieve them.

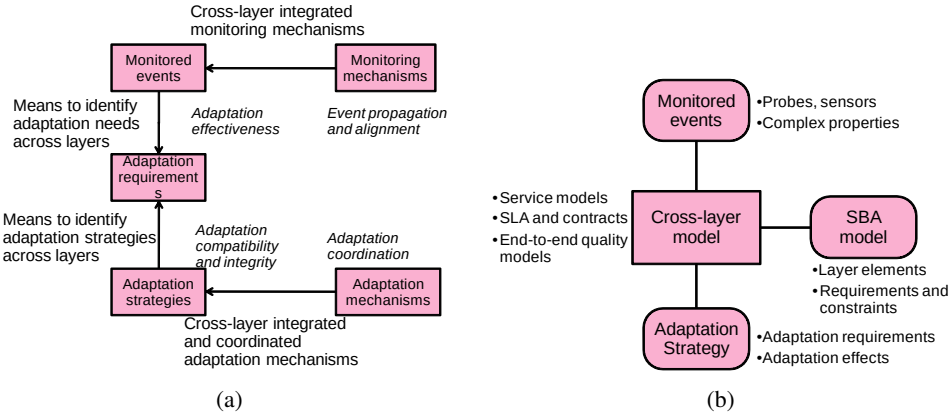

Figure 2.1: Cross-layer adaptation and monitoring framework

Required cross-layer adaptation and monitoring mechanisms.

Figure 2.1(a) relates the general A&M framework [11] with the problem of cross-layer adaptation and monitoring. In particular, the elements of the conceptual model of the general framework, i.e., Monitoring mechanisms, Monitoring events, Adaptation requirements, Adaptation strategies, and Adaptation mechanisms, are explicitly related to the identified requirements for cross-layer adaptation and monitoring and to the corresponding mechanisms to realize those requirements. This includes, in particular

- *Cross-layer integrated monitoring mechanisms* built on top of the existing and new monitoring capabilities should provide a holistic, integrated infrastructure for the SBA monitoring. In particular, such infrastructure should allow for (i) expressing layer-specific events and properties in a uniform manner, and (ii) relate events of different layers to each other to enable their correlation, aggregation and alignment.
- *Cross-layer integrated and coordinated adaptation mechanisms* ensure that the adaptation activities being triggered at different layers or across those layers are properly managed, i.e., control, execute, and coordinate them.
- *Means to identify adaptation needs across layers* aim to properly reveal the adaptation requirements, when the changes concern not a single layer, but the whole application. That is, these mechanisms should provide ways to (i) properly identify the source of the problem and the corresponding requirements and (ii) map those requirements onto the relevant functional layers.
- *Means to identify adaptation strategies across layers* aim to (i) validate the available adaptation actions and strategies against the whole application model; (ii) foresee whether the adaptation strategies are sufficient to achieve the corresponding requirements; (iii) expand, join, and align various adaptation strategies at different layers when the single strategies are insufficient or require some actions at other layers.

Required models for cross-layer adaptation and monitoring.

To enable the above mechanisms, it is crucial to explicitly relate different conceptual elements to each other and across different layers. To accomplish that, there is a need for high-level unified models, which would relate specific elements of the application, specific associated adaptation strategies and mechanisms, and specific monitoring events and mechanisms available at different layers and peculiar to different approaches. The novel cross-layer solutions will rely on such models in order to propagate

and aligne monitored events, to reason about cross-layer adaptation requirements and strategies, their impacts, and dependencies, and to control and coordinate specific adaptation mechanisms.

As it is shown in Figure 2.1(b), these unified model should include,

- *cross-layer representation of monitored events* where cross-cutting aspects of the SBA capture relevant information and its sources to be observed at a particular layer and abstract from the low-level realization and specification details;
- *cross-layer representation of adaptation strategies* enables the analysis of adaptation integrity, effectiveness, and compatibility, as the adaptation actions and strategies at different layers are characterized in terms of the relevant cross-layer models;
- *cross-layer representation of the application model* is essential to reflect the relations and the impacts of the adaptation activities on the different layers of the application architecture.

The approaches and results developed in the scope of the WP-JRA-1.2 and presented in this document, provide a wide spectrum of solutions for the cross-layer unified representation of the application (e.g., model of assumptions for the SBA context in Section 2.2.4 or non-functional SLAs in Section 2.2.8), of the monitored events (e.g., values of quality factors in Section 2.2.3 or SLA violations in Section 2.2.7), and of the adaptation strategies (e.g., languages and policies described in the approaches of Section 2.2.1 or 2.2.5). In the section 2.2 we will show in details how these approaches contribute to the vision and what are the gaps still to be addressed and elaborated in this direction.

2.1.3 Template for presenting the results

In order to provide a syntactic overview of the contributions of the partners and to relate them to the baseline for the cross-layer adaptation and monitoring principles, techniques, and methodologies presented in PO-JRA-1.2.3, a synthetic template is proposed (Table 2.1). The template is necessary for the identification of the gaps in the integrated adaptation and monitoring framework developed in the scope of the WP-JRA-1.2 and therefore will provide an input for the upcoming activities in the holistic integrated cross-layer adaptation and monitoring PTMs to be presented in deliverable CD-JRA-1.2.5.

The template consists of four building blocks. First, it describes how the presented approach contributes to the required cross-layer mechanisms and tools presented in PO-JRA-1.2.3 (here represented in Figure 2.1(a) on page 9), namely integrated monitoring mechanisms, integrated and coordinated adaptation mechanisms, means to identify adaptation needs across layers, and means to identify adaptation strategies across layers. When applies, this description show how the presented approach refines those mechanisms.

Second, the template characterizes the specific cross-layer model selected by the approach. This characterization contains three types of information. The cross-layer aspect addressed in the model may speak of generic information (i.e., the model may be used for a wide range of domains, scenarios, and concerns) or may refer to a specific concern of the application and even in a specific domain (i.e., privacy, reputation, behavioral properties). Component of the model refers to the SBA itself, to the adaptation actions and/or to monitoring events and properties as described in Figure 2.1(b) on page 9. Key model elements describe the specific feature of the model applied by the approach in order to represent the application aspect at different functional layers.

Third, the approach is characterized in terms of the elements of the functional layers it covers and addresses. These elements are presented and described in PO-JRA-1.2.3. More specifically, the monitoring events and/or the adaptation actions of the three functional layers are defined. This is specifically important to understand the coverage of the service-based application components by the initial integration aimed in this deliverable.

Finally, the template is used to characterize potential extensions of the proposed approach across layers towards inclusion of more mechanisms, models (i.e., more aspects and features), and layer elements (within the same layer and/or at different layers). This description is essential for identifying the

future research activities to be performed in order to include the approach in the holistic adaptation and monitoring framework studied in the work package.

Title	the title of the research work
Authors	Authors of the contribution
Short description	brief description of the key aspects of the research work with respect to the cross-layer adaptation and monitoring

Target cross-layer mechanisms

Integrated monitoring mechanisms	Specific integrated monitoring mechanisms (if any)
Integrated and coordinated adaptation mechanisms	Specific integrated adaptation mechanisms (if any)
Means to identify adaptation needs across layers	Specific means to identify adaptation requirements (if any)
Means to identify adaptation strategies across layers	Specific means to identify, select, and integrate adaptation actions (if any)

Cross-layer model

Aspect	Specific type(s) of properties, activities, and domains addressed by the approach
Component	Specify and refine the specific part of the cross-layer representation of the application, adaptation and monitoring
Key model elements	Specify the key modeling elements of the approach

Possible extensions

Cross-layer mechanisms	Possible extensions of the approach towards cross-layer mechanisms
Cross-layer model	Possible extensions of the approach towards modeling
Layer elements	Possible extensions of the approach towards layer elements to be considered

Covered elements at functional layer

	Monitoring Events	**Adaptation Actions**
Business Process Management		
Service Composition		
Service Infrastructure		

Table 2.1: Contribution overview template

2.2 Summary of the individual contributions to the integrated cross-layer adaptation and monitoring principles

2.2.1 Replacement Policies for Service-Based Systems

Title	Replacement Policies for Service-Based Systems ([23])
Authors	Khaled Mahbub and Andrea Zisman
Short description	This work presents a set of policies for dynamic adaptation of service based application where the adaptation may be triggered due to various situations in different SBA layers including unavailability or malfunctioning of services; changes in the functional, quality, or contextual characteristics of the services; changes in the context of the service-based system environment; emergence of new services; or changes in the requirements of the system. A prototype tool incorporating the replacement policies and the deployment of the changes in service-based systems using proxy services has been implemented in order to illustrate and evaluate the work.

Target cross-layer mechanisms

Integrated monitoring mechanisms	–
Integrated and coordinated adaptation mechanisms	–
Means to identify adaptation needs across layers	A set of algorithms are specified that identify what should be replaced in the service based systems where the adaptation is triggered due to the different situations in the different functional layers of the service based systems.
Means to identify adaptation strategies across layers	A set of algorithms are described that specify how the service based systems should be adapted where the adaptation is triggered due to the different situations in the different functional layers of the service based systems.

Cross-layer model

Aspect	Various aspects (e.g. quality characteristics, functional characteristics, contextual characteristics) of service based systems.
Component	Service based system, policies to adapt the service based system
Key model elements	XML based language to specify the functional characteristics, quality characteristics and contextual characteristics of the service based system.

Possible extensions

Cross-layer mechanisms	Adaptation should be performed in the service infrastructure layer in addition to the service composition layer.
Cross-layer model	More robust adaptation policies that enable adaptation in the business process model as well as the service composition layer.
Layer elements	–

Covered elements at functional layer

	Monitoring Events	**Adaptation Actions**
Business Process Management	Changes of requirements or emergence of new requirements	Composition process is adapted according to the described algorithms
Service Composition	Functional/non-functional characteristics of participating services in service based system	Composition process is adapted according to the described algorithms
Service Infrastructure	Unavailability of a service, emergence of new service	Composition process is adapted according to the described algorithms

Extended abstract

The need to change service-based systems during their execution time has been recognized as an important challenge in service oriented computing. There are several situations from different SBA layers that may trigger changes in service-based applications such as unavailability or malfunctioning of services; changes in the functional, quality, or contextual characteristics of the services; changes in the context of the service-based system environment; emergence of new services; or changes in the requirements of the system. In order to provide support for dynamic changes in service-based systems, it is necessary to use replacement policies specifying what needs to be changed (what), the ways that the changes need to be executed (how), and the moment that the changes should be performed in the systems (when).

Changes in a service-based system can range from the replacement of a service by another service, or a composition of services, to changes in the execution process (e.g., conditions, loop statements, variables, exception handlers). The changes in a service-based system can be performed by stopping the system, making the necessary changes, and resuming the system. Other approaches can be used when replacing a service by another service in a system such as binding partner links during execution time of the system; using proxy services as place holders for the services in a composition, instead of having concrete services referenced in the system; or even using an adaptation layer based on aspect oriented programming with information about alternative services.

