Grant Agreement N° 215483

Title:	*Comprehensive, integrated adaptation and monitoring principles, techniques and methodologies across functional SBA layers considering context and HCI*
Author:	*UNIDUE, TILBURG, CITY,FBK, POLIMI, SZTAKI, UCBL, USTUTT*
Editor:	*Deliverable leader (FBK)*
Reviewers:	*Michael Parkin (TILBURG)*
	Dragan Ivanovic (UPM)
	Andreas Metzger (UNIDUE)
Identifier:	*CD-JRA-1.2.5*
Type:	*Deliverable*
Version:	*1.0*
Date:	*March 16, 2011*
Status:	*Final*
Class:	*External*

Management Summary

This deliverable presents the research results obtained within the scope of workpackage WP-JRA-1.2 towards the comprehensive integrated adaptation and monitoring principles, techniques and methodologies across functional layers, proactive and context-aware adaptation. To bring the results together and to provide a coherent view on the different techniques using common realizing architecture a set of integration scenario has been defined and elaborated. Based on the integrated model defined in previous documents, these scenarios aim to define the reference architecture and approach relating various contributions, as well as to define the concrete interfaces and dependencies between them. The scenario presented in this deliverable refer to the some of the key research problems studied in the scope of the workpackage and the project, namely cross-layer quality driven monitoring and adaptation, proactive adaptation and context-aware monitoring and adaptation.

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).

Grant Agreement N° 215483

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	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

The S-CUBE Deliverable Series

Vision and Objectives of S-CUBE

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

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

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

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

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

Foreword

The deliverable CD-JRA-1.2.5 "Comprehensive, integrated adaptation and monitoring principles, techniques and methodologies across functional SBA layers considering context and HCI" aims at addressing the following goals:

- To further develop and consolidate the comprehensive and integrated SBA adaptation and monitoring framework focusing on the problem of proactive adaptation at the different application layers;
- To provide, through the definition of three integration scenarios, a common basis for the different fragmented and partial approaches developed within the project and to open up further integration of different principles, techniques and mechanisms through common interfaces and architecture mechanisms.
- 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.

Table of Contents

1 Introduction

The Monitoring and Adaptation of Service-based Applications (SBAs) remains as one of the most challenging research problems both for the enterprise SOA and Internet of Services [23]. Dealing with the intrinsic structural and conceptual complexity of such systems and taking into account the openness and dynamicity of the environments where such applications operate, the challenges revealed within the area of SBA adaptation and monitoring concern the integration and aggregation of different approaches across functional SBA layers, availability of proactive adaptations, and flexibility with respect to different types of context and contextual changes.

In the previous work within workpackage WP-JRA-1.2 the problem of integration of adaptation and monitoring approaches, methodologies, and mechanisms available at different functional SBA layers has already been studied. Starting from the baseline for cross-layer adaptation and monitoring (Deliverable PO-JRA-1.2.3), where the requirements for such integration have been presented, the initial integration of the existing and new research results has been provided (Deliverable CD-JRA-1.2.4). This integration demonstrated the mapping of available solutions onto the generic SBA stack and provided a set of cross-layer integrating approaches both at the methodological and the realization level. However, the concrete integration of the wide range of specific approaches, concrete interfaces and relations has not yet been provided. We believe that such a model of concrete integration, as well as the definition of the relations and dependencies between those approaches, plays a fundamental role in construction of a comprehensive, integrated adaptation and monitoring framework being developed in JRA-1.2.

Together with the problem of integration of more traditional "reactive" approaches, there is a need to study the problem of proactivity in adaptation at different layers and settings. The ability to anticipate the need for adaptation enables an SBA to prevent the failures and deviations that are critical in the open world of SBAs. Similarly, there is a need to consider the context of the SBA in a holistic manner and to provide integrated approaches over context-driven SBA monitoring and adaptation.

In this deliverable we further develop a comprehensive and integrated SBA adaptation and monitoring framework, that focuses on the above problems. In order to accomplish this, we introduce the integration scenarios that follow common architecture but address specific problems, namely cross-layer SBA quality, proactive SBA adaptation, and context-driven adjustment of the monitoring and adaptation specification mechanisms. The goal of the scenarios is to provide a common infrastructure for the adaptation and monitoring problem in hand, providing a common basis for the different fragmented and partial approaches developed within the project. Starting from this common basis the integration scenarios open up further integration of different principles, techniques and mechanisms through common interfaces and architecture mechanisms.

1.1 Contribution to WP Challenges and Objectives

With respect to the research challenges from the Integrated Research Framework identified in the workpackage WP-JRA-1.2, this deliverable provides several contributions. Each of the integration scenarios identified and described in the document focuses on one of the following challenges:

- **Comprehensive and integrated adaptation and monitoring principles, techniques, and methodologies**. Scenario 1 primarily focuses on the problem of integrating the adaptation and monitoring aspects across different layers having the quality model of the SBA as the key driving aspect. Specifically, a common and generic methodology based on data mining approaches is used as a backbone of the scenario, to which different approaches for monitoring and adaptation at different layers are attached.

- **Proactive Adaptation and Predictive Monitoring**. Scenario 2 addresses this challenge by providing an approach based on explicit modelling and monitoring of various assumptions. The violations of those assumptions are then used to validate the SBA requirements and to trigger various forms of adaptation.

- **Context- and HCI-aware SBA monitoring and adaptation**. This challenge is directly addressed by tScenario 3, where initial approaches take into account context changes while performing adaptation and monitoring.

The definition of integration scenarios contributes to the first challenge as they provide a reference model and a set of interfaces that will be further exploited for the definition of more flexible holistic framework, where all the aspects (cross-layer integration, context, and proactiveness) are present.

1.2 Relations to other WPs and Integrated Research Framework

The results presented in this deliverable have clear relationships with the other S-Cube workpackages and with the overall Integrated Research Framework. As a matter of fact, some of the research papers represent joint results of several workpackages; the integration scenarios cover also variety of aspects in these regards. Apart from the natural relations with the workpackages that deal with functional SBA layers (JRA-2.1, JRA-2.2, and JRA-2.3), the presented results are also related to S-Cube workpackages as follows:

- JRA-1.1. Research results regarding the model-driven management of services, adaptation driven by service evolution, modeling of SBA assumptions, and context-based adaptation are tightly coupled with the engineering and design workpackage. In order to underle the relation between scenarios a mapping of the activities to the SBA life-cycle is presented. In this mapping, we explicitly characterize the necessary design-time activities and steps relevant for the specific constituent elements.

- JRA-1.3. Scenarios to target proactive adaptation using assumptions exploits the run-time QA approaches to check the violation of requirements in order to identify the needs for adaptation. Also in case of quality-driven monitoring and adaptation the reference quality model of JRA-1.3 is used as the means to represent different quality factors across functional layers and their relations.

We also remark that one of the goals of this deliverable is to directly contribute to the S-Cube Integrated Research Framework (IRF) developed in the workpackage IA-3.1. The integration scenarios defined here extend the IRF by providing the model and instantiation of the cross-cutting integration of research results. These scenarios contribute to different views of the IRF (namely the architectural and life-cycle views) and define common interfaces between different components across layers. Moreover these integration scenarios will be used to define a set of validation scenarios in the scope of IA-3.2 with the aim of analyze and validate the developed monitoring and adaptation techniques and approaches.

1.3 Structure of this Deliverable

This document presents the results of integration of SBA monitoring and adaptation approaches based on the integration scenarios. Section 2 provides the unified model for integration scenarios, in particular, defines its structural architecture and the way different research results may contribute to the scenario in the form of summarizing template.

Sections 3, 4, and 5 represent three integration scenarios that focus on quality-driven cross-layer monitoring and adaptation, assumption-based proactive adaptation, and context-based adaptation and monitoring, respectively. For each of those scenarios we define:

- The shared approach used by the scenario to resolve the problem within and across different functional layers;

- The refined architecture, specific components and their relations that implements the approach as well as the description of the specific activities across SBA life-cycle;

- The contributions of the partners onto the scenario, specifying the role, the relevant architecture components and the life-cycle activities.

We remark that the deliverable provides only a structured view on the individual contributions from the viewpoint of the integration scenarios; the papers themselves are attached to the deliverable and are listed in annex A of the document.

Finally, the deliverable concludes with the discussions regarding the open points, gaps, and future steps with respect to the objectives of the workpackage.

2 Integration Scenarios

The goal of this Section is to define a model of integration for the fragmented and specific technical results obtained by the partners in the area of SBA adaptation and monitoring within specific functional layers and addressing the specific aspects. This model of integration, referred to as an *integration scenario*, aims at refining the overall adaptation and monitoring framework presented in previous deliverables in a three specific ways.

First, the scenarios describe specific architecture and various modules that, operating at and across different functional SBA layers, realize an instantiation of a particular SBA monitoring and adaptation procedure. In other words, scenarios define the integration of different specific solutions that address some specific problem of SBA adaptation (e.g., adaptation due to deviation of the SBA quality) in an integrated, cross-layer manner (e.g., across different layers and/or different quality dimensions).

Second, the scenarios focus on a particular class of the applications and a particular problem. In this way, the scenarios clearly shape the input models and specifications that facilitate the definition of the interfaces and relations between different components and their realizing techniques.

Third, the scenarios combine the otherwise isolated solutions and techniques. Operating through common interfaces, these techniques complement each other in order to provide a more complete solution for the described instance of SBA adaptation and monitoring.

Introducing and defining such integration scenarios achieves the following goals.

- The scenarios allow us to overcome the problem of fragmentation and isolation of different solutions provided individually at different technology layers. In many cases, the independent solutions are insufficient or incompatible since the problem of holistic SBA monitoring and adaptation has not been investigated. To solve the problem, the solutions have to be appropriately extended in order to support the mutual requirements and inputs. In this setting, the scenarios provide the means for expressing such relations and provide concrete interfaces between the elements.

- The scenarios aim at covering a concrete adaptation problem or application domain in an end-to-end manner. That is, the scenario covers all the important aspects and layers of the application, thus enabling us to clearly identify, where the existing approaches may be exploited and what the critical gaps are. Based on the identified gaps, the required extensions and new solutions will be developed.

- As the scenarios focus on particular problem domains and aspects, they provide a vertically refined vision of such problems and aspects. Within the scenarios the concrete interfaces, components, and their implementations are defined for the purpose of those aspects. However, those aspects potentially complement each other within the same application. Therefore, the underlying architectures and components may be interleaved providing a basis for the definition and development of more flexible and holistic SBA adaptation and monitoring platform.

- The use of a common architecture, definition of the relations, and the use of a common application type and domains, enables the end-to-end validation of different contributions. We remark that the scenarios presented in this document do not aim at covering all the possible problems and applications. Here we define the initial set of (sufficiently generic) scenarios that have been jointly identified by the partners and that are relevant from the point of view of the research agenda of the workpackage JRA-1.2. This initial list will be eventually extended, as well as the scenarios defined here will be further refined in the upcoming phases of the S-Cube project.

We remark that the reference architecture and life-cycle refined by the three scenarios exploit the concepts and elements of the Integrated Research Framework (e.g. S-Cube Life Cycle view). In particular, the usage of the integrations scenario to validate monitoring and adaptation research results in the upcoming year will be the facilitated by the fact that the model of the integration scenarios follows the approach used in IA-3.2 to define validation scenarios.

In the following sections we will define the integration scenarios and characterize the architecture and the key concepts behind the model of the scenario. We will also demonstrate how the specific research results may be accommodated within the scenarios and provide the characterization of the scenarios in these regards. This generic model will be then instantiated in three different scenarios that incorporate and relate contributions from the partners that has been studied through the work on the deliverable.

2.1 Reference Scenario Architecture

Based on the reference model of the conceptual SBA adaptation and monitoring framework defined in Deliverable CD-JRA-1.2.2 "*Taxonomy of Adaptation Principles and Mechanisms*" [1], we present a generic architecture for a complete monitoring and adaptation solution that will be instantiated by different scenarios.

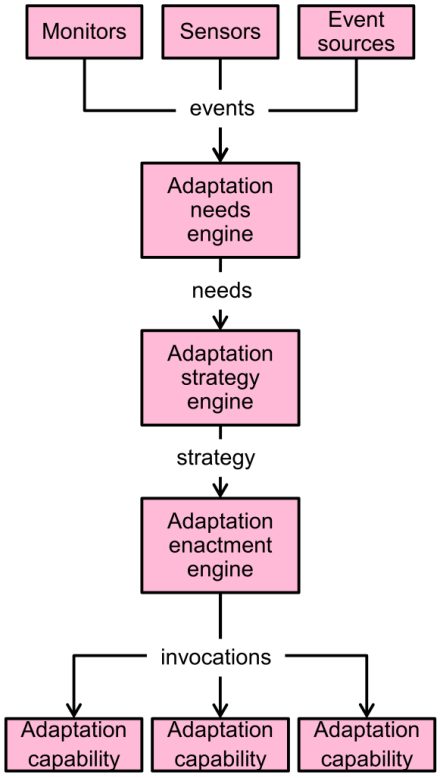

Figure 1 Reference A&M architecture

The architecture shown in Figure 1 defines a set of component specifications, which will be replaced by the concrete implementations (or their combinations), and which defines a common flow of information between those components based on a set of interfaces. The differentiation of the presented components provides only a basis for the integration; concrete interfaces and implementations are defined for each scenario separately.

The components of the reference A&M architecture from Figure 1 are:

- *Monitors, sensors and event sources* are mechanisms that detect, aggregate and correlate different types of SBA events within and across functional SBA layers. Different types of *events* are used to trigger the analysis of necessity of doing adaptation and its nature. The

events here are interpreted in generic way; they may refer to "traditional" events occurring during execution, or to any form of changes in the application, its specification, environment etc.

- *Adaptation needs engine* is a component that is responsible for defining whether the SBA adaptation is needed. Depending on a specific scenario, this may involve considering human participation, using complex analysis techniques, etc. The *adaptation needs* triggered in this way identify the nature of the problem as an input to the adaptation strategy selection. Note that such an engine may be distributed across the layers: an event occurring at one layer (e.g., business-level change) may require the adaptations to be performed at other layers (e.g., adjustment of the realizing service orchestration).

- *Adaptation strategy engine* is a component that selects a specific strategy to be applied to meet the particular adaptation need. This is a technique, or set of techniques, that may be exploited for the adaptation decision making and may include basic rule-based approaches as well as sophisticated reasoning tools. The outcome of this component is the *strategy* defining concrete adaptation actions to be exploited on top of the available adaptation capabilities.

- *Adaptation enactment engine* is a component that enacts the adaptation actions identified by the strategy by means of invoking individual adaptation capabilities. The engine usually includes a number of specific modules operating at different layers, or targeting a specific type of adaptation.

- *Adaptation capabilities* are specific building blocks for constructing different types of complex adaptations. Again, the set of the capabilities and their interfaces is specific for each scenario; they may represent different aspects and functional layers.

The role of the architecture instantiated within the scenario is twofold. On one hand it defines the placement of the specific contributing techniques and approaches, and the relations and interfaces between them. On the other hand it helps to identify the gaps in its realization that require further research activities.

2.2 Scenario Description

Apart from the architecture instantiation, the complete integration scenario model consists of a set of elements that characterize the scenarios from different perspectives. This includes the characterization of the applications (and possibly the relevant application domains) for which the scenario is defined, the characterization of the adaptation and monitoring problem the scenario addresses and the corresponding reference model, the description of the overall approach that in the scenario drives different monitoring and adaptation activities, and the involvement of different solutions in those activities. The latter aspect is captured both statically, i.e., by defining the architectural components where the concrete technique is involved, and dynamically, i.e., by representing the different relevant steps to be carried out throughout the reference SBA life-cycle [2].