Furthermore, the moment to execute changes in a service-based system should also be considered in order to avoid (1) making changes when it is not really necessary or (2) making changes that may cause the system to behave incorrectly. Therefore replacement policies should take into consideration (a) the situations that trigger changes in the system, (b) the type of changes that needs to be performed in the system, and (c) if the parts in the system that require changes are being used. In this paper, we describe different types of replacement policies for situations that may arise in different functional layers of service-based systems. In our work, changes in a service-based system consist of replacing a service participating in the system by another service. We assume service-based systems represented in BPEL due to its popularity. We use a proactive service discovery framework that allows for the identification of replacement services in parallel to the execution of the SBA. We use proxy services to support changes in the system during execution time, avoiding changes in the original service-based system. The replacement policies take into consideration the position of a service S that may need to be replaced with respect to the current execution point of the system. There are three different positions that are considered, namely:

- *not_in_path:* when service S in not in the current execution path of the system, i.e., S appears in a different branch of the systems execution path or before the current point in the execution path;
- *current:* when service S is in the current execution point of the system;
- *next_in_path:* when service S is in the current execution path of the system, and will be invoked some time in the future.

Depending on the positions described above in the framework we consider cases when (1) replacements are required to be performed so that the system can continue its operations; (2) replacements that can wait to be performed after the current execution of the system; and (3) no replacements are required. Detailed description of the policies is given in [23].

Contribution to Cross-Layer Adaptation and Monitoring

This work presents a set of policies for dynamic adaptation of service based system where the adaptation may be triggered due to various situations in different functional layers of service based systems. For example, changes in the requirements or emergence of new requirements of the service based system in the BPM layer trigger adaptation of the composition that realizes the business process. Again changes

in the SI layer (e.g. unavailability or malfunctioning of a service participating in the service based application or emergence of a new service) trigger adaptation of the composition that realizes the business process to ensure that the service based system fulfils the requirements set in the BPM layer. In this work we describe a set of policies to handle these types of cross layer adaptation of service-based systems.

2.2.2 A Monitoring Approach for Runtime Service Discovery

Title	A Monitoring Approach for Runtime Service Discovery ([20])
Authors	Khaled Mahbub, George Spanoudakis and Andrea Zisman
Short description	In this work we propose a monitor based runtime service discovery framework (MoRSeD). The monitor component of the framework identifies the situations that may trigger the need for runtime service discovery (e.g. unavailability or malfunctioning of a participating service). In our framework services are identified based on structural, behavioral, quality and contextual characteristics of a system represented in query languages. The framework supports identification of services based on service discovery queries in both classic pull mode and proactive push mode of query execution.

Target cross-layer mechanisms

Integrated monitoring mechanisms	Specification of properties for different functional layers of SBA, e.g. contextual properties in the SI layer or behavioral properties in the BPM layer. Satisfiability of the specified properties is verified against the messages exchanged between a service-based system and the services participating in the system.
Integrated and coordinated adaptation mechanisms	–
Means to identify adaptation needs across layers	The monitor detects the runtime violation of different properties specified in the different functional layers of SBA that identifies the faulty services taking part in the service based system which need to be replaced.
Means to identify adaptation strategies across layers	–

Cross-layer model

Aspect	Various aspects (e.g. quality characteristics, functional characteristics, contextual characteristics) of service based systems.
Component	Service based system, monitoring (violation of properties specified in different functional layers of service based system)
Key model elements	XML based language to specify the functional characteristics, quality characteristics and contextual characteristics of the service based system.

Possible extensions

Cross-layer mechanisms	Automatic adaptation of service composition and/or service infrastructure layer.
Cross-layer model	Specification of adaptation strategies.
Layer elements	–

Covered elements at functional layer

	Monitoring Events	**Adaptation Actions**
Business Process Management	Behavioral properties of a service based system	–
Service Composition	–	–
Service Infrastructure	Unavailability of a service, change in the context of a service.	–

Extended abstract

The identification of services during the execution of service-based systems to replace services in them has been recognized as a key issue in service oriented computing. Existing approaches for runtime service discovery support the discovery process in pull mode in a reactive way, where services are identified when there is a need to do so. There are several situations that may trigger the need for runtime service discovery including, (i) unavailability or malfunctioning of a participating service, (ii) changes in the structure, behavior, quality, or context characteristics of a participating service, (iii) changes in the context of the service-based system, or (iv) availability of a better service due to the provision of a new service or changes in the characteristics of an existing service. Given the above situations and the need to provide better precision when identifying services to replace existing services during runtime, it is necessary to consider different characteristics of the services such as structural, behavioral, quality, or contextual characteristics. However, it is not possible to assume that services will be described in terms of all the above characteristics. Current approaches for service registries guarantee the existence of structural descriptions of services as WSDL specifications. In order to fulfill the need to identify services based on other criteria and not only structural characteristics, it is necessary to have a mechanism to verify the behavioral and contextual characteristics of services even when there are no available behavioral specifications and up-to-date contextual values of distinct aspects of the services (e.g., location, availability, response time) in the registries.

In this work we propose a monitor based runtime service discovery framework (MoRSeD). The monitor component of the framework identifies the situations that may trigger the need for runtime service discovery. In other word the monitor is responsible to (a) identify that services become unavailable; (b) identify that there are changes in the behavioral or contextual characteristics of the services participating in service-based system or replacement candidate services, (c) identify that there are contextual changes in the service-based system environment, and (d) verify if behavioral and contextual properties specified in the service discovery queries are satisfied by services. The monitor intercepts all the runtime messages exchanged between the service-based system and its constituent services and verifies the satisfiability of the properties against these messages. The monitor invokes the context services to verify the satisfiability of contextual properties.

MoRSeD can execute service discovery in both pull and push modes. The pull mode of query execution is performed to (a) identify services that may be initially bound to a service-based system, (b) as a first step in the push mode of query execution, (c) due to changes in the context of an application environment, and (d) when a client application requests a service to be identified. On the other hand, the push mode of query execution is performed in parallel to the execution of a service-based system, in a proactive way, in order to identify services due to any of situations (i) to (iv) described above.

In the framework, queries are specified in an XML-based query language, called SerDiQueL that allows for the representation of different criteria, namely: (a) structural, describing the interface of a required service, (b) behavioral, describing the functionality of a required service, and (c) constraints, describing additional conditions for a service. These additional conditions may be concerned with quality characteristics of a service, or interface or functional characteristics of a service that cannot be described in terms of the structural and behavioral descriptions used in SerDiQueL. The constraints in a query can be classified as contextual and non-contextual. A contextual constraint is concerned with information that changes dynamically during the execution of a service-based system or its participating services. A non-contextual constraint is concerned with information that does not change dynamically. The matching of service discovery queries against services is executed in a two-stage process. The first stage is called filtering stage, where hard non-contextual constraints in a query are evaluated against service specifications and a set of candidate services that comply with these constraints are identified. The second stage in the matching process is a ranking stage that is executed in a three sub-stage process. In the first sub-stage structural and behavioral characteristics of a query are evaluated against the candidate services and partial distance is computed for each candidate service with respect to a query. In the second and third sub-stage soft non-contextual and contextual constraints are evaluated respectively. In the ranking stage,

the overall distance between a query and each candidate service is calculated by taking the average of all the partial distances. The result of the ranking stage is a set of candidate services that have an overall distance with a query that is below a certain distance threshold. A detail description of MoRSeD can be found in [20].

Contributions to Cross-Layer Adaptation and Monitoring

This work presents a framework for runtime service discovery where the discovery is triggered by a monitor. The monitor component of this framework supports cross layer monitoring to detect the situations that trigger runtime service discovery. More specifically, the monitor is able to monitor properties specified at the service infrastructure level to identify the unavailability of a service participating in the SBA, or change in the context characteristics of a service participating in the SBA. Again the monitor is able to monitor properties specified at the business process management layer to identify the change in the behavioral characteristics of a service participating in the SBA.

2.2.3 Adaptation of Service-Based Applications Based on Process Quality Factor Analysis

Title	Adaptation of Service-Based Applications Based on Process Quality Factor Analysis ([15])
Authors	Raman Kazhamiakin, Branimir Wetzstein, Dimka Karastoyanova, Marco Pistore, and Frank Leymann
Short description	The work presents a novel approach for cross-layer SBA adaptation. It uses decision trees for showing the dependencies of KPIs on process quality factors from different functional levels of an SBA. It then uses the analysis results to come up with an adaptation strategy to improve the KPI values. The approach includes creation of a model which associates adaptation actions to process quality metrics, extraction of adaptation requirements based on analysis results, and identification of an adaptation strategy which can consist of several adaptation actions on different functional levels of an SBA.

Target cross-layer mechanisms

Integrated monitoring mechanisms	Analysis of influential quality factors based on data mining and dependency tree analysis techniques
Integrated and coordinated adaptation mechanisms	–
Means to identify adaptation needs across layers	SMT-based analysis algorithms to perform search in the results of the dependency tree analysis to identify critical combination of quality factors to be improved across functional layers.
Means to identify adaptation strategies across layers	A specific algorithm to select a combination of adaptation actions (adaptation strategy) that improves the identified quality properties and satisfies the optimality criteria (in particular, the least negative effect)

Cross-layer model

Aspect	Quality characteristic of the SBAs at different layers
Component	SBA model (quality metrics), Adaptation (adaptation actions), Monitoring (violation of KPIs)
Key model elements	Formal model of quality metrics (range, target, potential influential factors); Formal model of adaptation actions (mechanism, positive/negative effect)

Possible extensions

Cross-layer mechanisms	Integrated and coordinated cross-layer adaptation execution engine
Cross-layer model	More sophisticated model of adaptation actions and their effects
Layer elements	Realization and integration of specific adaptation actions at different layers: ad-hoc process modification; service replacement; service re-composition/fragmentation; infrastructure re-configuration

Covered elements at functional layer

	Monitoring Events	**Adaptation Actions**
Business Process Management	KPI violation in business processes	generic [1]
Service Composition	values of PPMs	generic [1]
Service Infrastructure	values of QoS metrics	generic [1]

[1] In the scenario the following adaptation actions are referred to: SLA re-negotiation, outsourcing of process fragment, service replacement through dynamic discovery/binding and service replacement through predefined selection

Extended abstract

One of the major concerns for enterprises is to ensure the quality of their service-based applications, realized as business processes. Thereby, process quality goals are specified in terms of Key Performance Indicators (e.g., order fulfillment time), i.e. key process metrics that contain target values which are to be achieved in a certain period. KPIs of business processes that are implemented in terms of SBAs are typically monitored using business activity monitoring technology. If monitoring results show that KPIs do not meet target values, the adaptation of SBA is needed in order to adjust the quality of the process to the required level. The adaptation, however, cannot be done directly as the violation of KPI may occur due to a wide range of possible causes. Indeed, SBAs can be viewed in terms of three functional layers, namely (i) business processes, (ii) service compositions that implement these business processes, and (iii) services and service infrastructure. The problem may occur at any of these levels and, therefore, further analysis is necessary in order to find out the actual source(s) of the problem. That is, the process *quality factor analysis* aims to identify the lower level process metrics (e.g., duration of process activities, type and amount of ordered products etc.) or QoS metrics (e.g., availability of IT infrastructure) mostly influence KPI target violations.

This, however, is not enough. To guarantee that the adaptation reaches its goal, it is necessary to properly identify, aggregate, and coordinate those adaptation activities, which 1) address the influential quality factors identified by the analysis and 2) to ensure that they will not affect other critical KPIs and metrics. An appropriate modeling of the SBA, its quality factors and KPIs is needed, as well as of the corresponding adaptation actions and their outcomes (positive and negative effects on the quality metrics of the SBA).

The work presented in [15] aims to provide a solution towards the above problems. The work represents a generic cross-layer adaptation and monitoring framework that aims to improve SBA quality when the KPIs are violated. The overall framework is represented in Figure 2.2. This approach consists of the following four phases:

Quality modeling for analysis and adaptation. At design time the metrics model and the adaptation actions model are created. In the *metrics model*, the user specifies the application KPIs, and the quality metrics representing the potential influential factors of KPIs coming from different functional layers and components of SBA. Obviously, the user does not yet know the influential factors (but might suspect them), however he has to model potential metrics so that they are monitored in the first place and thus can be used during analysis. In the *adaptation actions model*, the user(s) specifies the adaptation actions (available at different layers) per metric and the effect of those actions on application metrics specified in the metrics model. In particular, this model allows for defining whether an action contributes positively or negatively to a certain quality factor, i.e., whether it improves the value of a metric. Same as for metrics, the user just specifies the potential adaptation actions in isolation at this phase, without knowing yet which of those will be needed at adaptation time and in which combination.

Analysis of influencing quality factors. In the second phase, based on the *metrics model*, the monitoring of KPIs and potential influential quality metrics is performed across the instances of the application; the information is continuously aggregated and updated. Then the metrics related to *previous executions* of a given application are analyzed in order to identify the reasons, i.e., the *influential factors*, which lead to the undesired values of the specified KPIs. More precisely, machine learning techniques are used to construct a decision tree which shows for which value ranges of influential application metrics a KPI is satisfied or violated (see also [32]). As a result, one identifies those tree paths of application metrics (and their value ranges) that correspond to the bad values of the KPI and thus fail the underlying business goal. From the identified tree paths we extract a set of (alternative) *adaptation requirements* each consisting of a conjunction of predicates over metric values which lead to KPI satisfaction. The result of the analysis characterizes thus those factors of

the application that should be improved, i.e., that are subject of adaptation, and how they should be improved (their values).

Identification and selection of an adaptation strategy. In the next phase the approach aims to combine, and enact concrete *adaptation actions* that address the identified requirements as part of a coherent *adaptation strategy*. This phase relies on the adaptation action model, where the effect of those actions on different application metrics is described. It takes into account that an adaptation action contributes positively or negatively to a certain quality factor, i.e., whether it improves the value of a metric. After identification of a set of alternative adaptation strategies, one strategy is selected based on certain criteria. In particular, one of the criteria is the number of negative effects the strategy has: the less this number the better strategy is.

Process adaptation. The selected adaptation strategy is used for adaptation of the process model or process instance by executing all contained adaptation actions. After adaptation, the existing KPIs and metric definitions might have to be adapted thus closing the loop. The presented work does not describe the adaptation execution explicitly, rather it focus on the monitoring and strategy identification aspects. This part of the framework will be designed and presented in the future works.

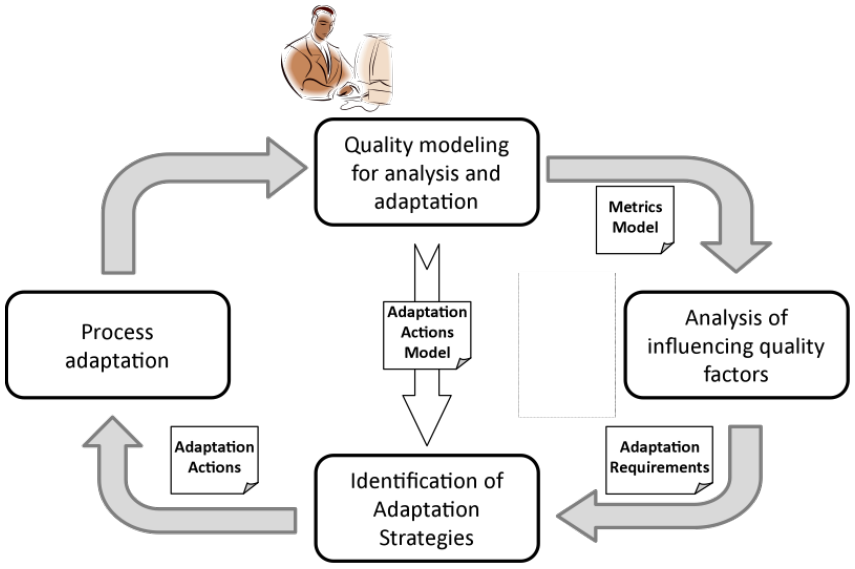

Figure 2.2: Quality factor analysis and adaptation framework

The work relies on a simple yet expressive formal model that on one side makes the analysis efficient, and on the other side is generic enough to incorporate and integrate models and aspects of different layers, adaptation actions, and their effects. The model includes the following elements:

Quality Metric model. Quality metric is represented with its value domain, the set (or range) of target values, and a set of other potential influential factors for this metric. This model is equivalently applicable to the quality metrics of different elements/layers of the SBAs.

Adaptation Action model. To model adaptation of the SBA elements the adaptation activities are represented with the implementing mechanism (that, however, out of the scope of the presented work) and with effects of the adaptation action. The effect is characterized as a set of metrics, to which the adaptation action contributes positively (i.e., improves them) and a set of metrics to which the action contributes negatively (i.e., worsens them). Again, also this model is agnostic to the specific

implementation of the adaptation mechanism and is very suitable for the cross-layer analysis and adaptation.

The set of analysis techniques exploited in the presented work adopt the solutions from data mining and decision tree algorithms [34] (to identify influential quality factors), satisfiability modulo theories (SMT) [26] to identify the adaptation requirements (i.e., to select the set of metrics that should be improved in order to improve the violated KPI), and a specific search-based algorithm to select the adaptation strategies in an optimal way according the defined criteria.

Contributions to Cross-layer Adaptation and Monitoring

The work in [15] defines a cross-layer adaptation and monitoring framework, where the focus is on the monitoring, analysis and improvement of quality properties of the SBA across all its layers and components. The contributions of the presented approach target various elements of the problem.

First, the analysis of influential factors is performed across all the metrics at different layers and relies on the capabilities to monitor those metrics at different layers. In terms of cross-layer adaptation and monitoring requirements, this corresponds to the lack of alignment of monitoring events problem.

Second, based on the quality factors identified, the approach aims to identify the adaptation requirements across the layers, i.e., to identify which metrics at different layer should be improved.

Third, the framework aims to identify the best possible combination of adaptation actions at different layers that will address the adaptation needs in a holistic and integrated way. These two contributions, therefore, aim to address the lack of adaptation integrity and compatibility problems.

The model used in this approach aims to capture the relevant quality properties at different functional layers and the adaptation actions. The focus is on modeling the quality properties and their target values, as well as on the cross-layer relations between those properties.

The future extensions of this work aim to 1) complement the presented results with the adaptation execution and coordination capabilities and to 2) extend the model and the corresponding analysis techniques in order to capture more sophisticated properties and features of the quality of SBAs.

2.2.4 Improving the Adaptation of Service-Based Applications by Exploiting Assumption-Based Verification Techniques

Title	Improving the Adaptation of Service-Based Applications by Exploiting Assumption-Based Verification Techniques ([9])
Authors	A. Gehlert, A. Bucchiarone, R. Kazhamiakin, A. Metzger, M. Pistore, K. Pohl
Short description	The work presents an approach to support SBA adaptation. The approach relies on explicit modeling of domain assumptions that are necessary for the proper functionality of the SBA, and their consequent verification and monitoring to trace the possible problems and to trigger appropriate adaptation actions.

Target cross-layer mechanisms

Integrated monitoring mechanisms	Monitoring of SBA assumptions extracted from the design time modeling and verification
Integrated and coordinated adaptation mechanisms	–
Means to identify adaptation needs across layers	Run-time verification of SBA against the requirements identified through the trace link of the monitored violated assumptions
Means to identify adaptation strategies across layers	Use of explicit trace links to the SBA elements to identify the elements of the specification that are subject to adaptation

Cross-layer model

Aspect	Assumptions over the SBA context and external services
Component	SBA model (assumptions and requirements), Monitoring (assumption violations)
Key model elements	Explicitly codified assumptions; trace links between assumptions and the requirements and between the assumptions and SBA elements

Possible extensions

Cross-layer mechanisms	New types of the analysis techniques to accommodate new types of assumptions; cross-layer adaptation mechanisms
Cross-layer model	New types of assumptions targeting different layers and aspects
Layer elements	Assumptions at BPM, SCC, and SI layers including KPIs, resource allocations, QoS properties

Covered elements at functional layer

	Monitoring Events	**Adaptation Actions**
Business Process Management	–	–
Service Composition	Deviations from conversational service protocol; violation of temporal properties over composition execution	Service replacement, re-composition
Service Infrastructure	–	–

Extended abstract

Service-based applications (SBAs) need to operate in a highly dynamic world, in which their constituent services can continuously change, fail or even become unavailable. The owner of the SBA cannot control the externally provided services and the context of the SBA. Therefore, SBAs need to be built in a way that they can adapt to deviations from their requirements. Monitoring is typically used to identify such deviations and, if needed, to trigger an adaptation of the SBA. However, existing monitoring approaches exhibit several limitations:

- The approaches for monitoring individual services (or individual context properties) can recognize

whether those services deliver the specified quality or functionalities (or the context matches the expectation). Yet, it remains open whether those individual mismatches lead to a violation of the SBA's requirements, which would necessitate an adaptation.

- Monitoring the requirements of the whole SBA can uncover deviations from those requirements. However, this will not provide information about the root cause that lead to this deviation. Without this information, the adaptation activities may be useless or even dangerous to the application, as they do not address the real synopsis of the requirement violation.

Given the complexity of the SBAs and their constituent functional layers, the diversity of possible requirements, contexts, changes and violations may be very large. In these settings, the identification of proper source of the requirement violation is the cornerstone for the dynamic SBA adaptation. Such functionality aims to identify what should be adapted in the application in order to compensate those violations, i.e., aims to provide means to identify adaptation needs across functional SBA layers.

The work presented in [9] addresses this challenge. To achieve this, the approach builds on a clear separation between the requirements for the SBA and the assumptions under which it is supposed to operate. Furthermore, it distinguishes between the system itself and its domain (context). In particular, the constituent services of the SBA and the underlying infrastructures belong to the domain. Indeed, they are delivered by different providers and the control over these elements is outside of the SBA bounds. The assumptions are used to capture those properties of the SBA environment that make the SBA working according to its requirements. Assumptions may be used to capture functional and non-functional properties of the SBAs; they may refer to different aspects and elements of different layers of the functional stack.

The approach is realized through the following phases (Figure 2.3):

1. At design time the designer documents assumptions and requirements separately ((1) in Figure 2.3). The assumptions may be extracted from the domain knowledge (the most used services and their APIs, the properties of the infrastructures in different settings), may be obtained from historical information, etc. The key aspect is that the assumptions, the SBA, and the requirements are explicitly related in this phase through special trace links: the one that relate assumption to the requirement (the requirements hold only if the assumption holds) and the one relating assumption to the part of the SBA that is affected by the assumption violation and, therefore, is subject to adaptation.