2.2.1 Application model

The application model characterizes, at a high level of abstraction, prospective applications for which the integrated solution of the scenario is relevant. This model defines the key components of the application and their characteristics that are relevant from the adaptation point of view. We remark that the application model is defined on top of the elements of different functional layers. The examples of these elements are also represented in [3,4]. A typical model of the SBA refers to complex long-running business processes associated to high-level business metrics (i.e., the model of BPM layer), implemented on top of BPEL-based compositions of services and process fragments (the model of SCC layer), that run on top of complex cloud-based infrastructures (the model of SI layer). The application model may include application domain-specific notions depending on the level of abstraction.

2.2.2 Adaptation and Monitoring Problem

The adaptation and monitoring problem is used to drive the definition of the integration scenario. It characterizes the requirements the monitoring and adaptation activities should address. These requirements are expressed in terms of the application model; they define the expected behaviour in terms of the application model properties (e.g., expected reactions in cases of deviation from the application quality model).

The description of adaptation and monitoring problem is based on the taxonomy of relevant A&M characteristics presented in [1]. Besides, this definition refines those generic concepts and defines concrete properties to be addressed by the integration scenario. We would like to stress the fact that the problem specified for the scenario aim at covering the SBA in an integrated manner, i.e., to take into account the aspects of different functional SBA layers.

2.2.3 Overall Approach

Each scenario defines also the common approach exploited within the scenario to carry out the corresponding problem. The approach defines the key idea used to implement and relate the components and their outputs to each other. In other words, the approach relates different research contributions making them constituent parts of the complete solution to cover all the phases from monitoring to adaptation enactment across different layers.

Besides presenting the reference idea, this part of the scenario description provides the holistic view on the target integrated solution. In particular, it defines:

- A *refined architecture* of the integrated solution, where the specific elements and contributions are properly placed;
- A *refined SBA life-cycle* for the integrated solution, where the specific activities and steps relevant for the constituent elements are represented.

2.2.4 Individual Contributions

Finally, the scenario is accomplished with the descriptions of the specific contributions. To provide a more systematic vision, these contributions (each defined in a corresponding research paper) are represented with a short template that summarizes the role and the participation of the result into the scenario.

Title	Title of the research work
Authors	Authors of the research work
Type	Type of contribution (Methodology / Model / Technique / Application / Experimental Evaluation / ..)
Short Description	Brief description of the contribution with respect to the research problem presented in the integration scenario

Adaptation and Monitoring Problem

Contribution to the adaptation problem	Specific adaptation problem addressed by the approach (if any)
Contribution to the monitoring problem	Specific monitoring problem addressed by the approach (if any)

Refined Architecture

Architecture elements	Specify and refine the relevant components of the architecture
Requirements/Constraints	Specify any important requirement/constraint of the proposed approach on the architecture components

Refined SBA life-cycle

Life-cycle activities	Specify what are the steps of the life-cycle covered by the approach

Open problems and possible extensions

Open problems and extensions	Specify what are the open research questions and limits of the proposed approach and describe some future extensions

3 Quality-driven Multilayer SBA Monitoring and Adaptation

3.1 Application Model

The SBA reference model in this scenario primarily targets applications implemented as long-running business processes and workflows. Such processes are orchestrations of various software-based or human-based services, made available through standardized interfaces and protocols. The compositions themselves are defined as executable BPEL processes that interact with the services defined using WSDL specifications. Both intra-enterprise and external services may take part of such compositions. Internal services are realized on top of sophisticated service provisioning architectures, such as cloud-based and grid based distributed systems.

A key element of the application model in this scenario is the set of quality characteristics that the different functional layers of the application should satisfy. At the BPM layers, for instance, these characteristics have a form of Key Performance Indicators (KPIs) that state expectations regarding application efficiency in terms of execution time, costs, user satisfaction, reliability, etc. As these are propagated to the lower levels of the architecture, they become Process Performance metrics (PPMs) that characterize the service orchestration, the SLAs of the individual services, the configurations of the infrastructure, etc. These quality properties define the overall application requirements that ideally should be maintained within a single execution of the business process and across the different executions. To express the KPI and PPM requirements of this form one can refer to the SBA quality model specified in [6].

3.2 Adaptation and Monitoring Problem

The primary adaptation and monitoring problem in this scenario considers the quality requirements expressed as KPI, PPM, SLA, and other metrics of the application, across its functional layers. More specifically, the goal of the monitoring is to observe different quality values corresponding to the specified requirements, and, in case of the violation of the target values, to adapt the running business process (or future instances) so the violation is either prevented or corrected.

The monitoring and adaptation in such scenario has to face two key challenges. First, the violation of the high-level SBA requirements, expressed as KPIs, may be motivated by different factors and, more importantly, at different layers and components. Given the complexity of the application, and its service-oriented architecture, it is not possible to immediately discover which specific element caused the overall quality degrade. Therefore, there is a need for appropriate methods for performing a reasonable “diagnosis” of such problems.

Second, even if the problem (or a set of problems) is identified, it may not be immediately clear whether the associated adaptation action is suitable. Indeed, given the complexity of the applications, as well as the need to satisfy a range of requirements, the adaptations should be analyzed with respect to the impact they may have on other elements of the SBA and on the other requirements.

The integration scenario presented here aims to address these two challenges in the scope of the quality-driven adaptation of the service-based business processes.

3.3 Overall Approach

The approach used in this integration scenario is based on the framework presented in the work by Khazamiakin et.al. (2009) [7]. This framework realizes two principal ideas.

First, in the framework we exploit data mining techniques to perform the diagnosis of the problem that leads to the violation of the high-level quality requirements. To achieve this, we continuously monitor different quality properties of the different SBA elements and pass the monitored data to the quality factor analyzer. That analyzer exploits the data mining techniques to identify the impact of the elementary quality factors on the overall quality property. The resulting model, represented as a

dependency tree, gives the model of the adaptation requirements: it specifies which factors should be improved and in which combination in order to avoid a degradation in overall quality.

Second, regarding the individual quality properties at different layers, we associate different adaptation actions (also in that layer) which are expected to improve the specific quality factor and, therefore, contribute to an overall improvement in quality. This model, together with the combinations of factors to be improved provides a basis for the definition of a combined adaptation strategy. Finally, this strategy is realized using specific adaptation capabilities.

As adaptation activities associated to different quality factors we currently consider the following:

- *Service replacement*. The service replacement activity is associated to the metrics of a single process activity realized by a particular service. Specifically, these are the metrics that characterize the QoS of individual service. To evaluate which services to use as replacements the reasoning is based on current quality properties of the available services. This adaptation applies in case of external services.

- *Infrastructure management*. This refers to the operations over the configuration of the infrastructure (scheduling, resource allocation, etc.) on top of which the internal services run.

- *Re-composition*. This is an adaptation that aims to change the realizing service orchestration. Specifically, the change may refer to replacement of a process fragment in case when a service to be improved is a stateful, or may refer to changing one part of the process (e.g., sequence of activities) with another predefined subprocess (e.g., parallel execution of the same activities). In the former case, the adaptation is realized through the use of automated composition techniques.

3.3.1 Refined Architecture

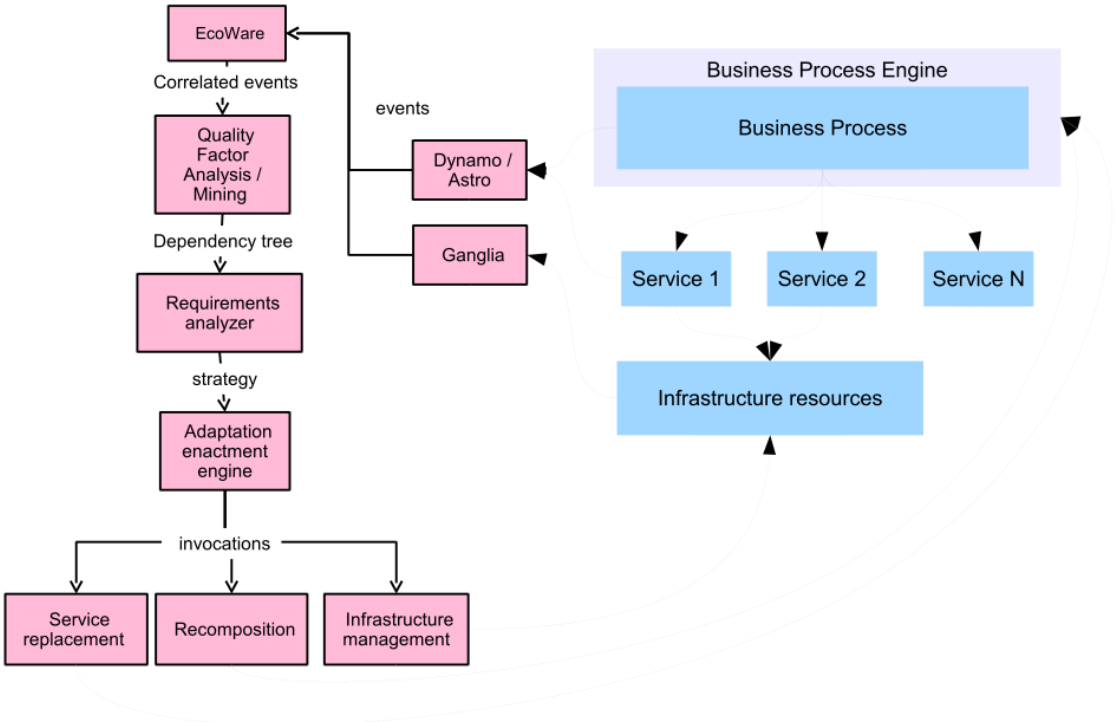

Figure 2Architecture of the scenario

Figure 2 shows the refined architecture of the scenario. The business process implementation, i.e., BPEL process instance, is running inside the process engine. The process instance interacts with the external and internal services (Service 1, Service 2, etc). The internal services are running on top of

the cloud-based infrastructure.

Monitoring

In order to collect the relevant quality information, the monitoring is performed in two steps. First, the basic information is gathered from the appropriate sources, in particular from:

- *Business process instances*. The information about the executions of the single activities, the whole process and its sub-processes is collected from the specific sources, made available via appropriate instrumentation of the process engine.

- *Constituent Services*. The information about the invocation of single services (both internal and external) is collected from the corresponding components.

- *Service Infrastructure*. Specific probes are attached to the relevant components of the infrastructure that manages the resources behind the internal services.

As can be seen from Figure 2, gathering information from business processes and services is performed by Astro and Dynamo monitoring tools [13], while service infrastructure is monitored by, e.g., Ganglia [22].

Once collected, the basic information on the different sources are aggregated and correlated by EcoWare [20] and then passed to the quality factor analyzer.

Quality Factor Analysis

To identify the need for SBA adaptation we first perform the analysis of quality factors. The analysis is based on machine learning techniques, more specifically on decision tree inference algorithms. The result of this analysis is a decision tree, which is called a dependency tree as it shows the main quality factors the KPI depends on. The leaves of the tree show whether the KPI is satisfied or violated in relation to its target values, and the number of process instances which led to this path. Nodes of the tree represent the quality factors influencing that KPI. The transitions leaving the node are equipped with the values of the corresponding factor representing the condition on that factor.

The KPI dependency tree represents the adaptation needs as it contains the “violation” leaves that the adaptation should eliminate. We remark that the adaptation may change the current configuration of the application that will have impact on the future instances of the business process, or may target the currently running instance, for which the overall tree is continuously reduced as the new monitored data arrives.

Requirements Analyzer and Strategy Selection

Starting from the dependency tree, the concrete requirements are extracted and transformed into the adaptation strategy (or set of adaptation strategies) that will be executed in order to improve the quality characteristics.

The first step of the strategy selection is the requirements analysis. The idea is to extract from the analysis tree the “good” paths (i.e., those that lead to “satisfaction” leaves) and to ensure the quality factors on those paths reach the required values. Specifically, we identify the (set of) sets of quality factors together with the target values for each of those factors. Based on these requirements, the concrete strategy is identified.

The strategy selection process is based on the following two principles. First, each metric may be associated with the set of alternative adaptation actions. For example, for the metrics associated to the external services one may associate service replacement policies or SLA re-negotiation actions

equipped with the target values from the identified requirements; for internal services it is possible to associate infrastructure management actions; for the process fragments one can associate alternative implementations of those fragments, etc.

The second step is to select the best possible candidates and their combination. Different decision mechanisms may be proposed for this purpose. Currently, we consider the following proposals that may be exploited in isolation or in combinations with the others:

- Selection of services based on their current characteristics taking also into account the impact on the other high-level SBA requirements [8];
- Selection based on degree of QoS satisfaction with fuzzy parameters [9];
- Use of soft constraints for QoS-based service selection [21];

The result of this step is the specific adaptation strategy or a list of strategies ordered according to some global criteria that combines different adaptation actions with concrete parameters.

Adaptation Enactment Engine and Adaptation Capabilities

Adaptations are enacted through the invocation the corresponding adaptation capabilities for each of the adaptation activities in the adaptation strategy. To coordinate different activities and to ensure their alignment it is possible to exploit a common adaptation framework, such as the one presented in [10]. In this framework the business process engine is instrumented using aspect-oriented programming (AOP) techniques to perform various adaptations, including service substitution, re-execution of an activity, use of alternative process fragment, etc. To accommodate other forms of adaptations, e.g., infrastructure adaptation, the framework may be extended with the corresponding calls to the infrastructure management API. The adaptation strategies are encoded in the form of composite adaptation rules.

3.3.2 Refined SBA Life-Cycle

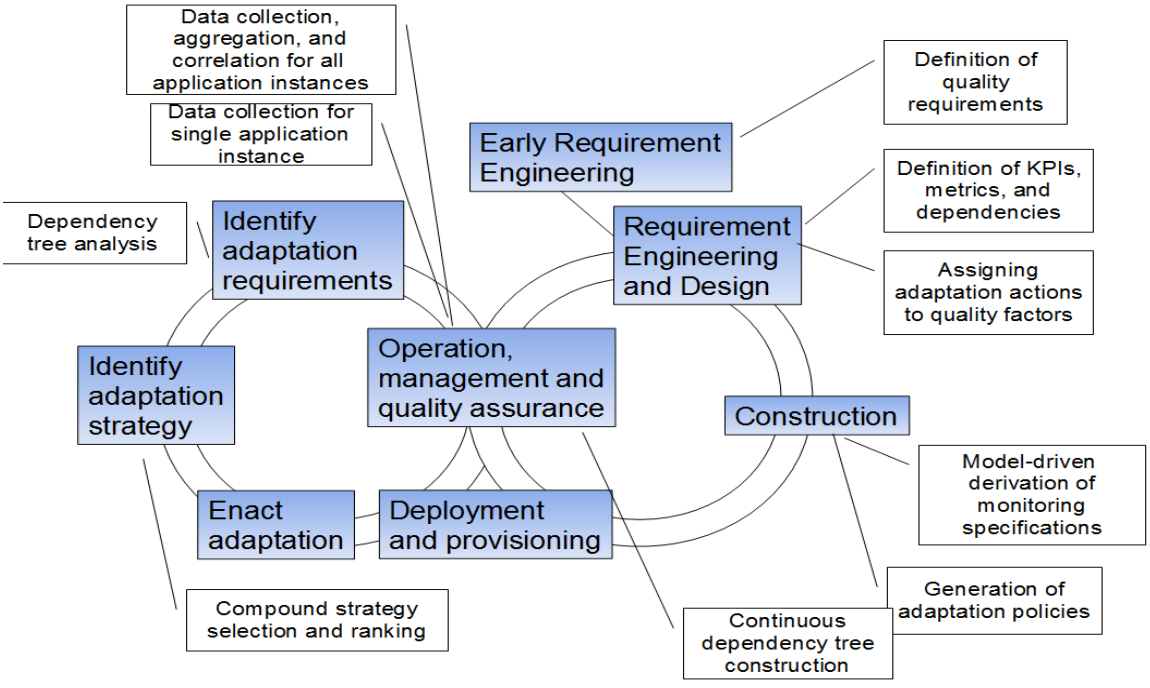

Figure 3 Mapping of the scenario to the S-Cube Life-Cycle

Figure 3 demonstrates the mapping of the key activities of the scenario onto the life-cycle of adaptable SBAs.

In the **Early Requirements Engineering** phase, the key activity for this scenario refers to the identification and elicitation of the primary quality requirements of the future business processes. Specifically, here the relevant quality dimensions are identified and characterized.

In the **Requirements Engineering and Design** phase, along with the definition of the process model and its projection onto the underlying service compositions and infrastructures, the quality requirements are materialized into the Key Performance Indicators (KPIs), quality metrics of the involved components, and their dependencies. To accomplish this, the Model-driven Management of Services [20] is used. In this approach, the quality properties are incrementally refined into the specific metrics across the different elements of the SBA, taking into account the future architecture of the latter. Besides, in this phase the different adaptation actions are associated to the identified quality metrics in order to be able to eventually recover their degradation. In this way, the metrics of the individual services are associated with the replacement actions, while the quality metrics of the infrastructure are associated with the available management actions, such as renegotiation, resource allocation, etc.

During the phase of SBA **construction**, the specific adaptation and monitoring activities involve the definition of the monitoring specification and of the concrete adaptation policies. As for the monitoring specification, the model-driven approach described in [20] is used. Specifically, the refined quality metrics are mapped to the different information sources and properties of the corresponding architecture elements using the appropriate metamodel. Starting from the model of metrics and raw data sampling model, the approach allows for automated extraction of the relevant simple and composite monitoring specifications. These specifications will then be loaded into the management infrastructure at the deployment time. Currently, the definition of the adaptation policies is not following model-driven approach and is based on a set of predefined configurations of the specific adaptations.

During the run-time cycle of the SBA, the adaptation and monitoring activities reflect the overall approach behind the scenario. Specifically, during the **Operation, Management and Quality Assurance** phase the raw data is collected and is aggregated according the monitoring specification defined during the design. Using the monitoring tools (e.g., Dynamo, Ganglia, etc.) the raw data is collected and then aggregated (e.g., using the EcoWare tool). The data collected is continuously used to build and evolve the dependency tree that characterizes the model of the quality factors and their dependencies across the different layers. Besides, the single metrics of every running instances are used to "instantiate" the tree with respect to that instance in order to define whether the adaptation is required for that instance.

The **Identify Adaptation Needs** phase is realized using the dependency tree analysis that aims at distinguishing the negative values of KPIs from the positive ones, and at identifying the influential factors for the former ones. This is done both for each instance and for the whole application model. Based on the identified factors, a set of adaptation requirements is extracted in the form of the combinations of factors to be improved and their target values.

The **Identify Adaptation Strategy** is built on top of the adaptation policies associated to the quality metrics. For each required combination the corresponding adaptation policies are chosen. To rank different possible solutions, various decision mechanisms are used, including the global KPI constraints, fuzzy decision support and the use of soft constraints. The latter methods are particularly useful in case of conflicting strategies.

The **Enact adaptation** phase is achieved through the invocation of interfaces of the specific adaptation actions as presented in the scenario architecture.

3.4 Individual Contributions

3.4.1 A Fuzzy Service Adaptation based on QoS Satisfaction

Title	A Fuzzy Service Adaptation based on QoS Satisfaction
Authors	B. Pernici and S. H. Siadat
Type	Technique / Experimental Evaluation
Short Description	The work addresses the issue of selecting adaptation strategies. We propose a fuzzy service adaptation approach that works based on the degree of QoS satisfaction. In particular, we define fuzzy parameters for the QoS property descriptions of Web Services (e.g. availability, satisfaction, response time). This way, partial satisfaction of parameters is allowed through measuring imprecise requirements. The QoS satisfaction degree is measured using membership functions provided for each parameter. Experimental results show the effectiveness of the fuzzy approach using the satisfaction degree in selecting the best adaptation strategy.

Adaptation and Monitoring Problem

Contribution to the adaptation problem	Our proposed approach for selecting adaptation strategies is based on quality changes as adaptation triggers. The selection is based on degree of QoS satisfaction with fuzzy parameters that is calculated using a fuzzy inference system. The decision making works based on an algorithm that define whether the QoS changes are compatible or incompatible with the SLA with respect to a predefined threshold degree for QoS satisfaction. The two main decisions are the internal renegotiation in which the changes are compatible with the service description in the contract and service replacement in which the changes are incompatible with the existing contract.
Contribution to the monitoring problem	--

Refined Architecture

Architecture elements	□ *Quality factor analysis and mining*: Quality factor analysis is based on fuzzy logic techniques. Various quality parameters are given as inputs into a fuzzy inference system that works based on if-then fuzzy rules. An example of if-then rule can be the following: *If (ResponseTime is compatible)* *and (Availability isCompatible)* *then (Satisfaction is high)* □ *Requirements analyser*: The output of the fuzzy inference system, the overall satisfaction degree of QoS, is taken as requirement of adaptation. An algorithm evaluates which adaptation strategy to take regarding to the predefined threshold degree for QoS satisfaction.
Requirements/Constraints	The approach assumes the relevant quality information is collected from the monitoring phase. We consider limited quality parameters in our web service quality model; however, our quality model can be extended to include broader quality metrics without major modification.

Refined SBA life-cycle

Life-cycle activities	□ *Requirements Engineering*: QoS definition, fuzzy parameters. □ *Construction*: definition of if-then fuzzy rules. □ *Identify adaptation needs*: quality factor analysis, identifying degree of QoS satisfaction. □ *Identify adaptation strategy*: Selection of adaptation strategies based on degree of QoS satisfaction.

Open problems and possible extensions

Open problems and extensions	- Identifying a broader variety of quality parameters. - Involving other adaptation requirements (e.g. cost and importance of QoS) - Applying hierarchical fuzzy systems for the adaptation strategy selection.

Extended abstract

Quality of Service (QoS) once defined in a contract between two parties may change during the life-cycle of Service-Based Applications (SBAs). Changes could be due to system failures or evolution of quality requirements from the involved parties. Therefore, Web Services need to be able to adapt dynamically to respond to such changes. However, formulating quality of service parameters and their relationship with adaptation behaviour of a service based system is a difficult task. Furthermore, an

essential issue to be addressed is how to efficiently select an adaptation while, there exists different strategies.

We propose a fuzzy service adaptation approach that works based on the degree of QoS satisfaction. In particular, define fuzzy parameters for the QoS property descriptions of Web Services. This way, partial satisfaction of parameters is allowed through measuring imprecise requirements using membership functions provided for each parameter. We define fuzzy rules for inferring the overall QoS satisfaction degree and its relation with adaptation actions.

The main point of using fuzzy logic is to find a relation and to map our input space to the output space. The inputs here are namely service availability and response time and the output is the overall satisfaction degree of them. For each QoS parameter in the service description we provide a membership function that represents the level of satisfaction the parameter. The membership functions map the value of each parameter to a membership value between 0 and 1. We use a piece-wise linear function, named trapezoidal membership function, for this purpose.

Having defined the membership functions, the mapping between the input and output space will be done by defining a list of *if-then statements* called *rules*. We use the satisfaction degree calculated using the fuzzy inference system for the adaptation decision making. The decision making mechanism works based on an algorithm that evaluates which adaptation strategy to take with respect to the predefined threshold degree for QoS satisfaction. The two main decisions are *internal renegotiation* in which the changes are compatible with the service description in the contract and *service replacement* in which the changes are incompatible with the existing contract. The former case deals with the internal contract modification with the same provider and requester while the earlier case requires the selection of a new service and establishment of a new contract which can result in a huge loss of time and money.

We have built and simulated a fuzzy inference system to interpret rules. We evaluate the effectiveness of the fuzzy approach with a non-fuzzy approach with respect to the stability of the system in terms of number of times a service needs to be replaced. The fuzzy approach performs a service replacement only if the result of the QoS satisfaction is lower than a threshold. While in the non-fuzzy approach, the replacement decision is done based on the precise evaluation of the QoS value ranges.

Our experimental results show the effectiveness of the fuzzy approach using the satisfaction degree in selecting the best adaptation strategy and reducing the number of service substitutions for minor deviations. Using fuzzy parameters we allow partial satisfaction of the parameters. Therefore, the decision making for adaptation is not based on the precise evaluation of the quality ranges and it is rather imprecise and allows the parameters to be relaxed. The non-fuzzy approach involved the maximum number of service replacement which includes more queries for the service selection.

3.4.2 Selection of Service Adaptation Strategies Based on Fuzzy Logic

Title	Selection of Service Adaptation Strategies Based on Fuzzy Logic
Authors	B. Pernici and S. H. Siadat
Type	Technique / Experimental Evaluation
Short Description	In this paper, a Fuzzy Inference System (FIS) is adopted for capturing overall QoS and selecting adaptation strategies using fuzzy rules. The overall QoS is inferred by QoS parameters, while selection of adaptation strategies is inferred by the overall QoS, importance of QoS and cost of service substitution. In particular, hierarchical fuzzy systems were used to reduce the number of rules. Our approach is able to efficiently select adaptation strategies with respect to QoS changes. We test and compare our fuzzy inference adaptation with a naive adaptation approach that works based on precise measurement of QoS in order to show the performance of the approach in reducing the number of service

	substitutions for minor deviations.

Adaptation and Monitoring Problem

Contribution to the adaptation problem	We are interested in changes of service quality as symptoms and triggers for adaptation. Besides, we consider cost of service substitution and importance of QoS as requirements/factors of adaptation for selecting the best strategy. We take advantage of the fuzzy logic for measuring overall QoS and selecting service adaptation strategies. We concentrate on QoS changes in a SBA and their relations with adaptation actions using fuzzy rules. We apply hierarchical fuzzy systems to address the rule explosion problem and to reduce the number of rules. This also provides the ability to efficiently manage different categories of input parameters, increase the scalability and improve the reusability of the system. In particular, our approach employs two fuzzy inference systems that use if-then rules to manage QoS changes and to support a decision making to decide which adaptation strategy is the best to be taken. We consider three adaptation strategies: do-nothing, renegotiation, and substitution.
Contribution to the monitoring problem	--

Refined Architecture

Architecture elements	• *Quality factor analysis and mining*: Quality factor analysis is based on fuzzy logic techniques. Various quality parameters are given as inputs into a QoS assessment engine which is a fuzzy inference system that works based on if-then fuzzy rules. • *Requirements analyser*: A decision making engine that is a fuzzy inference system is placed in the architecture for selecting adaptation strategies. The engine uses the overall degree of QoS received from the QoS assessment engine together with other adaptation factors to infer a decision making for adaptation. Information about adaptation requirements are kept in an adaptation rule base. The output of the engine, after defuzzification, represent the adaptation strategy needs to be taken.
Requirements/Constraints	The approach assumes the relevant quality information is collected from the monitoring phase. Our fuzzy system requires human expertise to define an exact membership function for each parameter and determine the number of rules.

Refined SBA life-cycle

Life-cycle activities	• *Requirements Engineering*: QoS definition, fuzzy parameters. • *Construction*: definition of if-then fuzzy rules. • *Identify adaptation needs*: quality factor analysis, identifying overall QoS, importance of QoS and cost of substitution. • *Identify adaptation strategy*: Selection of adaptation strategies based on an inference approach and adaptation priority.

Open problems and possible extensions

Open problems and extensions	- Identifying a broader variety of quality parameters (e.g. context). - Applying learning techniques to optimize number of rules. - Applying adaptive learning techniques to customize the membership functions.

Extended abstract

Web Service adaptation and evolution is receiving huge interest in service oriented architecture community due to dynamic and volatile web service environment. Regarding quality of service changes, Web Services need to be able to adapt dynamically to respond to such changes. However, formulating quality of service parameters and their relationship with adaptation behaviour of a service based system is a difficult task. In this paper, a *Fuzzy Inference System* (FIS) is adopted for capturing overall QoS and selecting adaptation strategies using fuzzy rules. We argue what is missing here is an appropriate decision making process for selecting adaptation strategies that takes into consideration different adaptation requirements. In this work we are interested in changes of service quality as symptoms and triggers for adaptation. Besides, we consider cost of service substitution and importance of QoS as requirements/factors of adaptation for selecting the best strategy. Therefore, the overall QoS is inferred by QoS parameters, while selection of adaptation strategies is inferred by the overall QoS, importance of QoS and cost of service substitution.

We consider four quality parameters in our web service quality model: response time, availability, security and reputation. We apply *hierarchical fuzzy systems* to address the rule explosion problem and to reduce the number of rules. This provides the ability to efficiently manage different categories of input parameters, increase the scalability and improve the reusability of the system. Particularly; we use two fuzzy inference engines, namely, *QoS assessment engine* and *decision making engine*. The former is used to infer the overall degree of QoS and the earlier is used to choose the adaptation actions. Each inference engine uses its own fuzzy rules. QoS-related rules are stored in a *knowledge base* and adaptation-related rules are stored in an *adaptation rule base*. QoS parameters perform as input variables for the QoS assessment engine. Fuzzy rules in the knowledge base can be used for inferring the overall QoS degree. The decision making engine works based on the results of the QoS assessment engine. Fuzzy rules in the adaptation rule base will be used for inferring an adaptation strategy. We consider three adaptation strategies: *do-nothing, renegotiation*, and *substitution*.

Our approach is able to efficiently select adaptation strategies with respect to QoS changes. We test and compare our fuzzy inference adaptation with a naive adaptation approach that works based on precise measurement of QoS in order to show the performance of the approach in reducing the number of service substitutions for minor deviations. In general, the experiments show that our approach intelligently reduces the number of service substitutions in comparison to the naive method. This is done by a trade-off between overall degree of QoS, importance and cost of adaptation. This way, minor deviations could be ignored due to high cost of service adaptation and partial satisfaction of QoS parameters is allowed.

3.4.3 Preventing KPI Violations in Business Processes based on Decision Tree Learning and Proactive Service Substitution

Title	Preventing KPI Violations in Business Processes based on Decision Tree Learning and Proactive Service Substitution
Authors	B. Wetzstein, A. Zengin, R. Kazhamiakin, M. Pistore, D. Karastoyanova, F. Leymann
Type	Methodology / Experimental Evaluation
Short Description	The work presents the instantiation of quality-driven SBA adaptation scenario, focusing on the instance adaptation via service substitution. The work describes complete methodology, from defining and monitoring KPI and potential influential quality factors to identification of adaptation need, selection various strategies and their ranking. Besides, the work presents the experimental evaluation of the approach.

Adaptation and Monitoring Problem

Contribution to the adaptation problem	With respect to the adaptation problem of the scenario the work focuses on the adaptation in service composition driven by service quality factor degrade leading to KPI violation. As an basic adaptation action the service replacement is used, while the service selection is driven by the dependency analysis and the revealed target values.
Contribution to the monitoring problem	--

Refined Architecture

Architecture elements	□ *Quality factor analysis and mining*: Quality factor analysis is based on data mining techniques. Various collected metrics values across process instance executions are feed into the WEKA data mining tool to build KPI dependency tree. □ *Requirements analyser*: adaptation requirements are extracted from the "green" paths of the instantiated dependency tree. Only "adaptable" metrics are considered (i.e., those that can be improved by service replacement). The list of service replacement strategies is ordered according multi-criteria ranking. □ *Adaptation enactment engine + service replacement*: the engine is implemented as the extension of Apache Orchestration Director Engine (ODE) process engine. The implementation deals with blocking instance execution, performing the analysis, and service replacement based aspect-oriented approach.