2. In the second phase the designer verifies the system at design time and deploys the system only if the system passes the verification step ((2) in Figure 2.3). The verification aims to check that the modeled SBA satisfies all the functional and non-functional requirements given the set of assumptions made in the previous phase. If the verification fails, the set of assumptions should be extended or the system design should be changed. As an outcome, we obtain a system specification and a set of documented assumptions, which ensure the expected functionalities and properties of the SBA.

3. At run-time the assumptions are monitored ((3) in Figure 2.3). The monitoring aims not to monitor the requirements or the constituent services, but to check that the assumptions made at design-time phase hold. This also allows one to target more compact set of monitored properties, making the monitoring and the analysis more efficient.

4. When a violation of the assumption is detected, it is necessary to identify consequences of this violation for the SBA. Given the trace link between the assumption and the requirements, the violation allows one to see which requirements may be affected. To see whether this negative effect really takes place, run-time verification of the SBA against these specific requirements is triggered ((4) in Figure 2.3). This verification may take into consideration the actual situation (e.g., specific context, the new functionality of the service or its new QoS levels).

5. If the SBA does not pass the verification given the new set of assumption, an adaptation is triggered in ((5) in Figure 2.3). In our approach the identification of the problem source is straightforward as the model of assumption has already been associated to the elements of the application model that rely upon those assumptions. This relation allows us to identify the component of the system that is subject to adaptation. A specific element of the SBA domain, such as constituent services and the context (e.g., connectivity, or user profiles) may be further associated with the appropriate adaptation actions. The ability to precisely identify the critical elements of the domain allows us to identify adaptation actions that are the most appropriate in a given situation and, therefore, avoid redundant or harmful adaptations.

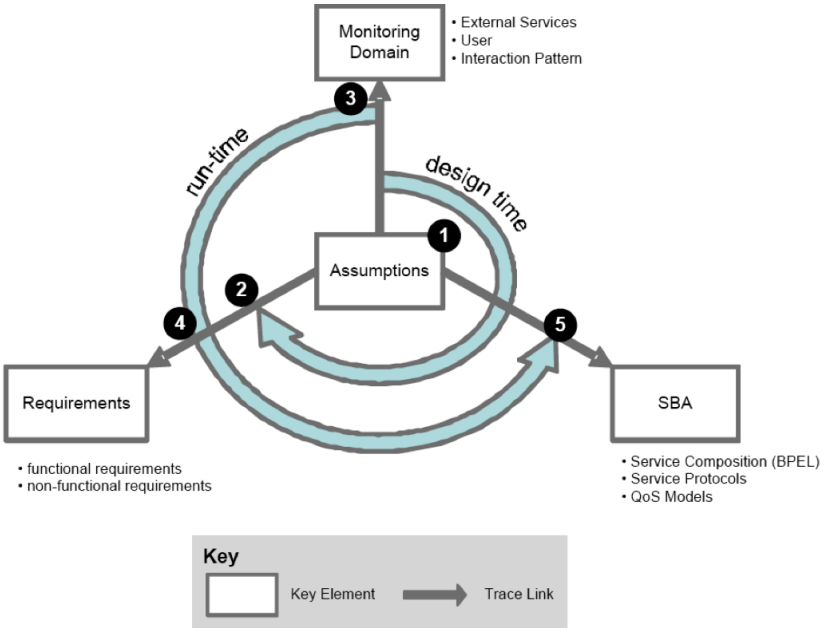

Figure 2.3: Assumption-based framework

In [9] the presented framework is instantiated with the specific types of the SBA requirements and aspects. In particular, it is demonstrated how the behavioral requirements and behavioral assumptions of the SBA and the environment (constituent services) can be modeled, formalized, verified (using model checking techniques), and monitored. As for non-functional properties, the time properties over the underlying process and service executions have been considered. For this purpose, the formalization builds upon the timed verification and monitoring techniques.

Contributions to Cross-layer Adaptation and Monitoring

The presented approach represents a generic framework that may be used to address the problem of the SBA adaptation across functional layers.

The key factor for this is the explicit model of the assumptions and the trace links. This explicit model is agnostic to the specific aspect or notation used to capture elements of the SBA or of its domain at different layers. As was mentioned before, the assumptions and requirements may characterize various properties and are not tailored to a specific solution. In this way it may relate different layers, approaches, and notations.

Second, the framework may be used to integrate variety of analysis (verification and monitoring) techniques that are used to address particular properties and models. Initial results show how the behavioral and time properties, as well as the verification techniques may be exploited in the framework.

These results will be extended in the future. Specifically, it is planned to address all the layers of SBA simultaneously, namely the BPM, the Service Composition, and the Service Infrastructure layer. For example, on the BPM layer, assumptions could be stated in terms of business KPIs and expectations. On the SI layer, assumptions could be stated, for instance, about the availability of the deployment platform or the reliability of the communication infrastructure, about the resource usage, etc. As for the analysis, simulation techniques or the ones relying on the probabilistic information may be adopted.

2.2.5 Self-supervising BPEL Processes

Title	Self-supervising BPEL Processes ([1])
Authors	Luciano Baresi and Sam Guinea
Short description	The approach presented in this article fosters the idea of dependable BPEL processes by proposing supervision rules as means to let designers decide the self-supervision capabilities they want to ascribe to their processes. Supervision consists of monitoring and reaction. By the former, we synchronously check the execution to see whether everything proceeds as planned. The latter is only activated if an anomaly is discovered, and attempts to fix the execution and keep things on track.

Target cross-layer mechanisms

Integrated monitoring mechanisms	Probes added to the process execution by means of special-purpose aspects injected directly into the execution engine. The process remains untouched, but the monitoring elements are added to the executor. The properties that can be monitored must be defined through a proprietary language called WSCoL.
Integrated and coordinated adaptation mechanisms	Local and global recovery actions that can be properly composed into complete strategies, and be activated in particular points of the composition. A dedicated language called WS-ReL defines the set of primitive actions.
Means to identify adaptation needs across layers	The user is in charge of setting the probes. The violation of a monitoring constraint usually triggers an adaptation strategy.
Means to identify adaptation strategies across layers	–

Cross-layer model

Aspect	Nothing specific
Component	Nothing specific
Key model elements	Nothing specific

Possible extensions

Cross-layer mechanisms	Correlation of monitoring constraints and recovery actions at the different levels.
Cross-layer model	Identification of key elements at the different levels and materialization of their relationships
Layer elements	Up to the user as long as they provide a WSDL interface and do not interfere with the actual state of the process

Covered elements at functional layer

	Monitoring Events	**Adaptation Actions**
Business Process Management	Violation of WSCoL assertions	Those defined by WSReL
Service Composition	Violation of WSCoL assertions	Those defined by WSReL
Service Infrastructure	Violation of WSCoL assertions	Those defined by WSReL

Extended abstract

This approach augments BPEL processes with self-supervising capabilities. This is achieved by defining appropriate *supervision rules*. First of all, each rule must indicate the precise point in the process in which it is considered. This is done by specifying the rule's `location`, that is an XPath expression that uniquely identifies a BPEL *invoke, receive, reply,* or *pick* activity within the process definition. This means that any BPEL activity that interacts with the outside world is a valid location. When defining the location we also specify if the rule is to be considered before or after the activity's execution (i.e., if it is a pre- or a post-condition).

A supervision rule also contains a set of `supervision parameters`. This meta-level information is used at run time to decide whether a rule needs to be considered or not. The reason is that supervision necessarily introduces a performance overhead, and we want to be able to tailor the exact amount of supervision depending on the needs at hand without changing or redeploying the process.

Finally, a supervision rule is made up of a monitoring expression, specified in WSCoL, and a set of alternative recovery strategies, specified in WSReL.

Supervision parameters allow a designer to tailor the degree of supervision that will be achieved by specifying when a rule can be "switched-off". Our supervision parameters are *priority*, *validity*, *delay*, and *trusted providers*. All four are optional, and for each there is a default meaning.

WSCoL

WSCoL (Web Service Constraint Language) is the assertion language we defined to specify what the process expects from partner services. It is evocative of assertion languages, such as ANNA (Annotated Ada), and JML (Java Modeling Language), but given the syntax of BPEL and Web services, WSCoL also takes inspiration from XML technology (e.g., XPath).

WSCoL is holistic. We do not limit ourselves to asserting on the process' internals; we also consider aspects regarding the environment the process is run in. Our definition of environment is very broad, and comprises whatever data we can collect at run time through external probes. For example, we can also assert on data belonging to previous process executions. This is reflected in the three kinds of variables handled by WSCoL.

The syntax for WSCoL assertions is defined as:

$$
\begin{array}{lll}
\langle \mathsf{asrtn} \rangle & ::= & \neg\langle \mathsf{asrtn} \rangle \mid \langle \mathsf{asrtn} \rangle \&\& \langle \mathsf{asrtn} \rangle \mid \langle \mathsf{asrtn} \rangle \| \langle \mathsf{asrtn} \rangle \mid (\langle \mathsf{quant} \rangle\ \langle \mathsf{alias} \rangle\ \mathit{in}\ \langle \mathsf{values} \rangle, \langle \mathsf{asrt} \rangle) \mid \\
 & & \langle \mathsf{term} \rangle \langle \mathsf{rop} \rangle \langle \mathsf{term} \rangle \\
\langle \mathsf{trm} \rangle & ::= & \langle \mathsf{var} \rangle \mid \langle \mathsf{trm} \rangle \langle \mathsf{aop} \rangle \langle \mathsf{trm} \rangle \mid \langle \mathsf{const} \rangle \mid \\
 & & \langle \mathsf{var} \rangle . \langle \mathsf{sfun} \rangle (\langle \mathsf{trm} \rangle *) \mid (\langle \mathsf{afun} \rangle \langle \mathsf{alias} \rangle\ \mathit{in}\ \langle \mathsf{var} \rangle, \langle \mathsf{trm} \rangle) \\
\langle \mathsf{rop} \rangle & ::= & < \mid \leq \mid == \mid \geq \mid > \\
\langle \mathsf{quant} \rangle & ::= & \mathit{forall} \mid \mathit{exists} \mid \mathit{numOf} \\
\langle \mathsf{aop} \rangle & ::= & + \mid - \mid \times \mid \div \mid \% \\
\langle \mathsf{sfun} \rangle & ::= & \mathit{abs} \mid \mathit{replace} \mid \mathit{substring} \mid \ldots \\
\langle \mathsf{afun} \rangle & ::= & \mathit{sum} \mid \mathit{avg} \mid \mathit{min} \mid \mathit{max} \mid \mathit{product}
\end{array}
$$

where var is a variable, a variable alias, or a special purpose alias called `$instanceID` which returns the ID of the process currently being run, sfun are simple functions that mimic those commonly used in XPath, and afun are aggregate functions meant to be used with variables that have multiple values (containers). Boolean, relational (relop), and arithmetic operators (arop) follow their usual definitions.

Designers can use universal and existential quantifiers to express constraints over finite sets of values[1]. Their meanings are straightforward. When using a quantifier, the designer must define three parts. The alias names a variable that will be used as a parameter in the upcoming assertion, values uses the syntax shown previously for variables to define the range of values that the alias can assume, and assertion defines the parametric assertion.

[1] Since we deal with finite sets of data these constructs do not actually add expressive power to the language, but have been included for convenience.