Requirements/Constraints	The approach relies on the ability to collect and properly correlate the quality metrics for the purpose of the analysis.

Refined SBA life-cycle

Life-cycle activities	□ *Requirements Engineering*: KPI definition. □ *Construction*: definition of probes, sensors, and data correlation mechanisms. □ *Operation, Management, and QA*: monitoring of metric values for a single instance; continuous evolution of dependency tree. □ *Identify adaptation needs*: quality factor analysis. □ *Identify adaptation strategy*: identification of service replacement strategies and their ranking. □ *Enact adaptation*: dynamic AOP-based service binding

Open problems and possible extensions

Open problems and extensions	- Integration of other types of adaptations. - Relation and integration of other types of selection strategies. - Analysis of impact of adaptations on other SBA components.

Extended abstract

This paper presents an adaptation approach that aims to address the problem of preventing KPI violation for a single business process instance. Very often the business processes realized on top of service-oriented architectures have a complex structure and involve a variety of services. It is not often possible to identify which properties of which constituent services have an impact on the KPI and have to be replaced. Furthermore, to prevent a violation in a single instance there is a need to anticipate the possible adaptation.

To accomplish this, the work relies on the quality factor analysis approach in order to automatically identify influential factors for KPI violations. The execution of the business process and its KPIs are continuously monitored. Based on historical data of this type the decision trees that express the influence of different service metrics on the business process KPI are constructed using data mining techniques. The resulting decision tree is used as a prediction model: for the running process instance the decision tree is used to identify the adaptation requirements in terms of services to be replaced and the target values they should meet. Multivalued ranking with global constraints is then used to order possible alternative adaptation scenarios. The analysis is done across the execution of process instance in different "checkpoints", thus enabling proactive adaptation of the instance.

The overall framework is completely realized and the experimental evaluation is performed. Specifically, the monitoring is done via instrumentation of BPEL engine where the implementing service orchestration is run; the data is collected and aggregated with ESPER monitoring tool. The data mining is performed with the Waikato Environment for Knowledge Analysis (WEKA) tool [29],

and the decision tree is analysed with an ad-hoc algorithm. The execution blocking and service replacement is performed via instrumentation of BPEL engine using AOP techniques. The experimental results show that the adaptation done in this way allows for preventing the violations in around 90% of cases. Furthermore, taking the adapted instance into consideration while learning dependencies further improves the quality of the proactive adaptation.

3.4.4 LAYSI: A Layered Approach for SLA-Violation Propagation in Self-manageable Cloud Infrastructures

Title	LAYSI: A Layered Approach for SLA-Violation Propagation in Self-manageable Cloud Infrastructures
Authors	Ivona Brandic, Vincent C. Emeakaroha, Michael Maurer, Sandor Acs, Attila Kertesz, Gabor Kecskemeti, Schahram Dustdar
Type	Methodology/Experimental Evaluation
Short Description	Brief description of the contribution with respect to the research problem presented in the integration scenario

Adaptation and Monitoring Problem

Contribution to the adaptation problem	The paper discusses an approach to autonomously adapt the different layers of the service infrastructure according to the SLA agreements with higher layers of the service based application
Contribution to the monitoring problem	The work proposes to monitor possible SLA-violations in the various layers of the infrastructure enabling either the propagation of possible violations to the higher layers or alternatively an autonomous reaction in the infrastructure.

Refined Architecture

Architecture elements	• *Monitors*: SLA-Violation sensors of the autonomic service instances • *Adaptation needs engine*: Autonomic manager of the current infrastructure layer (e.g. meta negotiation, meta brokering, automatic service deployment) • *Adaptation strategy engine*: The negotiation broker automatically analyses if the current SBA layer can handle the SLA-Violation without propagating it to higher levels for renegotiation • *Adaptation enactment engine*: Decision between service instance replacement or renegotiation • *Adaptation capabilities*: Dynamic binding (allows the use of new service brokers, the deployment of new service instances), SLA-renegotiation (allows the propagation of SLA-violations towards higher layers of the service based application)
Requirements/Constraints	The proposed techniques require the presence of a knowledge base describing the possible actions and reactions on the various SLA-violations

Refined SBA life-cycle

Life-cycle activities	• *Construction*: Developers identify the SLA-violation cases and specify the possible adaptation strategies in the knowledge base • *Operation, management and QA*: The autonomic manager controls the various infrastructure layers to react the possible SLA-violations • *Identify adaptation requirements*: When an SLA-violation is detected the Autonomic manager identifies the possible adaptation strategies according to the knowledge base of the infrastructure layer where the SLA violation occurred. • *Identify adaptation strategy*: The Negotiation broker decides whether the violation can be handled within the current infrastructure layer or a propagation is needed. • *Enact adaptation*: The negotiation broker initiates a renegotiation with the higher level SBA components if the infrastructure cannot handle the ongoing service request with the current SLA constraints • *Deployment and provisioning*: The infrastructure level adaptation could initiate a new service deployment to fulfil the constraints of the SLA or alternatively it can even select to reschedule the service request in a different infrastructure (e.g. redirect some requests to cloud infrastructures)

Open problems and possible extensions

Open problems and extensions	• Definition of SLA constraints for interfacing with higher level SBA components • Cost analysis of the interventions for avoiding future SLA-violations

Extended abstract

Cloud computing represents a promising computing paradigm where computing resources have to be allocated to software for their execution. Self-manageable Cloud infrastructures are required in order to achieve that level of flexibility at one hand, and to comply to users' requirements specified by means of Service Level Agreements (SLAs) at the other. Such infrastructures should automatically respond to changing component, workload, and environmental conditions minimizing user interactions with the system and preventing violations of agreed SLAs. However, identification of sources responsible for the possible SLA violation and the decision about the reactive actions necessary to prevent SLA violation is far from trivial.

First, in this paper we present a novel approach for mapping low-level resource metrics to SLA parameters necessary for the identification of failure sources. Second, we devise a layered architecture for the bottom-up propagation of failures to the layer, which can react to sensed SLA violation threats. We discuss a novel model for mapping of low level resource metrics to user defined SLA parameters. Thereafter, we present a communication model for the propagation of SLA violation threats to the appropriate layer of the Cloud infrastructure, which includes negotiators, brokers, and automatic service deployer.

3.4.5 A Soft-Constraint Based Approach to QoS-Aware Service Selection

Title	A Soft-Constraint Based Approach to QoS-Aware Service Selection
Authors	M.A.Zemni, S.Benbernou, M.Carro
Type	Model/Methodology
Short Description	Service-based systems should be able to dynamically seek replacements for faulty or underperforming services, thus performing self-healing. It may however be the case that available services do not match all requirements leading the system to grind to a halt. In similar situations it would be better to choose alternative candidates which, while not fulfilling all the constraints, allow the system to proceed. Soft constraints, instead of the traditional hard constraints, can help naturally model and solve replacement problems of this sort. In this work we apply soft constraints to model SLAs and to decide how to rebuild compositions which may not satisfy all the requirements, in order not to completely stop running systems.

Adaptation and Monitoring Problem

Contribution to the adaptation problem	With respect to the adaptation problem of the scenario the work focuses on the adaptation in service composition driven by QoS in an SLA. As a basic adaptation action the service replacement is used, while the service selection is driven by the required values in the SLA.
Contribution to the monitoring problem	--

Refined Architecture

Architecture elements	□ *Quality factor analysis and mining*: In order to find ranked solutions related to user preferences, Quality factor analysis is based on Soft constraint solving problem (SCSP) techniques. □ *Requirements analyser*: adaptation requirements are extracted from the Quality factor analysis step. Only acceptable values are considered (i.e., those that can be improved by service replacement). The list of service replacement strategies is ranked according to user preferences. □ *Adaptation enactment engine + service replacement*: The implementation deals with performing the analysis, and service replacement instrumented using aspect-oriented techniques.
Requirements/Constraints	The approach relies to overconstrained problems and offers a feasible solution by ranking solutions.

Refined SBA life-cycle

Life-cycle activities	□ *Requirements Engineering*: Definition of quality requirements in an SLA □ *Construction*: definitions of quality service relaxation and penalties. □ *Operation, Management, and QA*: monitoring the quality values. □ *Identify adaptation needs*: quality of values on the SLA □ *Identify adaptation strategy*: identification of service replacement strategies and their ranking depending of the service offerings. □ *Enact Adaptation*: The invocation of Interfaces.

Open problems and possible extensions

Open problems and extensions	- Integration of other types of adaptations. - Analysis of impact of adaptations on other SBA components.

Extended abstract

A (web) service can be defined as a remotely accessible software implementation of a resource, identified by a URL. A set of protocols and standards, such as WSDL, facilitate invocation and information exchange in heterogeneous environments. Software services expose not only functional characteristics, but also non-functional attributes describing their Quality of Service (QoS) such as availability, reputation, etc. Due to the increasing agreement on the implementation and management of the functional aspects of services, interest is shifting towards non-functional attributes describing the QoS. Establishing QoS contracts, described in the Service Level Agreement (SLA), that can be monitored at runtime, is therefore of paramount importance. Various techniques to select services fulfilling functional and non-functional requirements have been explored, some of them based on expressing these requirements as a constraint solving problem (CSP). Traditional CSPs can either be fully solved (when all requirements are satisfied) or not solved at all (some requirements cannot be satisfied). In real-life cases, however, over-constraining is common (e.g., because available services offer a quality below that required by the composition), and problems are likely not to have a classical, crisp solution. Solving techniques for soft CSPs (SCSP) can generate solutions for overconstrained problems by allowing some constraints to remain unsatisfied.

Our framework takes into consideration the penalties agreed upon on the SLA by building a new (Soft) Service Level Agreement (SSLA) based on preferences where strict customer requirements are replaced by soft requirements allowing a suitable composition. This agreement has to include penalty terms to be applied while the contract terms are violated. The framework seamlessly express QoS properties reflecting both customer preferences and penalties applied to unfitting situations. The application of soft constraints makes it possible to work around overconstrained problems and offer a feasible solution. Our approach makes easier this activity thanks to ranked choices. Introducing the concept of penalty in the Classical SCSP can also be useful during the finding and matching process

3.4.6 Model-driven Management of Services

Title	Model-driven Management of Services
Authors	Luciano Baresi, Mauro Caporuscio, Carlo Ghezzi, and Sam Guinea
Type	Methodology/Model/Prototype implementation/Experimental Evaluation)
Short Description	The work presents a model-driven approach for engineering manageable services. The approach models quality dimensions, management objectives, and key performance indicators. These concepts are exploited at runtime thanks to appropriate transformations. The methodology is supported by a prototype framework called ECoWare. An initial performance evaluation is also provided

Adaptation and Monitoring Problem

Contribution to the adaptation problem	--
Contribution to the monitoring problem	The paper contributes to the monitoring phase of the integration scenario. In particular, we use the event correlation capabilities of the ECoWare framework to understand behaviors that are witnessed at two different layers in the system (software vs. infrastructure).

Refined Architecture

Architecture elements	□ *Dynamo*: an aspect-oriented extension of the ActiveBPEL execution engine. Processes are instrumented with the code needed to gather behavioral events at runtime. □ *ECoWare*: Esper: a tool for complex event processing. Our event processing elements are automatically derived, from the management model, as specific instantiations of esper processors. *Siena*: a distributed event bus for communicating processors. SienaAdapter: needed to translate events to and from the format used within the Siena event bus.
Requirements/Constraints	The approach relies on the capability to capture runtime events within the system, and to translate them into ECoWare events. It is not necessary to use Dynamo, other runtimes could be used if they satisfy this requirement.

Refined SBA life-cycle

Life-cycle activities	□ *Early Requirement Engineering*: high level definition of quality dimensions of interest (CIM). □ *Requirement Engineering and Design*: concrete definition of quality dimensions of interest through pipe-and-filter composition of data sources and KPI processors. □ *Construction*: The data extraction, the event processors, and their inter-linkings are all automatically synthesized by the approach. □ *Deployment and Provisioning*: Synthesized event sources and processors are deployed to Dynamo and to the ECoWare system automatically. □ *Operation, Management and Quality Assurance*: Dynamo and ECoWare are runtime frameworks. Together they allow for complex monitoring of quality dimensions through event processing.

Open problems and possible extensions

Open problems and extensions	Currently the approach does not explicitly support events coming from the infrastructure level. A common abstraction for the software and the infrastructure layers may be needed.

Distributed ownership in service oriented systems means that it is problematic to measure and control quality of service indicators. Management activities are ever more important, since a complete validation of the system is impossible at design- or deployment-time. Management activities must become an integral part of the system's development process, from requirements elicitation, to the implementation, and to the execution. In this paper we present the Model-Driven Management of Services (MDMS) approach. It supports the explicit modeling of quality dimensions, management objectives, and key performance indicators, and the transformations required to exploit these notions at runtime. The paper also introduces ECoWare, a prototype framework for the deployment and operation of managed services.

MDMS provides a comprehensive MDE solution that considers management throughout the application's lifecycle. The application and its management features are developed in parallel, using different yet related models, transformations, and tools. At the CIM (Common Information Model) level, we analyze and collect requirements for the managed system, both for its functional design and its management.We use BPMN and appropriate annotations. At the PIM (Platform Independent Model) level, QoS dimensions are mapped to quantitative KPIs (Key Performance Indicators). We have two models at this level. The RDS (Raw Data Sampling) model defines what data are going to be needed at runtime. The KPI model describes how these data are correlated and aggregated to calculate the KPI's value. At the PSM (Platform Specific Model) level, we transform the RDS and the KPI models into concrete data collection models for the Dynamo composite service execution environment, and into processing models for ECoWare.

The RDS models are transformed into Dynamo sensor models using QVT (Query/View/Transformation), a standard for model transformation propose by the Object Management Group (OGM). Dynamo provides two different kinds of sensors: interrupt and polling sensors. Since the data produced by Dynamo do not directly comply with the event format used within EcoWare, they are transformed using specific adapter components, before being sent to EcoWare.

The KPI models are transformed into EcoWare configurations. EcoWare consists of a distributed event-bus implemented in Siena, and a number of processors, connected to that bus, that consume the events produced by Dynamo in a pipe-and-filter fashion. The processors are built as esper event consumers.The number of processors and their local configurations are automatically synthesized from the KPI models.

Early experiments demonstrate that the overhead introduced by Siena and Esper are negligible with respect to the overhead introduced by Dynamo for AOP-based data collection. The use of Dynamo, instead of a regular instance of ActiveBPEL, added an average 11 milliseconds to the execution of the well-known LoanApproval example process. Each sensor adds an additional 2 milliseconds.

4 Assumption-based Proactive Monitoring and Adaptation

4.1 Application Model

As a model of the SBA in this scenario we consider business processes that are realized on top of executable service compositions implemented in BPEL [11]. An important aspect, however, is that here we focus specifically on hybrid applications, i.e., those that incorporate both external third-party services and internal services realized within the organization where the business processes operate.

As for the external services, they are accessible over standard Web service protocols. The contractual aspects regarding the execution of those services and their characteristics are controlled by corresponding Service Level Agreements (SLAs).

The internal services, are realized on top of the service infrastructures, such as computational grids, clouds, or clusters.

The model of the applications of this type, besides the functional (e.g., service interface and protocol) and non-functional (e.g., SLAs) specifications of the involved services and the model of the service composition (i.e., BPEL specification), define also various requirements that regulate the correctness and quality of the application execution. These requirements may refer to the time properties of the overall process or of its constituent parts, constraints on the execution termination, costs, etc.

4.2 Adaptation and Monitoring Problem

Service-oriented applications (SBA) are deployed in dynamic and distributed settings. Due to these factors and also due to the fact that many aspects (like external services) are out of the control of the SBA developer, the applications are often subject to different failures and violations of the application requirements. These violations, even if properly handled and identified afterwards, may lead to the negative consequences such as contractual penalties. To avoid these situations, proactive adaptation is often needed.

In order to deal with these settings such applications are often provided with adaptation capabilities to react to failures during their operation. Failures can be for example unexpected changes of third party services. Failures can cause requirement violations, what might lead to negative consequences (e.g. contractual penalty). Thus, an SBA should be able to adapt proactively. The scope of the scenario is to be able to anticipate the needs for adaptation and to provide the corresponding monitoring and adaptation support in order to enable proactive adaptation of service-based business processes in case of failures and requirement violations.

The proactive behaviour of the adaptations we foresee in our scenario is of two forms. First, it is possible to adapt a single instance of the application, i.e., a single instance of the business process.

This happens when while executing a business process instance, it becomes evident that the subsequent execution will lead to the violations, and therefore there is a need to perform some changes in the running instance that would allow avoiding that. Second, it is possible to change the way the new instances of the process are executed. This takes place when one understands that the way the process instances operate is not optimal, there are new more efficient opportunities, or there are changes that require revision of the process implementation. This latter scenario refers more to the evolution of the business processes rather than adjusting running instances. It is important to remark, however, that also in this case the adaptation of the process model happens continuously, without bringing the application out of the production mode.

Another important aspect the scenario takes care of concerns unnecessary adaptations. In many cases the adaptation is carried out in case of certain events even if the overall requirement is not violated. To avoid such adaptations, different run-time techniques are necessary that analyze the need of adaptation in every concrete case.

4.3 Overall Approach

The idea that distinguishes the approach in this integrated scenario is based on the use of *assumptions*. An assumption represents some expectation regarding the elements of the SBA that are not under control of the process owner. More specifically, we refer to the assumptions regarding the properties of the third-party services captured with their SLAs, service interfaces and protocols captured with their specification under certain version, the assumption regarding the infrastructure involved (e.g., network throughput, use of the underlying resources, etc.). In the following figure the variable **a1** depicts an assumption about **Service 1**. **m1** is the monitored data.

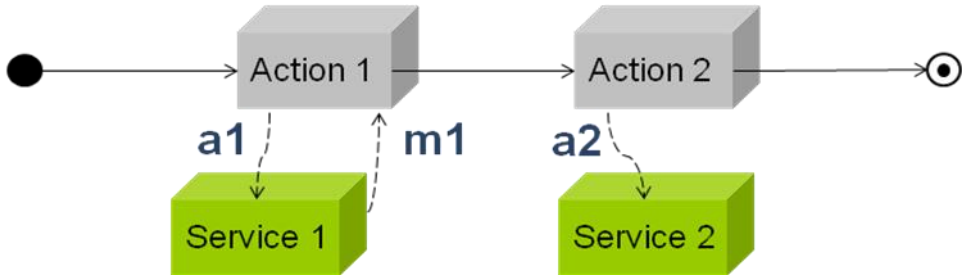

Figure 4 Assumptions about Services

The important aspect of the assumptions in our scenario is that the assumptions are used to relate the continuously monitored data to the SBA requirements. In particular, we exploit monitors to check whether the assumptions are still satisfied, and, in case of violation of a particular assumption, we perform immediate checks whether the overall requirement is violated, given the already known information and the assumptions about parts of the process that have not been executed. For the above example: if assumption **a1** about **service 1** is violated by monitoring data **m1**, we then check whether the requirement is violated given **m1** and the assumption **a2** about the not yet invoked **service 2**.

Currently, in the scenario we consider the following forms of adaptations. In case of an external service we request the service repository to provide an alternative service or re-negotiate the SLA associated with that service. In case of the internal service we can influence the infrastructure, on which the service is executed (SI layer). In case of necessity to recover the process, we perform process adaptation, where the new subprocess is automatically composed in order to bring the process back to the normal execution or to perform some recovery/compensation activities.

4.3.1 Refined Architecture

The architecture of the solution is represented in Figure 5. The application, i.e., business process is running within the process engine that communicates with the services through an Enterprise Service Bus (ESB), that routes the invocation of services and at the same time serves as an aggregation point for different types of events from different sources (e.g., from service calls, process engine, etc). The service invocation is operated by the proxy service component that enables dynamic rerouting of requests to different services. The information about available services and their current properties is stored in the service repository.

To coordinate these actions we use agent. The agents of the built-in multi-agent platform are controlling the adaptation loop: they control the monitoring, act on adaptation needs, decide on adaptation strategy and instrument the effectors to perform the adaptation. This can be seen as an extra layer on top of the existing components of the SBA (not depicted in Figure 5).

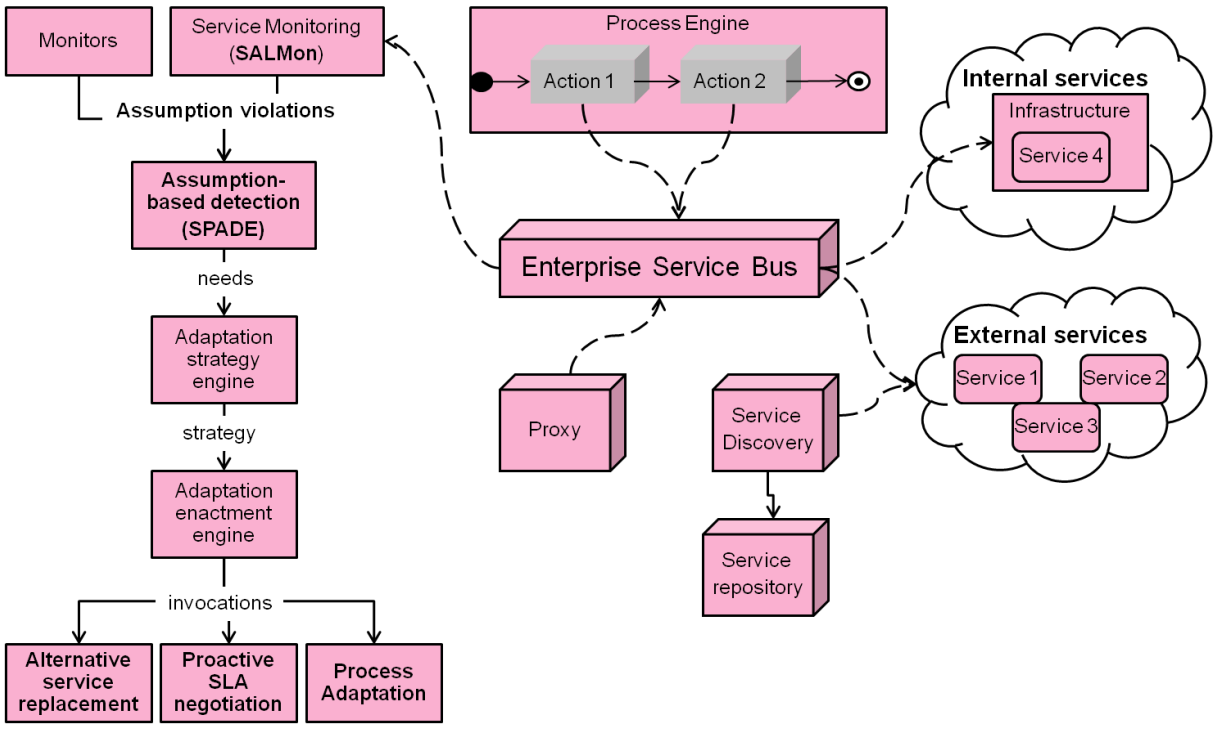

Figure 5 Architecture of the scenario

Monitoring

The key monitoring component is SALMon (cf. [12,29]). SALMon is used to monitor assumptions regarding the different properties of individual services. This includes both functional (e.g., service interface, protocol, signatures, etc.) and non-functional properties. With respect to the latter, SALMon is able to (1) monitor the QoS of services in a SBA and (2) check if the retrieved QoS fullfils the SLOs stated. In order to be easily integrable with other technologies, SALMon has been implemented as an SBA by itself, providing two services:

- Monitor service: to retrieve the QoS data of a service. The monitor service provides an interface to define what the quality metrics to retrieve in a SBA are (e.g: Response Time of service S1).

- Analyzer service: used to check on runtime if a given service fulfills the expected SLOs. It provides an interface to define the conditions to be checked in a SBA.

Besides, the monitoring framework makes use of *measure instruments*, which are the components that implement the logic needed in order to obtain the value of a concrete quality metric (e.g: Response Time, Availability,…) of a service. They are located in a module installed in the ESB and managed by the Monitor.

Additionally to SALMon, other monitoring solutions may be considered to evaluate more complex properties, such as [13].

Adaptation Needs Engine

If a violation of the stated conditions occurs, the Analyzer notifies it to the corresponding component, in this case, the *Specification and Assumption Based Detection (SPADE)*. The information given in the notification is the identifier of condition stated and the value of the metric that is violating the condition. In other words, the notification represents the violated assumptions and the corresponding data.

When the assumption is violated the SPADE component is used to evaluate the impact of the violated assumption on the corresponding application requirement. This check allows us to reduce amount of unnecessary adaptation.

The SPADE component actually represents a set of components, each having its own mechanism for the verification of the requirement. In particular, the following decision mechanisms are used:

- Run-time verification of the process model against KPI requirements [14]. Here the properties related for example to the execution times of the process are checked given the (BPEL) process model, the current state of the process the assumptions regarding the remaining services and the time currently spent (resulting from service monitoring).

- Evaluation of changes in external service interfaces and protocols [15]. In this case a special-purpose analysis is performed in order to estimate whether the impact of change is substantial, and therefore the new version of a service require appropriate changes in the service composition. If this is the case, the SPADE may trigger evaluation of the SBA, while keep using the old version of the service (if available) or switch to other service (if old version is not available). If the change does not harm the application, it is also possible to switch to the new version of the service.

- In case of violation of process context assumptions, the additional activities may be necessary in order to bring back the process to its expected context and/or to perform additional recovery and compensation activities [16]. The analysis of the context assumptions is performed based on the context properties associated to the overall application and to the specific subprocess executed currently. If the assumption is violated, the adaptation requires the process to get back to the normal execution by composing a new process that may change the critical properties as it is required by the process specification.

As an outcome of the analyses the SPADE framework passes to the adaptation strategy engine the specific input that depends on the concrete SPADE component that triggers adaptation. In particular, the first element may trigger service replacement or SAL renegotiation; the second one may trigger process model evolution or service replacement; the third one requires new service composition to be provided in order to bring the process to its normal execution context; finally the service infrastructure could be adapted, thus having an impact on non-replaceable internal services.

Adaptation Strategy Engine

The adaptation strategy engine is implemented as a multi-agent platform providing intelligent background control for the execution environment (cf. [27]). Business process instances are represented by individual Process Agents responsible for the proper and timely execution of the

processes. On the other hand, services are represented with Service Agents responsible for ensuring the availability of the services. These agents can be grouped according to customers, providers, etc. Agents implement intelligent behaviors and negotiation with each other in order to collect available information and make decisions on adaptation strategies.

Service Agents are instantiated by the service infrastructure, and they collect information on various QoS measures of the service. Based on the collected information, they send concrete adaptation solutions on request to Process Agents. Upon agreement, the Service Agent instructs the Adaptation Enactment engine to execute the selected adaptation method.

The Process Agents are instantiated by the adaptation extension of the Business Process execution engine before the process starts. The monitoring component triggers the responsible Process Agent whenever a deviation from the agreed SLA is forecasted. In this case, the Process Agents broadcasts a request for an adaptation solution to Service Agents, which, in turn, respond with their offered adaptation solution. The Process Agent can decide on which offer to take, based on the strategy implemented in its behavior and by using SPADE. By invoking SPADE the Process Agent assures the adherence to the SBA requirements. The Process Agent selects and notifies a Service Agent.

In the next phases the adaptation strategy engine may be equipped with the specific approaches that aggregate and coordinate different, possibly alternative, adaptations into a single strategy prioritized according some predefined criteria.

Adaptation Enactment Engine and Adaptation Capabilities

Once the selected Service Agent is notified, the proposed change is enacted. The enactment of the adaptation strategy can be accomplished through the following adaptation activities:

- *Service replacement*. To perform service replacement, the service repository is involved. In particular, the request for service replacement is transformed into the query to the *runtime service discovery tool*. The identification of alternative services is based on various characteristics of the published service such as structural, behavioural and quality characteristics that services should have in order to be acceptable replacements for a constituent service. These characteristics are specified in service discovery query where a query is associated with each of the constituent services of the service based system. The service discovery tool operates also in proactive mode, where the query is executed in parallel with the operation of the service based system in order to discover and maintain a set of candidate replacement services for the constituent services. The actual service replacement is performed via proxy component will route the abstract activity invocations within the SBA to the required services. The proxy server has been implemented as an HTTP server using Java socket programming. It receives calls from the service-based system when a participating service needs to be invoked and reroute the SOAP-messages to the alternative service, picked from the Service Repository.

- *SLA re-negotiation*. SLA negotiation broker performs SLA negotiation for each candidate service identified by the service discovery tool. In this process the desired level of service is negotiated with the selected candidate service. In this phase, the QoS characteristics of each candidate service are negotiated in order to achieve the best possible SLA for the services that is within the boundary constraints of the service provider and the consumer. Similar to service discovery process, the negotiation process can be either reactive or proactive. The framework allows to specify negotiation triggering rules, that determine the circumstances under which the negotiation of new service level agreements should start (e.g., when a provisionally agreed SLA is about to expire, or change in the requirement of SBS). Once an SLA comes into force, its guarantee terms and negotiation triggering rules become subject of monitoring. If the monitoring process detects violation of the SLA or the deployed service becomes unavailable then the service is replaced by the best candidate service identified in the service discovery process. The detection of violation of the conditions in negotiation triggering rules triggers the renegotiation/negotiation to establish a new SLA.

- *Process adaptation*. When the process should be adapted in such a way that the business process can be brought back to its normal execution, the adaptation activity consists of generating a new composed subprocess that performs the necessary modifications in the SBA environment. To accomplish this planning techniques are exploited. In particular, the composition planner takes as input the model of relevant services available, the current state of the process, the expected context configuration, and the additional requirements that characterize eventual recovery requirements. Based on that input a new process is generated that allows to get back to the same execution point in the process but also perform the necessary modifications in the SBA domain through the invocation of appropriate services. After that the special orchestrator component executes the new subprocess, and, if it terminates successfully, resumes the execution of the main process.

- *SI Adaptation.* Usually more than one external service expose a specific functionality, needed to meet the SBAs requirements. Adaptation capabilities like service substitution presume those redundant service facilities. In case of internal services we are facing the problem, that those services are instantiated usually once. In order to achieve a higher flexibility, we propose to influence the infrastructure of those internal services. E.g., in order to catch up lost time, caused by former service failures, the execution speed of an internal service could be increased, by assigning more CPU time to this service (SI layer).

4.3.2 Refined SBA Life-Cycle

Below we will present the key phases of the SBA life-cycle that are relevant for the scenario.