WSReL

WSReL (Web Service Recovery Language) extends upon the legacy of WSCoL to provide a programmable, flexible, and extensible solution for both *local* and *backward* recovery. The former tries to fix the anomaly in the current state of error, in a way that is similar to compensation. Indeed, once the corrective actions are performed, the system tries to continue its normal execution from the same state. The latter, on the other hand, tries to restore the system to a previous state in which the anomaly was not present.

The atomic actions we provide can be organized in four main groups. The first group is made up of simple actions that do not modify how the process is playing out. The second group comprises recovery actions that alter the amount of supervision being performed either by modifying the supervision parameters or the supervision rules themselves. The third group changes the services with which the process does business. The fourth group contains general actions.

Recovery Strategies

Designers can create multiple recovery strategies by mixing atomic actions. WSREL is reminiscent of rule-based approaches, and allows us to choose the more suitable course of action, depending on what is going on in the process and in the surrounding environment. The syntax for defining strategies is defined as:

⟨strategy⟩ ::= *try* {⟨step⟩} (*elsetry*{⟨step⟩})* (*else*{step})?
⟨strategy⟩ ::= *if*(⟨condition⟩) {⟨strategy⟩}
(*elseif*(⟨condition ⟩) {⟨strategy⟩})*
(*else*{⟨strategy⟩})?
⟨step⟩ ::= (⟨action⟩)+

where condition is a WSCoL expression or a special keyword NoResp, and action is an atomic action taken from those presented in the previous section. The special keyword NoResp is true if a service invocation did not answer before the timeout was reached.

A strategy is a sequence of steps. Each step is wrapped in a `try` block, and contains an ordered list of atomic actions chosen from those presented previously. The semantics of the sequence is that first we try to fix the anomaly by executing the atomic actions contained within the first step. If at least one of these actions requires monitoring to be re-enacted, we do so to verify if the anomaly persists. If we are not successful we try with the second block, and so on. To facilitate the definition of these steps, each is executed considering the original state of anomaly, as if no other block execution had been attempted until then.

We also allow designers to define more than one sequence so that the system can choose the most appropriate at run time. The syntax allows us to specify alternative branches that are chosen by checking WSCoL expressions. The order of the `if-elseif-else` branches determines the order in which conditions are evaluated, and how the overall recovery will play out. If designers want to react to a service not responding, this should be their first priority. If they fail to do so, the default behavior is to terminate the process.

Contributions to Cross-layer Adaptation and Monitoring

This paper presents two languages for specifying monitoring directives, in the form of assertions associated with service invocations, and recovery activities, as suitable sequences of atomic actions, to keep the correctness of the system above a given threshold. This means that the approach does not provide special-purpose primitives to foster cross-layer adaptation and monitoring, but the same languages, which are extensible, can be used to predicate and constrain the execution at different levels. Where to put the probes and how to deploy them is up to the designer: the framework, along with its two languages, offers

a neutral environment to state requirements, identify adaptation and execute them. The different layers can be integrated in two different ways:

- We can define special-purpose probes and actuators to correlate values, events, and activities at the different levels. The only constraint is that the must expose a WSDL interface, but then they could help the designer intertwine the activities at the different levels. No policies are predefined, but the designer is free to embed what s/he needs and prefers.
- We can also easily support probes that are invoked by assertions at a given level and grab data at a different level: for example we can predicate on speed and throughput at application level.

2.2.6 Integrated and Composable Supervision of BPEL Processes

Title	Integrated and Composable Supervision of BPEL Processes ([2])
Authors	Luciano Baresi, Sam Guinea, and Liliana Pasquale
Short description	The flexibility and dynamism embedded in BPEL processes require that suitable runtime supervision be deployed along with the processes themselves. Clients must be guaranteed of established SLAs even if presence of misbehaving partner services, and processes must react to anomalies efficiently and autonomously. Moreover, loosely coupled (or post-mortem) analyses are mandatory to anticipate possible problems, and optimize the new instances. Different proposals have already solved the problem partially, but none provides a complete (and flexible) solution. The paper argues that the many facets of supervision do not demand for a single holistic solution, and proposes a composable approach, where a single framework provides the glue for different probing, analysis, and recovery capabilities. The paper introduces the framework, exemplifies its main features on a simple case study, and describes a first prototype implementation.

Target cross-layer mechanisms

Integrated monitoring mechanisms	Possibly any since the approach proposed an extensible set based on plug-ins
Integrated and coordinated adaptation mechanisms	Possibly any since the approach proposed an extensible set based on plug-ins
Means to identify adaptation needs across layers	User-set probes
Means to identify adaptation strategies across layers	User-defined strategies

Cross-layer model

Aspect	–
Component	–
Key model elements	Extensible model based on process data, external/context data, and historical data

Possible extensions

Cross-layer mechanisms	None. The approach already fosters extensibility
Cross-layer model	None. The approach already fosters extensibility
Layer elements	None. The approach already fosters extensibility

Covered elements at functional layer

	Monitoring Events	**Adaptation Actions**
Business Process Management	Violation of monitoring assertions defined with the languages integrated in the framework	Adaptation actions provided by the approaches integrated in the framework
Service Composition	Violation of monitoring assertions defined with the languages integrated in the framework	Adaptation actions provided by the approaches integrated in the framework
Service Infrastructure	Violation of monitoring assertions defined with the languages integrated in the framework	Adaptation actions provided by the approaches integrated in the framework

Extended abstract

This paper proposes a unified framework built upon the decoupling of data collection, monitoring, and recovery. Data collection is independent of the types of supervision approaches combined. Monitoring uses these data to check functional and non-functional properties, while recovery uses them, together with the monitoring results, to attempt to fix the process, produce a log, or perform post-mortem activities to prevent them from happening again.

When conceiving our framework, we wanted to satisfy different requirements. It had to support different quality dimensions and different analyses. Synchronous analysis can be used to evaluate punctual process properties (e.g., the response time of partner services). Asynchronous analysis can be exploited to measure temporal process properties (e.g., the number of times a synchronous check is violated). Post-mortem analysis can be used to construct a symptom model of process failures. Our framework should apply suitable recoveries with different timeliness depending on the analysis that signaled the violation.

The distinction among data collection, monitoring and recovery allowed us to conceive a neater architecture. Data collection fosters the neat separation between monitoring and recovery activities. We support *internal data*, which carry the internal state of the process, *external data*, which provide information from the surrounding environment, and *historical data*, which represent information collected in past executions. Different monitoring approaches are also able to share partial results and collaborate towards a more complete final assessment. Our framework can trigger corrective actions on the same process instance, on different instances (of different processes), and also on the process definition. To this end, we allow the interplay between synchronous and asynchronous actions and we provide conflict resolution mechanisms associating them to a priority.

Figure 2.4 shows the overall architecture of our solution. Each *BPEL Engine* is an instance of *ActiveBPEL Community Edition Engine* augmented with AOP (aspect-oriented) probes to collect process state data. The *Data Manager* is responsible for collecting external data, and for retrieving and storing historical data from/in the *Data Repository*. The *Monitoring Farm* holds the monitoring plug-ins we want to use, while the *Recovery Farm* holds the recovery capabilities.

The *Event Controller* is the central element of our architecture and it is based on rule engine technology [25]. It is in charge of activating external and historical data collection, as well as any monitoring and recovery activity. While internal variables are passively received from the AOP probes embedded in ActiveBPEL, external and historical variable collection, like also monitoring and recovery activities, are triggered when the process starts or when it reaches a particular state. This is achieved by defining rules on the data contained in the working memory, that is, process state data, external and historical variables, collected through the *Data Manager*, and analysis results.

Our framework also provides a configuration tool called *Supervision Manager*, used to configure the various components of the framework. In particular, it configures the AOP probes to collect internal data, the *Event Controller*, to retrieve external and historical data, and defines how monitoring and recovery are activated. Finally, it also sets the *Monitoring Farm* with the constraints each plug-in is supposed to

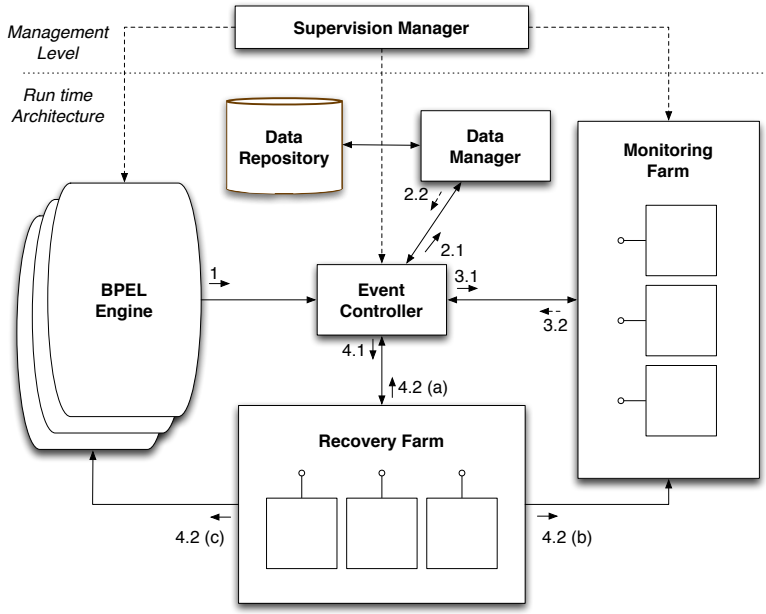

Figure 2.4: The Unified Framework.

check.

The numbered arrows of Figure 2.4 explain how the framework works. Before starting the execution of any process, the *Supervision Manager* configures the different components to allow them to correctly perform supervision tasks. After this, every time an executing process terminates an activity, the AOP probes collect internal data and give them to the *Event Controller* (transition 1), which inserts them in its embedded working memory. The data available in the working memory of the *Event Controller* can always activate suitable rules to require the *Data Manager* to retrieve external or historical data (transition 2.1) and store them in its working memory (transition 2.2).

Contributions to Cross-layer Adaptation and Monitoring

This paper proposes a framework to integrate different monitoring and recovery approaches through a shared space and user-defined rules. Even if the experiments were conducted on a well-defined set of approaches, the framework is able to accommodate many more approaches both for monitoring and adaptation. This means that they can easily work at different levels, and the integration among layers can be obtained through the shared space and rules. Again, the idea here is not to force the designer to adopt, or implicitly use, any predefined cross-layer monitoring and adaptation solution, but the aim is to provide suitable and trustable means to let the user integrate the elements s/he wants to use to supervise the execution of designed systems. This solution is extremely important for two reasons. Firstly, there is no solution that is able to cover all possible aspects, and thus any particular solution would force the designer to adopt a “partial” solution. Secondly, different processes, and even different instances of the same process, may need different supervision frameworks and strategies, and thus a single framework/solution would not be enough.

2.2.7 Autonomic Resource Virtualization in Cloud-like Environments

Title	Autonomic Resource Virtualization in Cloud-like Environments ([18])
Authors	Attila Kertesz, Gabor Kecskemeti and Ivona Brandic
Short description	We describe an autonomic architecture for SLA-based resource virtualization that incorporates enhancements of a meta-negotiation component for generic SLA management, a meta-brokering component for diverse broker management and an automatic service deployment for resource virtualization on the Cloud. We discuss how the principles of autonomic computing can be incorporated to the service infrastructure layer.

Target cross-layer mechanisms

Integrated monitoring mechanisms	–
Integrated and coordinated adaptation mechanisms	The negotiated service level agreements control the adaptation and autonomic behaviour of the service selection, brokering, and deployment functions of the infrastructure.
Means to identify adaptation needs across layers	The use of stricter service level agreements towards the lower levels of the SBAs enables re-negotiation actions on the higher levels as a response to lower level violations or predicted violations.
Means to identify adaptation strategies across layers	–

Cross-layer model

Aspect	Service Level Agreements
Component	Monitoring (tracking brokers, service instances), Adaptation (autonomous actions)
Key model elements	–

Possible extensions

Cross-layer mechanisms	Possible SLA violations can be propagated upwards to SC layer.
Cross-layer model	More detailed SLA model that can be useful for adapting the different components.
Layer elements	–

Covered elements at functional layer

	Monitoring Events	**Adaptation Actions**
Business Process Management	–	–
Service Composition	–	–
Service Infrastructure	tracking brokers, service instances	bootstrapping different negotiation strategy, re-evaluating broker ranking, deploying new service instances

Extended abstract

Grid Computing has succeeded in establishing production Grids serving various user communities all around the world. Cloud Computing is a novel infrastructure that focuses on commercial resource provision and virtualization. Both Grids and Service Based Applications (SBAs) already provide solutions for executing complex user tasks, but they are still lacking non-functional guarantees. The newly emerging demands of users and researchers call for expanding service models with business-oriented utilization (agreement handling) and support for human-provided and computation-intensive services. Providing guarantees in the form of Service Level Agreements (SLAs) are highly studied in Grid Computing. Nevertheless in Clouds, infrastructures are also represented as a service that are not only used but also

installed, deployed or replicated with the help of virtualization. These services can appear in complex business processes, which further complicates the fulfillment of SLAs in Clouds. For example, due to changing components, workload and external conditions, hardware and software failures, already established SLAs may be violated. Frequent user interactions with the system during SLA negotiation and service executions (which are usually necessary in case of failures), might turn out to be an obstacle for the success of Cloud Computing. Thus, the development of the appropriate strategies for autonomic SLA attainment represents an emerging research issue. Autonomic Computing is one of the candidate technologies for the implementation of SLA attainment strategies. Autonomic systems require high-level guidance from humans and decide, which steps need to be done to keep the system stable. Such systems constantly adapt themselves to changing environmental conditions.

Autonomic systems ([17]) require high-level guidance from humans and decide, which steps need to be done to keep the system stable. Such systems constantly adapt themselves to changing environmental conditions. Similar to biological systems (e.g. human body) autonomic systems maintain their state and adjust operations considering changing components, workload, external conditions, hardware, and software failures. Usually, autonomic systems comprise one or more managed elements e.g. QoS elements.

An important characteristic of an autonomic system is an intelligent closed loop of control. Typically control loops are implemented as MAPE (monitoring, analysis, planning, and execution) functions. The *monitor* collects state information and prepares it for the *analysis*. If deviations to the desired state are discovered during the analysis, the *planner* elaborates plans to alter the system, then these plans are passed to the *executor* for the enactment of the planned changes. As a result the autonomic system should turn to healthy state again. In case of an autonomous service based application the control loop could enable the co-operation of the layers during the analysis (when the overall state of the SBA could be taken into consideration if necessary) and the planner phases (multi layer action plans could be considered). As an opposite the monitoring and execution phases are local to a given layer of the SBA.

Autonomously managed SLA-based resource virtualization (SRV) approach. In our previous work [19] we presented a unified service architecture (SRV) that builds on three main components: agreement negotiation, brokering and service deployment using virtualization (based on grids, service based systems or cloud infrastructures). We suppose that service providers and service consumers meet on demand and usually do not know about the negotiation protocols, document languages or required infrastructure of the potential partners. The general architecture is highlighted in Figure 2.5.

The relevant actors of this architecture are: (1) The *meta-negotiator* is a component that manages Service-level agreements. It mediates between the user and the Meta-Broker, selects appropriate protocols for agreements; negotiates SLA creation, handles fulfillment and violation. (2) The *meta-broker*'s role is to select a service broker that is capable of executing a service with the specified user requirements, and propagate negotiation processes to brokers (by acting as a negotiator). (3) When service brokers receive service requests that could not be met without breaking the negotiated agreements, they use the *automatic service deployment* service that installs a service instance on demand with the help of virtualization and infrastructure as a service cloud computing. Finally (4) is the service that users want to utilize.

In [18] we focus on illustrating how autonomic computing can be applied in the SRV components of the architecture. Figure 2.5 shows the autonomic management interfaces and connections of the components. These management interfaces are used when the MAPE control loop is applied in the autonomic SRV system. With the meta-negotiation component we also present how the propagation of sensed changes could reach the target component with the help of a software actuator (called the VieSLAF framework).

We acquire the MAPE requirements and actions of the SRV architecture through several simple case studies presenting typical fault situations: (1) Negotiation bootstrapping, (2) service mediation,

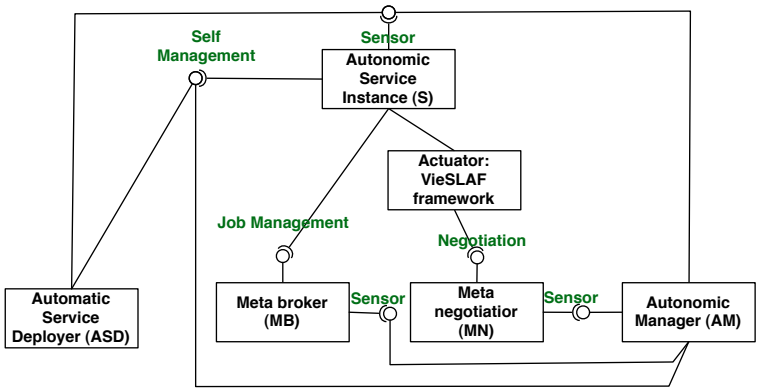

Figure 2.5: Autonomic components in SRV.

Component	Monitoring	Analysis	Planning	Execution
Meta-Negotiation	All candidate services are selected, where negotiation is possible or bootstrapping is required.	The existing knowledge base is queried and potential bootstrapping strategies are found (e.g. in order to bootstrap between WSLA and WS-Agreement).	In case of missing bootstrapping strategies users can define new strategies in a semi-automatic way.	Finally, utilizing the appropriate bootstrapping strategies starts the negotiation during the execution phase.
Meta-Brokering	The state information of all the interconnected brokers are tracked. We also keep track the past performances of the brokers based on feedback.	Query of the various broker availabilities and performance results.	In case of incoming service request the ranking brokers are determined based on the performance data. In case of a broker failure negative feedback is provided.	The broker with the highest rank is selected for service invocation.
Service Deployment	Tracking whether a service instance is healthy, defunct (cannot serve further requests) or overloaded.	Overloaded service instances require either a stronger virtual machine or load balancing. Defunct service instances just need a new host.	A deployment workflow is generated that could include state transfer, notification to brokers, proxy placement in case of decommission.	Deployment workflow executed. If a proxy is placed then it forwards requests to the newly deployed service.

Table 2.2: Summary of autonomic behaviour

(3) broker failures, (4) service-instance initiated deployment. In the paper we describe the adaptation strategies on each fault situation. These strategies are built upon adaptation actions that are separately developed for each component of the SRV architecture. These actions are executed when the architecture tries to avoid breaking the SLA requirements agreed with the upper layers (SCC, BPM) of the service-based application. The resulting MAPE control loop of the architecture is summarized in Table 2.2.

Contribution to Cross-Layer Adaptation and Monitoring

This work presents the autonomous behavior of the service infrastructure layer. This behavior is based on the negotiated SLA requirements with the other layers of the SBA. Therefore this paper contributes to the cross-layer adaptation and monitoring techniques by introducing the basic interface for altering the service infrastructure's behavior. This is a two way interface that is used first to mediate SLA requirements of the service composition and business process management layers, then second it is also used by the service infrastructure layer to present possible agreement violations in order to initiate SLA renegotiation. Finally this paper proposes several adaptation actions that can be used by the upper layers of the SBA to meet higher level goals than the autonomous behavior of the infrastructure level could achieve.

2.2.8 A Non-functional SLA for Cross-layer Monitoring

Title	A View-Based Monitoring For Privacy-Aware Web Services ([12])
Authors	H. Meziane, S. Benbernou, A. K. Zerdali, M. S. Hacid, M. Papazoglou
Short description	This paper addresses the problem of cross layer monitoring of private data in service-based applications, ensuring end-to-end quality of service (QoS) capabilities. The proposed approach is processed towards monitoring the conformity of the privacy agreement stated at the business level that spells out a consumer's privacy rights and how the service provider must handle consumers private information. At the infrastructure level, a Private Data Use Flow (PDUF) model is developed to handle the vulnerabilities. The process is playing a similar role to that of database view in database that provides an abstract representation of a relevant activity. The views can be defined at various abstraction levels to support the activity of the service.

Target cross-layer mechanisms

Integrated monitoring mechanisms	Specification of the non-functional QoS e.g., privacy rules as an SLA at business and infrastructure level (defined as views). The compliance of the privacy rules is checked at the business and the service levels.
Integrated and coordinated adaptation mechanisms	–
Means to identify adaptation needs across layers	By means the views, the monitor detects the threats and the violations of privacy rules specified in SLA across the layers of SBA, identifying the breaking service.
Means to identify adaptation strategies across layers	–

Cross-layer model

Aspect	non-functional characteristics of QoS of service based applications.
Component	Service based system, monitor, database of threats and vulnerabilities.
Key model elements	XML-based language to specify the non-functional QoS as SLA and SQL-like language for querying the system.

Possible extensions

Cross-layer mechanisms	Automatic Adaptation mechanism at composition and infrastructure levels
Cross-layer model	Adaptation strategies across layers.
Layer elements	–

Covered elements at functional layer

	Monitoring Events	**Adaptation Actions**
Business Process Management	non functional business rules	–
Service Composition	–	–
Service Infrastructure	Vulnerabilities, the evolution of the SLA	–

Extended abstract

The huge recent increase in web-based applications carried out on the Internet has accompanied by an exponential amount of data exchanged by the interacting entities through web-services and the growth of consumer awareness of their lack of privacy. Most of the time, web-based service providers require some personal information or financial information from their consumers. Such information may be used for a number of objectives; ranging from regulation access to their online services (authentication, authorization) to billing (accounting), to service maintenance and so on. As the number of inappropriate usage and leakage of personal data is increasing, privacy concerns is becoming one of most important concerns of service requesters, service providers and legislators.