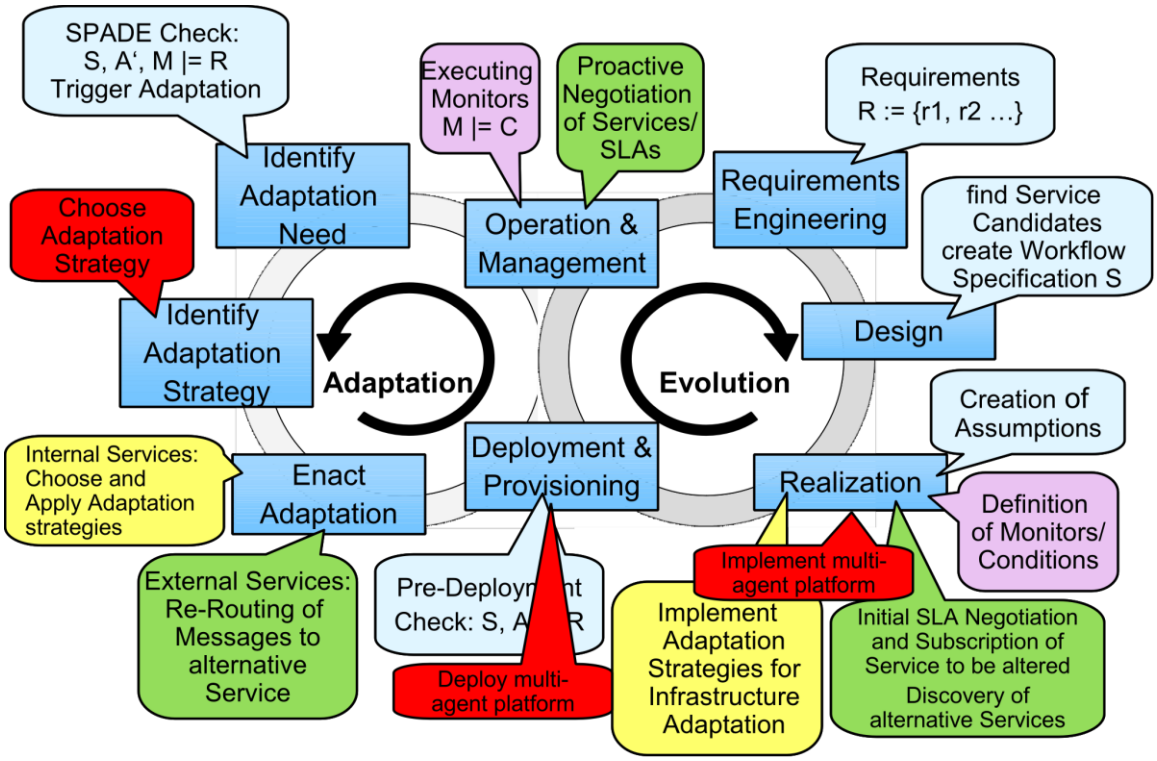

Figure 6 Approach along the S-Cube Life-Cycle

Starting with the **Requirements Engineering** phase the requirements are elicited and specified using template-based documents as introduced in [17]. Next, the application designer formalizes the elicited SBA requirements for example in ALBERT.

In the **Design Phase** service candidates are selected. Moreover, the SBA is specified in a service composition language such as BPEL [11]. Together with the main process, the specific recovery and

compensation rules are defined. Besides, at the design phase the relevant context properties are modelled; the services are related to these properties through preconditions and effects as specified in [16].

During the **Realization** phase the key activity for our scenario is the definition of the SBA assumptions and the definition of an adaptation strategy. In addition, the multi agent platform is configured.

As for the service assumptions, the first step is to perform the initial SLA negotiation for each constituent service used in the BPEL specification. The SLA negotiation process takes into account the negotiation rules specified by the application designer and this process is carried out by exploiting the technique described in [18,19]. In order to enable the exchange of services a set of potential alternative services for each of the constituent service is identified and a pre-agreed SLA is established with each identified service. The service discovery is driven by the discovery queries specified in the evolution phase [18,19]. Based on the negotiated SLAs the assumptions are derived.

As for the SBA context, the assumptions associate to different parts of the process specific context configurations under which the part may be executed.

The configuration of the multi agent platform comprises the definition of adaptation strategies. For this purpose the platform must have knowledge about the adaptation capabilities and the structure of the business process. Furthermore, the agent platform must have an access to the application requirements and to SPADE. The latter access is necessary, as during runtime the multi agent platform instruments SPADE to check whether after the proposed adaptation the requirements will be still met.

In the **Deployment & Provisioning** phase the developed application is check against its requirements. The SBA is formally verified to ensure that it satisfies the requirements under the given assumptions. To accomplish this one can apply, for example, the mechanisms described in [12] for this formal verification. If the designed application passes the formal verification, the application is ready for deployment.

After deployment the SBA is executed, which is called the **Operation & Management** phase. During this phase several mechanism are executed:

- The Service Repository checks for alternative services permanently.
- During the SBA runtime the adherence to the assumptions is monitored using SALMon. If a violation of an assumption is detected, SPADE performs a runtime analysis, as the application might violate the SBA requirements (cf. [14]).

In order to **Identify Adaptation Need[s]** the SPADE approach is applied. It performs a runtime re-verification of the service based application is performed to check, whether the application still satisfies the overall requirements. The workflow specification, the monitoring data of the invoked services and the assumptions of the not yet invoked services are checked against the requirements. If the service based application fails to pass the analysis, an adaptation of the service based application is necessary.

After SPADE identified the adaptation need, an appropriate adaptation strategy must be identified, which is performed during the **Identifiy Adaptation Strategy** phase. The agents are proposing a strategy, which includes the invocation of at least one adaptation capability. In order to check if the requirements are still met, the agents instrument SPADE. If this is the case the adaptation is enacted.

Finally the agent based adaptation engine **Enact[s] [the] Adaptation** executes the adaptation strategy. The available adaptation capabilities are invoked, such as service replacement, SLA negotiation, process adaptation, and even process model evolution.

4.4 Individual Contributions

4.4.1 Proactive SLA Negotiation for Service Based Systems

Title	Proactive SLA Negotiation for Service Based Systems
Authors	Khaled Mahbub and George Spanoudakis
Type	Methodology / Technique
Short Description	This work proposes a framework for proactive SLA negotiation that integrates this process with dynamic service discovery and, hence, can provide integrated runtime support for both these key activities which are necessary in order to achieve the runtime operation of service based systems with minimised interruptions. More specifically, our framework discovers candidate constituent services for a composite service, establishes an agreed but not enforced SLA and a period during which this pre-agreement can be activated should this become necessary

Adaptation and Monitoring Problem

Contribution to the adaptation problem	The proposed approach ensures that a service, which could be potentially used as a replacement service to adapt a service based system (SBS), will have an agreed set of guaranteed provision terms if the need to deploy it arises at runtime. Hence, when this need arises it will not be necessary to engage in a lengthy negotiation process interrupting the operation of the service based system. Therefore this approach facilitates proactive adaptation of service based systems.
Contribution to the monitoring problem	--

Refined Architecture

Architecture elements	□ The *runtime service discovery tool* is used to identify potential alternative services for the services that an SBS uses currently. The discovery process is driven by service discovery queries. These queries are associated with each of the constituent services of the SBS and specify the conditions that should be satisfied by any service that could replace them in the SBS. These conditions can refer to the structural (interface), behavioural, contextual, and quality characteristics that services should and, therefore, provide the criteria for discovering candidate constituent services. □ The *negotiation broker* is the component that manages the negotiation process on behalf of a service consumer (i.e., the composite service) or a service provider. Negotiation brokers are responsible for negotiating and agreeing the guarantee terms of an SLA. The negotiation process can be either reactive or proactive. In proactive negotiation, the negotiation process is carried out according to a two-phase protocol that may result in a provisionally agreed SLA but not activated SLA or negotiation failure. In reactive negotiation, the negotiation process is executed according to a single phase protocol that can result in an agreed and activated or negotiation failure. □ A monitor is used to detect if the SLA guarantee terms which should apply to the provision of a service are satisfied.
Requirements/Constraints	Monitoring of the SLA terms requires runtime collection of quality metrics.

Refined SBA life-cycle

Life-cycle activities	□ *Construction*: specification of service discovery query, negotiation rules. □ *Operation and Management*: identification of alternative services, negotiation of SLA, monitoring of enforced SLAs. □ *Identify adaptation strategy*: identification of best possible alternative service for a service to be replaced in SBS.

Open problems and possible extensions

Open problems and extensions	□ Thorough experimental evaluation of the framework □ Support for proactive negotiation of hierarchical SLA. □ Dynamic adaptation of negotiation rules, i.e. negotiation rules can be changed during negotiation process.

Extended abstract

This paper proposes a framework that integrates service discovery and monitoring of service in order to facilitate proactive SLA negotiation. The service discovery process is used by service consumer applications (i.e service based systems) in order to identify potential alternative services for the services that they currently use. The identification of alternative services is based on various characteristics of the published service such as structural, behavioural and QoS specified by the service consumer. Proactive SLA negotiation is weaved into the discovery process as an activity that is performed after the execution of the service discovery query and constitutes an extra check prior to putting a service into the replacement set. The objective of proactive negotiation is to establish an agreed but not enforced SLA and a period during which the consumer of the service will be able to activate the pre-agreement should this become necessary. Following this, the relevant candidate constituent service can be inserted in the replacement set. The negotiation process is carried out according to a two-phase protocol that may result in a provisionally agreed SLA but not activated SLA or negotiation failure. A provisional SLA is a service level agreement that has been agreed by service consumers but has not been activated yet. Such SLAs may be open ended or have an expiry date by which they will either have been activated or cease to exist. In the negotiation process, desired level of service and other individual or collective issues (e.g. penalty, SLA time frame) are negotiated with each alternative service. It should be noted that a service consumer may specify the bottom line of various QoS characteristics in the service discovery query that enables the service discovery process to identify potential alternative services. But in the negotiation phase, each alternative service is negotiated over the QoS characteristics in order to identify a service that fulfils the service consumer's (and also service providers') requirements in optimal way. The monitoring process monitors the runtime behaviour of the service provider and the service requester in order to detect if the agreed SLA is satisfied. If the monitor detects violations of the SLA or detects situations that requires proactive negotiation then the monitor triggers the framework to initiate a renegotiation. The negotiation process is also repeated when a pre-agreed SLA comes close to expiry and, therefore, it has to be renegotiated.

We have an initial implementation of the framework. The current implementation of the framework includes a negotiation broker for a rule-driven negotiation engine that we have developed based on Jess rule engine. All the major components of the framework (i.e., the negotiation brokers, runtime service discovery tool, and service listeners) have been implemented in Java.

4.4.2 Evolving Services from a Contractual Perspective

Title	Evolving Services from a Contractual Perspective
Authors	V.Andrikopoulos, S.Benbernou, M.P.Papazoglou
Type	Model/Methodology

Short Description	Uncontrolled changes can easily break the existing relationships between a service and its environment (its customers and providers). In this paper we present an approach that allows for the controlled evolution of a service by leveraging the loosely-coupled nature of the SOA paradigm. More specifically, we formalize the notion of contracts between interacting services that enable their independent evolution and we investigate under which criteria can changes to a contract-bound service, or even to the contract itself, be transparent to the environment of the service.

Adaptation and Monitoring Problem

Contribution to the adaptation problem	With respect to the adaptation problem of the scenario, the work focuses on the evolution of services. It is the case where the internal service influences the infrastructure, on which service is executed. The adaptation action takes place under different versions of services.
Contribution to the monitoring problem	Continuously monitored data to the SBA requirements through different service versions.

Refined Architecture

Architecture elements	□ *Monitor*: functional properties i.e. interfaces, protocols and signatures are monitored using SALMon. □ *SPADE*: Evaluation of changes in external services interfaces and protocols is handled in this component. □ *Adaptation strategy*: Impact of changes is performed. Either the old version of service is kept or switch to other service if change harms or not the application. □ *Adaptation enactment engine*: The implementation deals with performing various changes regarding a contract.
Requirements/Constraints	The approach relies to shallow changes in a contract through different versions.

Refined SBA life-cycle

Life-cycle activities	□ *Requirements Engineering*: Definition of requirements in a contract (interfaces, behavior) □ *Realization*: Service versioning based changes discovery. □ *Operation and management*: monitoring the changes. □ *Identify adaptation needs*: SPADE triggers adaptation for external services. □ *Identify adaptation strategy*: identification of service evolution strategies. □ *Enact Adaptation*: it is the case of external services: re-routing of messages to alternative service.

Open problems and possible extensions

Open problems and extensions	- Integration of other types of adaptations. - Analysis of impact deep changes.

Extended abstract

The goal of this work is to allow the independent evolution of loosely coupled interacting parties in a transparent manner so as to preserve their interoperability. In this context, the parties involved in an interaction can either be services, or services and client (service-based) applications. We only consider bilateral interactions, and for each such interaction we distinguish two roles: that of the producer and that of the consumer. It must be kept under consideration that the role of a service, unlike that of an application that always acts as a consumer, can vary depending on the interaction. An aggregate service for example plays both roles: that of the producer for its clients, and that of the consumer when it interacts with the aggregated services to compose a result. To achieve meaningful interoperability in this context, service clients and providers must come to a mutual agreement, a contract of sorts between them. A contract of this type formalizes the details of a service in a way that meets the mutual understanding and expectations of both service provider and service client. Building around this idea, we are presenting mechanisms to effectively deal with the evolution of the structural aspect of both parties, while preserving interoperability despite the changes that may affect them. After we lay down this foundation we discuss the evolution of interactions and contracts themselves.

We introduce the contract construct as the means to leverage the decoupling of the interacting parties. We present a contract constructing function that bridges the gap between service matching and service mapping. Following on, we build on contractual invariance and contractual evolution to show how to effectively deal with shallow changes to the service provider and client interaction - without the need for adaptation which may lead in turn to deep changes.

4.4.3 Adaptation of Service-based Business Processes by Context-Aware Replanning

Title	Adaptation of Service-based Business Processes by Context-Aware Replanning
Authors	Antonio Bucchiarone, Raman Kazhamiakin, Marco Pistore and Heorhi Raik

Type	Technique
Short Description	This work contributes to the scenario on assumption-based proactive SBA adaptation. Specifically, the presented approach focuses on modeling and monitoring business process context assumptions to timely trigger process adaptation and recovery. The work focuses on the modeling and adaptation parts. The adaptation is performed completely at run-time using automated service composition techniques.

Adaptation and Monitoring Problem

Contribution to the adaptation problem	With respect to the adaptation problem the approach contributes with explicit model of context properties and context assumptions to drive the execution and proactive adaptation of the processes. Specifically, before executing the activities the approach controls that the context state corresponds to the defined context assumptions, and, in case of violations, triggers process re-planning using automated service composition.
Contribution to the monitoring problem	This approach contributes to the monitoring problem with the model of context properties to be observed and their changes.

Refined Architecture

Architecture elements	□ *SPADE framework*: the relevant component controls the evolution of context and context assumptions. □ *Adaptation Strategy Engine*: the strategy is defined as the adaptation planning problem that includes the state of the context, of the process and of the service, the model of domain (assumed context model and services), and the target context state. □ *Adaptation Enactment Engine*: using the automated planning the adaptation problem is solved resulting a new sub-process to be executed to bring the process to a normal execution mode. □ *Process Adaptation* mechanism: the extension of the process engine component to control and block the normal execution of the process to enable context checks and injection of new adaptation subprocess.
Requirements/Constraints	Requires the ability to query and monitor business context state.

Refined SBA life-cycle

Life-cycle activities	• *Requirements Engineering*: Definition of coordination requirements. • *Design:* definition of service context policies and context effects. • *Operation, Management, and QA*: context monitoring. • *Identify adaptation needs*: validation of context assumptions. • *Identify adaptation strategy*: extraction of the adaptation problem definition. • *Enact adaptation*: construction and execution of adaptation sub-process using automated service composition techniques.