In the beginning, the interest of researchers and practitioners has converged on the functional aspects of those software services and their description. Because of the increasing agreement on the implementation and management of the functional aspects of those services, the interest of researchers is shifting toward the 'non-functional' or quality aspects of web-enabled services including security, privacy, availability, accessibility, etc. Most of these services require the consumer's personal information in one form or another, which makes the service provider in the possession of a large amount of consumer private information along with the accompanying concerns over potential loss of consumer privacy. In fact, as the amount of exchanged information exponentially grows, the number of inappropriate usage and leakage of personal data is increasing, privacy has emerged and is becoming one of the most important and the most crucial concerns and challenging issues. It is today one of the major concerns of users exchanging information through the web, including service requesters, service providers and legislators. Everyone who has purchased anything from the Internet had led the experience of pausing and wondering if is "safe" to enter one's credit card information. Clearly, the more one is exposed to new services on the Internet and the varied personal information that is demanded, by theses services, the more one wonders whether the personal information that ones enters would be kept safe. The search problem faced by Internet users today is not the lack of information from searches, but the challenge is how the web-based applications are more trustworthy to control the private data usage to keep more confidentiality. Such a need, leads to built and manage service-based systems that provide desired end-to-end QoS awareness. Traditionally, access control to any kind of data (e.g. private) has dealt only with authorization decisions on a subject's access to target resources. Obligations are requirements that have to be fulfilled by the subject for allowing access. Conditions are subjects and object-independent environmental requirements that have to be satisfied for access. In today's highly dynamic, distributed environment, obligations and conditions are also crucial decision factors for richer and finer controls on usage of data resources. More precisely, the challenge of private data management is how to do *usage control*, knowing that the private data is already used. In fact, while *access control* aspect of security and privacy is well understood, it is unclear of how to do *usage control*. The need is to assess the health of systems cross the layers. We investigated the self-protecting service management. We are sensitive to build system that anticipates, detects hostile activities dealing with the private data, identifies, and protects against threats from business level to infrastructure layer.

In response to the privacy concerns quoted above, in [4, 5] we proposed a *privacy agreement* model at the business level that spells out a set of requirements related to consumer's privacy rights in terms of how service provider must handle privacy information. The properties and private requirements can be

checked at a design time prior to execution, however, the monitoring cross layers of the requirements at run-time has strong motivations since those properties can be violated at run time. Thus, checking at run-time the compliance of the requirements defined in the privacy agreement is a challenging issue, see Figure 2.6 for the proposed architecture.

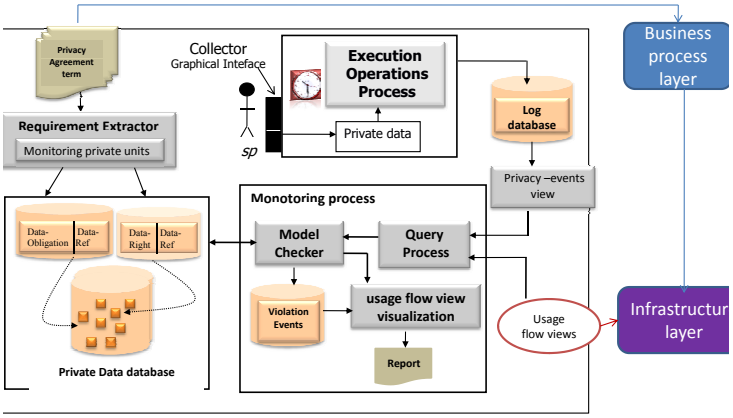

Figure 2.6: The cross-layer architecture of Private SLA

That issue must be properly addressed otherwise it could lead to agreement breaches and to lower service quality. Indeed, the private data use flow must be observed which means monitoring the behavior of the privacy agreement. From the results of the observations, analysis can be done to come up to an understanding, why the non-compliance took place and what remedy will be provided enhancing the privacy agreement.
The common approach developed to support requirements monitoring at run-time assumes that the system must identify the set of the requirements to be monitored. In fact, as part of the privacy agreement model, the set of privacy requirements to be monitored are needed from which *monitoring private units* are extracted and their occurrences at run-time would imply the violation of the requirements. Besides the functional properties (e.g. operations of the service), the time-related aspects are relevant in the setting of the privacy agreement. In addition, the non-compliance or failing to uphold the privacy requirements are manifested in terms of vulnerabilities must be identified. The approach features a model based on state machine that is supported by *abstractions* and *artifacts* allowing the run-time management. It's a four-phases approach:

- Identifying the user and provider needs at business level in order to protect the personal information.
- Formalizing the needs as a contract. The privacy policy and data subject preferences are defined together as one element called Privacy-agreement an extension of WS-Agreement, which represents a contract between two parties, the service customer and the service provider within a validity. We provide abstractions defining the expressiveness required for the privacy model, such as rights and obligations. The defined contract is considered as the output of the business layer.
- Providing a formal model for monitoring and controlling the privacy agreement. For that purpose, we introduce the private data use flow (PDUF) mechanism. We propose to express the PDUF as a state machine because of its formal semantics, well suited to describe the activation of different clauses of the privacy agreement. It is an effective way to identify the privacy vulnerabilities,

where service compliance to the privacy regulations may be compromised. The semantic of the state machine is to define all the triggered operations involving a private data from the activation of the agreement (initial state) to the end of the agreement (final state). In Figure2.6 is presented the current cross-layer architecture for Private SLA monitoring.

- Querying the private usage flow. We introduce the notion of the *usage flow view* while composing service (at the SCC layer). The *usage flow view* (playing the same role to that of *database* view in databases) provides views from the abstract PDUF (from business level) corresponding to the triggering clauses of the privacy agreement "output of business layer". In Figure 2.7 is depicted the cross-layers mapping.

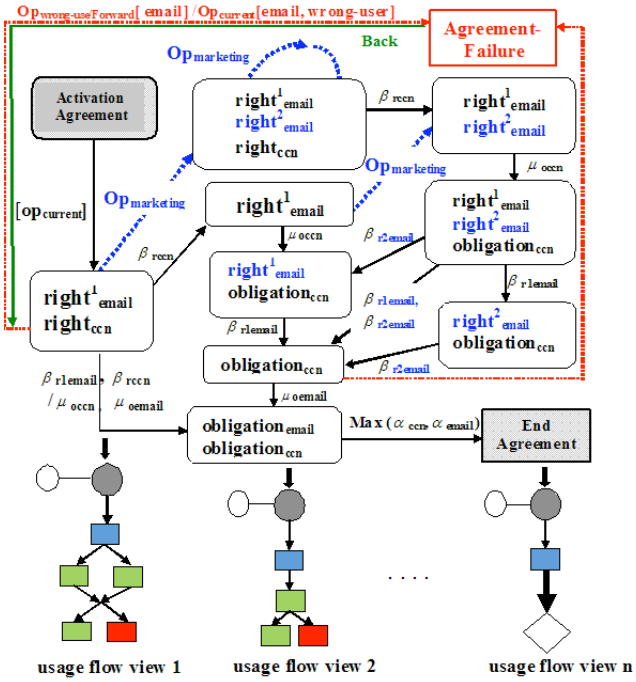

Figure 2.7: PDUF: The global usage flow views

Contributions to Cross-layer Adaptation and Monitoring

As stated in our previous deliverable PO-JRA-1.2.3, the part of the deliverable (at the end of the report is a paper to appear at 26th IEEE International Conference on Data Engineering, ICDE' 2010 conference) [12] aims at proposing a framework for contract-based monitoring with respect the non functional QoS of the customer's goal i.e. privacy. We discuss the concept of "privacy agreement" as a possible approach for monitoring SLA cross-layers in SBA. The monitor component (see Figure 2.6) supports cross layers monitoring to capture the threats and vulnerabilities at the infrastructure level for the services involved in the SBA. Moreover, it also supports monitoring the privacy in the contract at the behavioral level in the SBA.

2.3 Summary and analysis

In this chapter, we presented the initial integration approaches for cross-layer adaptation and monitoring. We have shown the diverse ways the problems during integration could be handled. In this section we aim to identify the well-investigated directions and the gaps of our current research based on the paper summary sections presented before.

	Summarized research papers							
⇓ *Research areas* ⇓	[23]	[20]	[15]	[9]	[1]	[2]	[18]	[12]
Integrated monitoring mechanisms	-	+	+	+	+	+	-	+
Integrated and coordinated adaptation mechanisms	-	-	-	-	+	+	+	-
Means to identify adaptation needs across layers	+	+	+	+	+	+	+	+
Means to identify adaptation strategies across layers	+	-	+	+	-	+	-	-
BPM Monitoring events	+	+	+	-	generic	generic	-	+
SCC Monitoring events	+	-	+	+	generic	generic	-	-
SI Monitoring events	+	+	+	-	generic	generic	+	+
BPM Adaptation Actions	generic	-	generic	-	generic	generic	-	-
SCC Adaptation Actions	generic	-	generic	+	generic	generic	-	-
SI Adaptation Actions	generic	-	generic	-	generic	generic	+	-

Table 2.3: Summary of research achievements towards the integration of cross-layer adaptation and monitoring principles, techniques and methodologies

Analysis and comparison is based on the findings presented in Table 2.3. First of all we analyze the research results from the perspective of "*Target cross layer mechanisms*". The most targeted topic is the identification of adaptation needs across layers. All research papers present ideas on this topic here we include the examples of specification on the partial replacement of the service based system and the user specified probes triggering the execution of adaptation strategies. From the summary tables of each previous section we also identified that the issue of *integrated adaptation and coordination mechanisms* is barely targeted, and it should be later discussed in the coming deliverables of the work-package.

If we consider our research from the point of "*covered elements at the functional layer*", then it can be seen that the integration in the most cases is supported by the service infrastructure layer (one notable exception is [9]). First this is simply the result of the availability of the most fine grained information for monitoring in the SBA, that results (a) the use of infrastructure layer monitoring systems to collect information for the higher layers, or (b) the higher layers aggregate the monitored values collected by the infrastructure layer. Second most adaptation strategies also require the support of the infrastructure layer at least when they involve changes in the infrastructure.

The topic of "*integrated and coordinated adaptation mechanisms*" is less covered in this current deliverable; we have focused on mainly the integration of monitoring systems on the different layers of the SBA. This is done purposefully here because integrated adaptation requires the global view of the SBA for informed decision-making. This global view however can hardly be provided by a fragmented monitoring system; therefore we have provided six research papers ([1,2,12,15,20,23]) that are focusing on building the baseline for the integration of adaptation by investigating integration possibilities among the monitoring systems of the different layers. Adaptation is not neglected usually (except two articles [12, 20]), however currently our research is still focused on the generic concepts behind the integration of adaptation systems between layers (e.g. providing languages to express adaptation strategies or using

use case scenarios that provide layer independent requirements for adaptation). As it can be seen from this and the previous deliverable we already have support for (i) identification of adaptation needs, (ii) identification of adaptation strategies, and (iii) several adaptation mechanisms. Therefore in the next deliverable we will aim to integrate concrete adaptation actions with the already available tools to provide the missing elements from these generic adaptation systems.

Chapter 3

Outlook and conclusions

This chapter aims to summarize the research directions followed by the project members in order to achieve the research tasks of the work-package. In this current deliverable we have presented the ways of consolidating our joint research in order to achieve the initial cross-layer integration of the adaptation and monitoring principles, techniques and methodologies. This work has been presented through several joint research papers. In section 2.1 we have defined a comprehensible methodology in order to present the coherency of the research direction of these co-authored articles. However, this methodology was also the key factor for the identification of the research gaps that should be leading concerns of future deliverables. The identified problems are highlighted in the following paragraphs.

Towards comprehensive integrated adaptation and monitoring principles

First in section 2.2.1 and 2.2.2 we have started with the investigation of replacing service instances in compositions that can be supported by runtime service discovery. As a result we have identified adaptation requirements of the service composition layer that can be propagated towards the service infrastructure layer. These requirements allow research on the adaptation actions to be performed in the service infrastructure layer. The resulting actions of this research will form the base to define the adaptation strategies applied after service discovery identifies possible replacements of a service instance. Extending the service replacement policies to support both the business process model and the service composition layer can strengthen the cross-layer behavior of this contribution.

Later on in section 2.2.7 we propose the autonomous behavior of the basic services in the service infrastructure layer. This behavior is guided by the service level agreements between the composition and the infrastructure layers. We propose to fire monitoring events for the SCC and BPM layers in case the infrastructure layer cannot operate autonomously without violating the agreements. Future research will focus on the issues about firing these monitoring events before actually violating the agreements. This research will require fine-grained agreement description that allows the cooperation between JRA-1.2 and JRA-1.3.

As described in section 2.2.8 service level agreements could also include non-functional properties like privacy requirements. We introduce the monitoring of the violation of these non-functional properties. In section 2.2.3 we also discuss monitoring of the different SBA layer aspects. These aspects cover the monitoring of KPIs in business process management layer, PPMs at the service composition and coordination layer and finally QoS metrics of the service infrastructure layer. Based on these monitoring results we plan to develop adaptation strategies in order to alter future service compositions and business processes. These strategies will involve automatic adaptation mechanisms at composition and infrastructure layers. The adaptation actions involved in the strategies will include ad-hoc process modification, service-replacement, service re-composition and infrastructure reconfiguration.

Finally in sections 2.2.4, 2.2.5 and 2.2.6 we introduce the BPM aspects of the adaptation and monitoring techniques. During design time we define domain assumptions that can be verified and monitored

during the execution of the SBA. Currently we support the monitoring of protocol deviations and temporal property violations in service compositions. We also present a language that can express the violation scenarios that could raise monitoring events. Then we present a language to describe adaptation actions. In our future plans we will target our research towards new types of analysis techniques to identify new kinds of domain assumptions that monitor the different layers and aspects of the SBA.

Towards integration of predictive monitoring and proactive adaptation models

In this section we shortly discuss the research issues and goals that initiate the work on the deliverable CD-JRA-1.2.6 titled "*Comprehensive, integrated principles, techniques and methodologies for context- and HCI-aware monitoring of SBAs and for the detection of unexpected situations*".

In current implementations of adaptive service-based applications (SBAs), monitoring events trigger the adaptation of an application after a change or a deviation has occurred. Yet, such reactive adaptations have several drawbacks (cf. [10]): executing faulty services, increasing execution time due to the execution of adaptation activities, and late arrival of events that trigger adaptation which would make the adaptation of the system not possible anymore. 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 SBA and before such a deviation can lead to aforementioned problems. However, proactive adaptation decisions should be taken in an informed way (e.g., by building confidence in the predicted future failures), to avoid unnecessary proactive adaptations that could be costly and/or faulty.

The key factor of proactive adaptation is the prediction of the future quality (and functionality) of a service-based application. Existing approaches ([6, 8, 21, 22, 28–30]) use data mining techniques to predict system parameters, faults and QoS attributes. However these approaches are typically dependent on the availability of historical and training data and, hence, not applicable in the case of frequently changing and previously unseen services which may invalidate past monitoring data due to those changes.

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 ([7, 31]) have highlighted the importance of online testing for SBAs, there have been no concrete techniques to exploit testing results for (self-) adaptation.

Finally, statistical testing ([3, 13, 24, 27, 33]) could be used for determining feasible sub-sets of test cases while still maintaining adequate test coverage and also to predict the reliability of the system under test. However, statistical testing requires a very large number of test cases to produce statistically sound data, therefore it poses a significant burden for its applicability in the SBA context.

Bibliography

[1] Luciano Baresi and Sam Guinea. Self-supervising bpel processes. Technical report, Dipartimento di Elettronica e Informazione, Politecnico di Milano, November 2009.

[2] Luciano Baresi, Sam Guinea, and Liliana Pasquale. Integrated and composable supervision of bpel processes. In Athman Bouguettaya, Ingolf Krüger, and Tiziana Margaria, editors, *ICSOC*, volume 5364 of *Lecture Notes in Computer Science*, pages 614–619, 2008.

[3] T. Bauer, F. Bohr, D. Landmann, T. Beletski, R. Eschbach, and J. Poore. From requirements to statistical testing of embedded systems. In *SEAS '07: Proceedings of the 4th International Workshop on Software Engineering for Automotive Systems*, Washington, DC, USA, 2007. IEEE Computer Society.

[4] Salima Benbernou, Hassina Meziane, and Mohand-Said Hacid. Run-time monitoring for privacy-agreement compliance. In *ICSOC*, pages 353–364, 2007.

[5] Salima Benbernou, Hassina Meziane, Yin Hua Li, and Mohand-Said Hacid. A privacy agreement model for web services. In *IEEE SCC*, pages 196–203, 2007.

[6] S. Casolari, M. Andreolini, and M Colajanni. Runtime prediction models for web-based system resources. In *Proc. IEEE International Symposium on Modeling, Analysis and Simulation of Computers and Telecommunication Systems MASCOTS 2008*, pages 1–8, 2008.

[7] P. Deussen, G. Din, and I. Schieferdecker. A ttcn-3 based online test and validation platform for internet services. In *Proceedings of the 6th International Symposium on Autonomous Decentralized Systems (ISADS)*, pages 177–184, 2003.

[8] S. Garg, A. van Moorsel, K. Vaidyanathan, and K. Trivedi. A methodology for detection and estimation of software aging. In *Proceedings of the 9th International Symposium on Software Reliability Engineering*, pages 282–292, Paderborn, Germany, November 1998.

[9] A. Gehlert, A. Bucchiarone, R. Kazhamiakin, A. Metzger, M. Pistore, and K. 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*, Sierre, Switzerland, March 2010.

[10] J. Hielscher, R. Kazhamiakin, A. Metzger, and M. 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*, LNCS. Springer, 2008.

[11] Julia Hielscher, Andreas Metzger, and Raman Kazhamiakin, editors. *Taxonomy of Adaptation Principles and Mechanisms*. S-Cube project deliverable, March 2009. S-Cube project deliverable: CD-JRA-1.2.2. http://www.s-cube-network.eu/achievements-results/s-cube-deliverables.

[12] H.Meziane, S.Benbernou, A.K.Zerdali, M.S.Hacid, and M.Papazoglou. A view-based monitoring for privacy-aware web services duplicate invoices detection. In *to appear in 26th International Conference on Data Engineering Conference, ICDE*, 2010.

[13] C. Kallepalli and J. Tian. Measuring and modeling usage and reliability for statistical web testing. *IEEE Trans. Softw. Eng.*, 2001.

[14] R. Kazhamiakin, M. Pistore, and Asli Zengin. Cross-layer Adaptation and Monitoring of Service-Based Applications. In *Proc. Of 2nd Intl. workshop on Monitoring, Adaptation and Beyond (MONA+)*, Collocated with ICSOC/ServiceWave'09, 2009.

[15] R. Kazhamiakin, B. Wetzstein, D. Karastoyanova, M. Pistore, and F. Leymann. Adaptation of service-based applications based on process quality factor analysis. In *Proc. Of 2nd Intl. workshop on Monitoring, Adaptation and Beyond (MONA+)*, Collocated with ICSOC/ServiceWave'09, 2009.

[16] Raman Kazhamiakin. *Baseline of Adaptation and Monitoring Principles, Techniques, and Methodologies across Functional SBA Layers*. S-Cube project deliverable, July 2009. S-Cube project deliverable: PO-JRA-1.2.3. http://www.s-cube-network.eu/achievements-results/s-cube-deliverables.

[17] Jeffrey O. Kephart and David M. Chess. The vision of autonomic computing. *Computer*, 36(1):41–50, 2003.

[18] Attila Kertesz, Gabor Kecskemeti, and Ivona Brandic. Autonomic resource virtualization in cloud-like environments. Technical Report TUV-1841-2009-04, Technical University of Vienna Information Systems Institute Distributed Systems Group, Argentinierstr. 8/184-1; A-1040 Vienna, Austria, December 2009.

[19] Attila Kertesz, Gabor Kecskemeti, and Ivona Brandic. An sla-based resource virtualization approach for on-demand service provision. In *Proceedings of 3rd International Workshop on Virtualization Technologies in Distributed Computing*. ACM, June 2009.

[20] George Spanoudakis Khaled Mahbub and Andrea Zisman. A monitoring approach for runtime service discovery. under preparation.

[21] I. Lee. *Software Dependability in the Operational Phase*. PhD thesis, Department of Electrical and Computer Engineering, University of Illinois, Urbana–Champaign, IL, 1995.

[22] D. J. Lilja. *Measuring computer performance. A practitioner's guide*. Cambridge University Press, 2000.

[23] Khaled Mahbub and Andrea Zisman. Replacement policies for service-based systems. In *2nd Workshop on Monitoring, Adaptation and Beyond (MONA+)*, 2009.

[24] J. Poore and C. Trammell. Engineering practices for statistical testing. *Crosstalk: The Journal of Defense Software Engineering*, 11:24–28, 1998.

[25] M. Proctor, M. Neale, P. Lin, and M. Frandsen. Drools Documentation. *Available on: http://labs. jboss. com/file-access/default/members/jbossrules/freezone/docs/3.0*, 1, 2006.

[26] C Tinelli. A dpll-based calculus for ground satisfiability modulo theories. In *Proceedings of JELIA-02*, volume 2424 of LNAI, pages 308–319, 2002.

[27] C. Trammell. Quantifying the reliability of software: statistical testing based on a usage model. In *ISESS '95: Proceedings of the 2nd IEEE Software Engineering Standards Symposium*, Washington, DC, USA, 1995. IEEE Computer Society.

[28] K. Vaidyanathan and K. S. Trivedi. A measurement-based model for estimation of resource exhaustion in operational software systems. In *Proceedings of the Tenth IEEE International Symposium on Software Reliability Engineering*, pages 84 –93, Boca Raton, FL, November 1999.

[29] W.M.P. van der Aalst and M. Pesic. *Test and Analysis of Web Services*, chapter Specifying and Monitoring Service Flows: Making Web Services Process-Aware, pages 11–55. Springer, 2007.

[30] R. Vilalta, C.V. Apte, J.L. Hellerstein, S. Ma, and S.M Weiss. Predictive algorithms in the management of computer systems. IBM, 2002.

[31] Q. Wang, L. Quan, and F. Ying. Online testing of web-based applications. In *Proceedings of the 28th Annual International Computer Software and Applications Conference (COMPSAC)*, pages 166–169, 2004.

[32] B. Wetzstein, P. Leitner, F. Rosenberg, I. Brandic, S. Dustdar, and F. Leymann. Monitoring and analyzing influential factors of business process performance. In *Proceedings of EDOC 2009*, Auckland, New Zealand, 2009.

[33] J.A. Whittaker and M.G. Thomason. A markov chain model for statistical software testing. *IEEE Trans. Softw. Eng.*, 20:812–824, 1994.

[34] I.H. Witten and E. Frank. *Data Mining: Practical Machine Learning Tools and Techniques*. Morgan Kaufmann, 2005.