Open problems and possible extensions

Open problems and extensions	– more sophisticated mechanisms that provide different ways to react at unexpected situations. – evolution mechanisms order to progressively improve processes instances that may then be used if a new process model shall be instantiated.

Extended abstract

Operating in open and dynamic environments, business processes often need to be adapted during the execution to react to changes and unexpected problems. The paper presents an adaptation approach that aims at addressing dynamic adaptation of business processes in case of unexpected business context changes. The approach relies on the following key elements:

- Context-aware model of the application. We explicitly model the business context, in which the process operates, and describe how the services and policies of the application are related to the context and its changes.
- Context-aware execution framework. While executing business process, the process context is continuously monitored in order to check whether it changes as expected by the policies and service specifications.
- Context-aware adaptation based on automated service composition. If a critical context change is detected, the adaptation tries to recover the process. This is achieved through construction of a composite service that, starting from the actual context state, performs the necessary changes in the domain to bring the process and its context to the expected state.
- Context model. The paper presents an explicit model of the context that consists of: context property diagrams that model context properties and their evolution (represent possible values of the property and the changes of the property values as transitions); service effect annotations to model how the services change the context, i.e., how the service changes the property value; business policies over the services to state in which context setting the service may be executed (annotating services with the preconditions on the context property values).
- Execution framework. The execution and adaptation of the reference process is controlled and coordinated by the Orchestrator component: based on the analysis of the current context and its changes, it decides whether to proceed with the execution or to perform the adaptation. When the observed values of the context properties violate the preconditions of the next activity to execute, the process adaptation is initiated. The Adaptor

component generates the new subprocess, which aims to do its best to "recover", so that the blocked activity can be executed and the main process can continue. Orchestrator starts the execution of the generated subprocess and then continues the execution of the main process. If during this execution a violation of context assumptions is detected again due to exogenous contextual changes, a new round of adaptation is undertaken. To accomplish the required adaptation, Adaptor generates a composition of those services, which, being executed together, achieve the necessary effect on the context. The construction of the composition is performed with the use of automated planning techniques: the service specifications, the model of context (i.e., context diagrams), and the goal specifications are transformed into the planning problem and the resulting plan is then transformed into the composed service.

4.4.4 Towards Proactive Adaptation: A Journey along the S-Cube Service Life-Cycle

Title	Towards Proactive Adaptation: A Journey along the S-Cube Service Life-Cycle
Authors	Andreas Metzger, Eric Schmieders, Cinzia Cappiello, Elisabetta Di Nitto, Raman Kazhamiakin, Barbara Pernici and Marco Pistore
Type	Methodology / Technique / Experimental Evaluation
Short Description	The proposed approach is both a technique, as it facilitates the adaptation need identification on a technical level, as well as a methodology, as it proposes a process model which aims for producing the necessary artefacts during design time.

Adaptation and Monitoring Problem

Contribution to the adaptation problem	The approach exploits run-time verification capabilities during the operation of service-based applications. It thereby provides a considerable improvement for what concerns the number of false-positive adaptation triggers when compared to state-of-the-art techniques.
Contribution to the monitoring problem	-

Refined Architecture

Architecture elements	• *Designtime methodology:* The approach suggests a process model along the phases of the SBA life-cycle model as elaborated by the S-Cube project. For an SBA to quickly and dynamically adapt to changes in service quality, the relevant artefacts, as well as the properties of the SBA and context are formalized during its design time to make them amenable to automated checks and decisions during run-time. • *Identification of adaptation needs:* If an assumption A (derived from an SLA) is violated, the approach checks, whether the requirements R are still satisfied. Beside the assumptions and the requirements, the approach uses the workflow specification S and the monitored data M to identify adaptation needs.
Requirements/Constraints	The approach assumes that SLAs exhibit reliable QoS-values in order to determine the adaptation needs.

Refined SBA life-cycle

Life-cycle activities	• *Requirements Engineering*: formalisation of the KPIs/SBA requirements • *Design*: creation of the BPEL workflow • *Realization*: SLA negotiation with bound services • *Deployment & Provisioning*: deployment check;

	the assumptions and the workflow specification is checked against the application formalized requirements • *Operation & Management*: monitoring of single SBA instances; check if the assumptions are violated • *Identify adaptation needs*: execution of the SPADE check in order to determine requirement violations

Open problems and possible extensions

Open problems and extensions	The present approach does not covered all possible control constructs available to build SBAs (such as loops and forks).

There are many techniques which monitor individual services. However, using these techniques to trigger adaptations is prone to false positives. As an example, let us assume that the invocation of a service B takes 450 ms instead of an expected maximum of 400 ms. This failure is observed by means of monitoring and leads to an adaptation of the SBA. However, the overall response time would still have met the required end-to-end response time even if no adaptation would have been performed. In this case an adaptation was triggered although it was not necessary, i.e. monitoring led to a false positive adaptation trigger.

The present approach suggests a process model along the phases of the SBA life-cycle model as elaborated by the S-Cube. For an SBA to quickly and dynamically adapt to changes in service quality, the relevant artefacts, as well as the properties of the SBA and context are formalized during its design time to make them amenable to automated checks and decisions during run-time. If an assumption A (derived from an SLA) is violated, the approach checks, whether the requirements R are still satisfied. Beside the assumptions and the requirements, the proposed approach uses the workflow specification S and the monitored data M to identify adaptation needs.

Extended abstract

5 Context-based Adaptation and Monitoring

5.1 Application Model

In this scenario we primarily focus on SBAs, in which the context of those SBA plays the key role in the various activities across the SBA life-cycle. A typical example of such applications is the one of the user-centered applications based on services and service compositions. In this application, a user is equipped with the possibility to perform different actions realized as services or service compositions, that should be customized to accommodate the specific context of the user, being his physical context (e.g., location) preferences (e.g., his role), and operational settings (specific services available). Very often these applications are made available on top of mobile devices, thus bringing additional infrastructural dimension to the SBA stack.

Along with the typical element of SBA that include the model of processes available, the realizing service compositions, and the infrastructure where the application is running, the key component of the application model refers to the SBA context and how this context influences the SBA functioning and management. A context model may capture all the dimensions that could trigger adaptation; six dimensions were identified to fulfil this need: Time, Ambient, User, Service, Business and Computational [24]. Moreover, the context model may be traversal with respect to the application model and may capture the properties of different functional SBA layers.

5.2 Adaptation and Monitoring Problem

The distinguishing factor of the above application refers to the fact that these applications operate in continuously changing environments. That is, the same application shall operate differently for different user roles with different preferences, and should run on different devices and consider different services available, etc. Indeed, these changes may require the adjustment of the application to better fit the new settings. But besides requiring the application to be context-aware and to adapt the changes in the context, the way the application is managed should also take into account different settings. In particular, the way the application is monitored and adapted (in case of failures, changing user requirements, etc.) should also follow the changes in the context in the SBA

Context-driven monitoring refers to the following problem. When context changes, the way the SBA is monitored may change as well, since the new settings may require, e.g., specific information to be collected or as different elements become part of it. To accomplish this it is necessary to *be able to adapt the monitoring specification in reaction to context changes.*

Context-driven adaptation refers to the fact that same adaptation is made differently in different contexts. For example, the service selection performed when an SBA user deals with his personal activities (e.g., select cheaper hotel/flight services while organizing personal trip) it is different from corporate activities (select more reliable and faster services). The issue here is to *be able to identify proper adaptation strategies and capabilities when the SBA context changes.*

Different strategies and mechanisms could be enacted, depending on the context change occurred; some patterns should be defined in order to define such relationships. What has to be done is to define the adaptation strategy selection mechanisms as well as the adaptation mechanisms themselves that explicitly take into account the contextual changes at different functional layers and across them.

5.3 Overall Approach

The approach exploited within this integrated scenario relies on the use of templates that characterize the monitoring and adaptation activities in general settings, which are then instantiated in different way for the specific contexts. For example in case of monitoring specification the templates dictate the generic rules that are parametric with respect to the context properties [26,27]. Given the concrete context, the specific template that fits better the concrete environment is instantiated. Similarly, the

adaptation rules may be specified with the generic patterns that are conditional with respect to the contextual properties [24].

More specifically, for the adaptation of monitoring specification, the approach is to define and exploit monitoring templates that characterize the monitoring problem in general terms, and that can be instantiated in a specific way in different contexts. The key challenge here is to support and automate the process to reduce to a minimum the adaptation effort.

In case of adaptation we consider the generic framework presented in [24]. In this framework, different adaptation strategies are explicitly associated to the different context dimensions and properties, thus defining which form of adaptation are allowed or preferred in case of different contextual changes. This model may be refined by explicitly associating the adaptation activities (i.e., the adaptation template) with the specific context properties (i.e., specific user role, device, preferences, locations, etc.).

5.3.1 Refined Architecture

The overall architecture is represented in Figure 7.

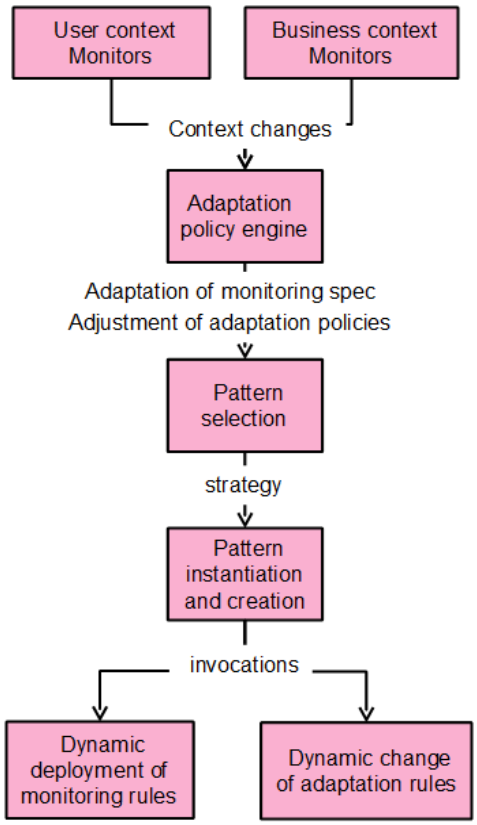

Figure 7 Approach along the S-Cube Life-Cycle

As is clear from Figure 7, the main components of the architecture are the following:

- ***Monitors** are* needed to observe the status of the application and to detect changes in the user and business context.

- ***Adaptation policy engine:*** Events are generated when context changes are identified and then they are sent to the adaptation policy engine.

- ***Pattern selection and instantiation:*** Context changes could lead to adaptation in the monitoring specifications, since the way the SBA is monitored may change or in the

application behaviour, since the way the SBA behaves needs an adaptation. If an adaptation is needed, the related pattern have to be selected. In particular, triggers could be enacted and an adaptation strategy could be chosen according to a certain event. Different strategies could be defined and the choice of the best strategy is led to an adaptation reasoner.

- ***Adapation engine:*** the adaptation mechanism related to the context changes should be enacted.

5.3.2 Refined SBA Life-Cycle

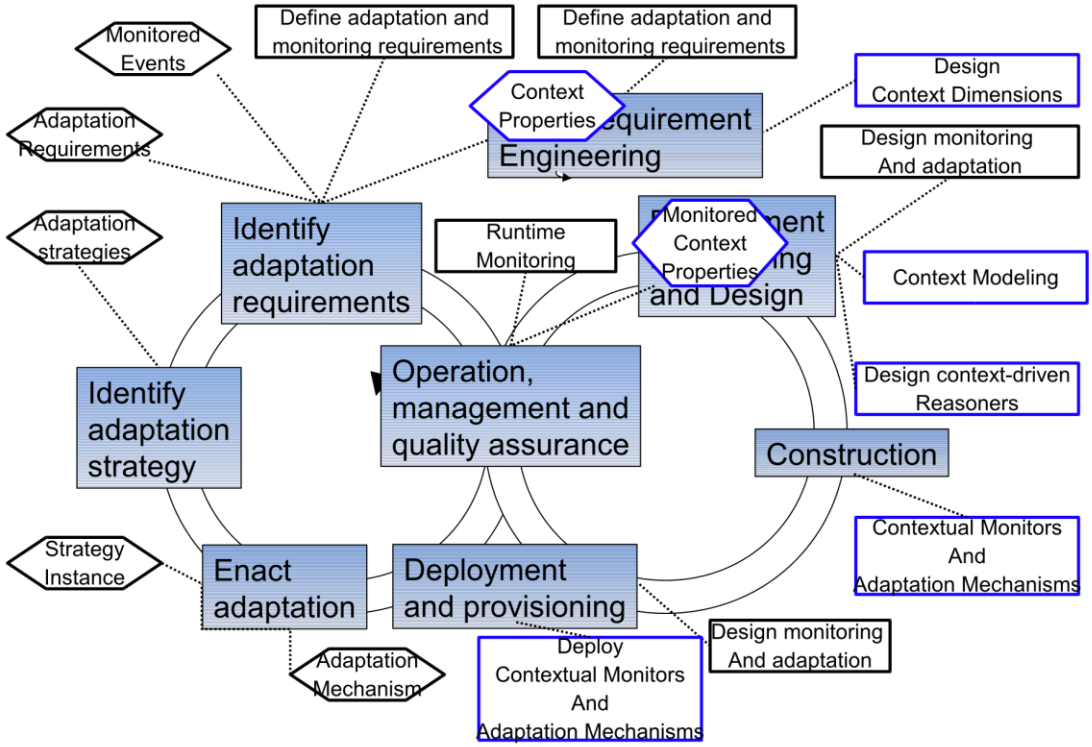

Figure 8 Approach along the S-Cube Life-Cycle

The proposed scenario fits well on the SBA life-cycle proposed in S-Cube as it covers all the phases of the lifecycle and includes several stakeholders. Each phase is characterized by different stakeholders having different goals.

In the ***Requirement Engineering and Design*** phase a user-centric context model is defined that attempts to capture all the main factors able to define the context for a SBA; in particular the context model aims at formalizing the user and application context. The dimension so the context model covers all the aspects that could be useful for the definition of the status of the application.

Information about the context could be exploited during the lifecycle in order to decide if an adaptation action is needed or necessary; this is made possible by the design of context monitor and reasoner. Besides the definition of a context model, monitoring patters are defined in such phase defining how monitoring specification could change on the basis of the context.

During the ***Construction*** phase, contextual monitors and adaptation mechanisms are developed and then deployed in the ***Deployment & Provisioning*** phase.

During the ***Operation and Management*** phase the application context is monitored in order to decide if an evolution or adaptation activity needs to be performed: adaptation could impact on the application or on the monitoring rules. If a context change is detected, it has to be checked if the application, the monitoring specification and the monitoring data fulfill the requirements and if the context change needs an adaptation (in the ***Identify Adaptation Needs*** phase). Then, during the phase of ***Identify Adaptation Strategy*** a suitable adaptation strategy has to be chosen and finally the adaptation mechanism should be enacted (in the ***Enact Adaptation*** phase).

5.4 Individual Contributions

5.4.1 A Pattern-based Approach for Monitor Adaptation

Title	A Pattern-based Approach for Monitor Adaptation
Authors	R. Contreras, A. Zisman
Type	Model
Short Description	The work presents a pattern-based approach to support monitor adaptation: adaptation of monitor rules used by a monitor tool. The work focuses on monitor adaptation due to changes in the context characteristics of the user interacting with the system and the set of services used by the system. The monitor adaptation is based on the use of patterns represented in Event Calculus for different user context types including: role, skills, needs, preferences, and cognition, as well as location, time, and environment.

Adaptation and Monitoring Problem

Contribution to the adaptation problem	With respect to the adaptation problem of the scenario the work focuses on the adaptation of the monitor driven by user context types, user interaction, and changes in the service composition.
Contribution to the monitoring problem	Identification of user context types. Ontology and XML-based user models to represent HCI context types. Definition of patterns for specifications of monitor rules

Refined Architecture

Architecture elements	--
Requirements/Constraints	Monitor Rules and Patterns should be expressed in Event Calculus The approach relies on the use of patterns for the different context types There should an event representing the interaction of the user with the system There should be models to represent user information

Refined SBA life-cycle

Life-cycle activities	□ *Requirements Engineering*: Context types definition, User Context Ontology □ *Construction*: creation of patterns for different user context types

Open problems and possible extensions

Open problems and extensions	□ Support automatic identification of monitor rules, modifications of monitor rules, and creation of new monitor rules □ Acquisition of user context types

Extended abstract

In this work we present a pattern-based approach to support monitor adaptation, i.e. adaptation of the monitor rules used by the monitor system. More specifically, our approach is concerned with HCI-aware monitor adaptation in which changes in the monitor rules are based on user's interaction with a service-based system and different types of user context. The user context information includes role, skills, needs, preferences, and cognitive information of the user represented in user models based on an ontology we have created.

There are very few approaches for service-based system monitoring and adaptation that consider user-based context. The ontology presented in our work aims to represent the various types of user context. From the ontology it is also possible to represent XML-based user models to support description of the characteristics of the users. We have additionally created patterns for monitor rules representing the different user context types. These patterns are used as templates for the monitor rules.

5.4.2 Identifying, Modifying, Creating, and Removing Monitor Rules for Service Oriented Computing

Title	Identifying, Modifying, Creating and Removing Monitor Rules for Service Oriented Computing
Authors	R. Contreras, A. Zisman
Type	Technique / Framework / Experimental Evaluation
Short Description	The work presents a pattern-based HCI-aware monitor adaptation framework to support identification, modification, creation, and removal of monitor rules. In the framework, changes in the monitor rules are based on user's interaction with a service-based system and different types of user context such as role, skills, cognition, needs, and preferences. The work also presents the experimental evaluation of the approach.

Adaptation and Monitoring Problem

Contribution to the adaptation problem	Adaptation of monitoring component in reaction to user context changes
Contribution to the monitoring problem	Identification, modification, creation and removal of monitor rules

Refined Architecture

Architecture elements	□ *Rule Adaptor*: responsible for the identification, modification, creation, and removal of monitor rules □ *Path Identifier:* retrieves the parts in the specification that are related to the context type □ *Rule Verifier*: verifies if an existing rule in the repository is still a valid rule for a service-based system □ *Monitor*: uses rules to verify the service-based system
Requirements/Constraints	Patterns and Rules are expressed in Event Calculus Service based system specifications are expressed in BPEL

Refined SBA life-cycle

Life-cycle activities	□ *Requirements Engineering*: Context types definition □ *Construction*: definition of patterns, path identifier mechanism, and rule adaptation and verification mechanisms. □ *Deployment:* identification of the path in the BPEL specification of the service-based system

Open problems and possible extensions

Open problems and extensions	□ Extension of the set of patterns, this can also include the automatic generation of patterns □ Integration of monitor component(s) □ Acquisition of user context

Extended abstract

This paper presents and approach for monitor adaptation, more specifically, the identification, modification, creation, and removal of monitor rules. It has been observed that in service-based system monitoring approaches dealing with rules, or properties, for verifying the correct behaviour of the system, assume monitor rules to be known in advance and previously defined.

Due to the dynamic nature of service-based systems, different types of user interaction, and different types of user context involved in the execution of a system it is necessary to have ways to dynamically identify. Modify, create or remove obsolete monitor rules. This work deals with the above problem by identifying, modifying, creating, and removing monitor rules. Our work relies on a pattern-based approach. In the work, patterns represent the templates for the monitor rules and are used to match potentially useful monitor rules from a repository. Initial experimental results show that our monitor adaptation approach is able to identify, modify, create and remove monitor rules from a repository in different scenarios, including an empty repository and a repository with several useful and dummy rules.

5.4.3 A Context-driven Adaptation Process for Service-based Applications

Title	A Context-driven Adaptation Process for Service-based Applications
Authors	A. Bucchiarone, R.Kazhamiakin, C.Cappiello, E. di Nitto and V.Mazza
Type	Methodology
Short Description	The work focuses on the role of the context in the adaptation activities. It proposes a framework to support the design of SBAs that targets the adaptation requirements raised by context changes.

Adaptation and Monitoring Problem

Contribution to the adaptation problem	With respect to the adaptation problem of the scenario the work focuses on the adaptation in service composition driven by changes in the context. For changes in context dimensions, suitable adaptation strategies are defined.
Contribution to the monitoring problem	

Refined Architecture

Architecture elements	• *Context monitor*: Context is modeled and monitored during the application execution in order to detect changes in the context that could need application adaptation. • *Adaptation Reasoner*. Context changes are related to adaptation strategies that could be associated. For a context change more than one strategy could be defined: the choice of strategy is led to the adaptation reasoner. • *Adaptation enactment engine:* Once chosen the strategy the adaptation mechanism should be enacted.
Requirements/Constraints	The approach relies on the capability to properly formalize and model the context for a service based application.

Refined SBA life-cycle

Life-cycle activities	□ *Requirements Engineering*: Definition of the context model for the service based application. □ *Construction*: definition of contextual monitors and adaptation mechanisms. □ *Deployment and Provisioning* deployment of contextual monitors and adaptation mechanisms. □ *Operation, Management, and QA:* monitoring of contextual properties. □ *Identify adaptation needs*: Define adaptation and monitoring requirements. □ *Identify adaptation strategy:* identification of adaptation strategies related to the context changes. □ *Enact adaptation*: enacting of the related adaptation mechanism.

Open problems and possible extensions

Open problems and extensions	Definition of a context reasoner. Context modeling. Relationships among adaptation strategies and adaptation mechanisms.

Extended abstract

The paper focuses on the role of the context in the adaptation activities for a service based application. It proposes a framework to support the design of SBAs that targets the adaptation requirements raised by context changes.

The approach is based on the service life-cycle proposed in S-Cube that emphasizes the relevance of the context elements in the different facets of adaptation, both during the design phase and at run-time. The paper considers all the issues related to the design of SBAs able to evolve together with the requirements and the execution context. In general, context is any information that can be used to characterize persons, places or objects that are considered relevant to the interaction between a user and the application The context has been modeled by considering a set of all the dimensions that can generally influence the system behaviour.
Context for a service based application should take into account dimensions related to the User context (preferences and settings of the user), the Ambient context (factors related to space and location), the Business context (business goals of the application), Time context (factors related to the time, season), service context (factors related to the services composing the application) and finally the Computational Context (physical devices used to invoke the application).

On the basis of this context model, the proposed approach provides guidelines for the identification of the relevant context dimensions to monitor (not all the dimensions are meaningful for all the applications) and for the definition of the adaptation triggers able to link context changes with suitable adaptation strategies.
In particular the focus of the paper is on how and when the context should be *defined*, how context should be *exploited* and *evolved* and, finally, what is the *impact* of the context changes on the adaptation activities.

The main contribution to the scenario is the capability of the context-driven adaptation process to capture the key aspects of adaptation and support designers from the requirements elicitation to the construction of proper adaptation mechanisms.

5.4.4 Modelling and Automated Composition of User-Centric Services

Title	Modelling and Automated Composition of User-Centric Services
Authors	Raman Kazhamiakin, Massimo Paolucci, Marco Pistore and Heorhi Raik
Type	Model, Technique

Short Description	The work contributes to the problem of context-driven SBA adaptation. Specifically, it provides an automated support for construction of service compositions, taking into account the specific user context, such as the tasks the user does, his preferences and decisions regarding those tasks and the services to be involved. To accomplish this, the work provides a way to capture the composition templates, exploited by the user to solve different tasks, and a technique to automatically instantiate those templates at run time.

Adaptation and Monitoring Problem

Contribution to the adaptation problem	The work contributes to the problem of selecting proper adaptation strategies in different user contexts: based on a set of predefined task composition templates, the approach is able to instantiate on the fly those templates with concrete services and construct an executable service composition to perform the SBA adaptation.
Contribution to the monitoring problem	-

Refined Architecture

Architecture elements	• P*attern instantiation and creation:* based on the specific adaptation need and the specific current context, the composition template is instantiated. Specifically, the automated service composition is created out of the template and the services associated to the context.
Requirements/Constraints	The approach relies on the ability to monitor and reason on the user context to drive the adjustment and instantiation of the templates.

Refined SBA life-cycle

Life-cycle activities	• *Development and Design*: definition of composition templates. Definition of means to associate and select services for specific templates in different contexts. • *Identify adaptation strategy*: instantiate composition templates based on the user context and associated services.

Open problems and possible extensions

Open problems and extensions	Support for the user to interactively influence the adaptation process: to define which services should be selected and to adjust/modify the templates.

Extended abstract

This work addresses the problem of user-centric service composition, that is, how, based on the needs, preferences and activities of the user, to construct the specific service composition. Such a problem requires purely run-time realization, as the specific combination of the user tasks, the preferred services and the services available for the adaptation cannot be anticipated at design time. Furthermore, the involved services may differ in the protocols and the activities involved, which requires very flexible approach.

In this work we provide two contributions to address this problem. First, we present a novel model to capture the composition templates: a high-level characterization of the set of eventual services, their relations and constraints to be satisfied to solve a specific user task. For example, for the task of organizing and managing a trip, a template to coordinate the overnight and transport reservations is associated, where it is defined how the two types of services should be aligned. The template, therefore, characterizes the service types and their possible evolution, the constraints on the coordination of the services, and the user involvement in the control over the services.

Second, we propose a technique for automatically creating a concrete composition out of the template given the user context. Specifically, automated service composition approach is used to generate a concrete composition for the template and for the services associated to the user context. For example, in case of business trip of a long distance performed by a specific user that prefers Lufthansa flights and Accor hotels, the above template is instantiated with the Lufthansa booking, cancellation, and flight status monitoring service, as well as the services for booking and managing Accor hotels reservation which correspond to the transportation and overnight service type in the template. Furthermore, the composition automatically builds the necessary user protocol to control and coordinate activities: to be aware of events, to provide inputs, and to trigger actions when the user changes the tasks to do (e.g., wants to cancel the trip).

6 Conclusions

To conclude, we here present a summary of the research directions followed by the project members in order to achieve the research tasks of the work-package.

In this deliverable we have reported on the work by the Network to further develop comprehensive and integrated SBA adaptation and monitoring framework, focusing on the problem of proactive adaptation at different layers and in settings. The ability to anticipate the need for adaptation is a key issue since it enables one to prevent the failures and deviations that are very critical in the open world of SBAs. Similarly, there is a need to consider the context of the SBA in a holistic manner and to provide integrated approaches over context-driven SBA monitoring and adaptation. In order to accomplish this, we introduced three integration scenarios that follow common architecture, but which address specific problems.

Each integration scenario identified and described in this deliverable focuses on a specific challenge identified in the workpackage WP-JRA-1.2. In particular, Scenario 1 primarily focuses on the problem of integrating the adaptation and monitoring aspects across different layers having the quality model of the SBA as the key driving aspect Scenario 2 addresses the proactive adaptation challenge providing an approach based on explicit modelling and monitoring of various assumptions; and Scenario 3 focuses on context- and HCI-aware SBA monitoring and adaptation.

The scenarios provide a common reference for the adaptation and monitoring problem in hand, providing a common basis for the different research works developed within the project. For each of those scenarios we presented the approach used by the scenario to resolve the problem within and across different functional layers, the refined reference architecture, the description of the specific activities across the SBA life-cycle. Finally, for each scenario we analyzed the contributions of the partners onto the scenario defining the role of the approach, the relevant architecture components and the life-cycle activities. An overview of the partner contributions can be found in Table 1.

An important contribution of the deliverable is the extension of the Integrated Research Framework, developed in the workpackage IA-3.1, providing the model and instantiation of the cross-cutting integration of research results. These scenarios contribute to different views of the IRF (namely architectural and the life-cycle) and to define the common interfaces between different cross-layer components.

Starting from this common basis the integration scenarios open up the further integration of different principles, techniques and mechanisms through common interfaces and architecture mechanisms.

We believe that the definition of integration scenarios, allowing a clear definition of the relations and dependencies between the different monitoring and adaptation approaches, is a key factor for the identification of the research gaps that should be leading concerns of future deliverables, and thus plays a fundamental role in construction of a comprehensive, holistic adaptation and monitoring framework being developed in JRA-1.2.

Scenario	Title	Authors	Type	Status
Quality-driven Multilayer SBA Monitoring and Adaptation	*A Fuzzy Service Adaptation based on QoS Satisfaction*	B. Pernici and S. H. Siadat	Conference paper	Accepted CAiSE 2011
	Selection of Service Adaptation Strategies Based on Fuzzy Logic	B. Pernici and S. H. Siadat	Conference paper	Submitted ICWS 2011
	Preventing KPI Violations in Business Processes based on Decision Tree Learning and Proactive Service Substitution	B. Wetzstein, A. Zengin, R. Kazhamiakin, M. Pistore, D. Karastoyanova, F. Leymann	Conference paper	Submitted SCC 2011
	LAYSI: A Layered Approach for SLA-Violation Propagation in Self-manageable Cloud Infrastructures	Ivona Brandic, Vincent C. Emeakaroha, Michael Maurer, Sandor Acs, Attila Kertesz, Gabor Kecskemeti, Schahram Dustdar	Workshop paper	In Proc. COMPSAC 2011
	A Soft-Constraint Based Approach to QoS-Aware Service Selection	M.A.Zemni, S.Benbernou, M.Carro	Conference Paper	In Proc. ICSOC. 2010,
	Model-driven Management of Services	Luciano Baresi, Mauro Caporuscio, Carlo Ghezzi, and Sam Guinea	Conference Paper	In Proc ECOWS 2010
Assumption-based Proactive Monitoring and Adaptation	*Proactive SLA Negotiation for Service Based Systems*	Khaled Mahbub and George Spanoudakis	Conference Paper	In Proc. SERVICES 2010
	Evolving Services from a Contractual Perspective	V.Andrikopoulos, S.Benbernou, M.P.Papazoglou	Conference Paper	In Proc. CAiSE 2009
	Adaptation of Service-based Business Processes by Context-Aware Replanning	Antonio Bucchiarone, Raman Kazhamiakin, Marco Pistore and Heorhi Raik	Conference Paper	Submitted ICWS 2011
	Towards Proactive Adaptation: A Journey along the S-Cube Service Life-Cycle	Andreas Metzger, Eric Schmieders, Cinzia Cappiello, Elisabetta Di Nitto, Raman Kazhamiakin, Barbara Pernici and Marco Pistore	Workshop Paper	In Proc. MESOA 2010

Context-based Adaptation and Monitoring	*A Pattern-based Approach for Monitor Adaptation*	R. Contreras, A. Zisman	Conference Paper	In Proc. ICSTE 2010
	Identifying, Modifying, Creating and Removing Monitor Rules for Service Oriented Computing	R. Contreras, A. Zisman	Workshop paper	In Proc. . PESOS 2010
	A Context-driven Adaptation Process for Service-based Applications	A. Bucchiarone, R.Kazhamiakin, C.Cappiello, E. di Nitto and V.Mazza	Workshop paper	Accepter. . PESOS 2011
	Modelling and Automated Composition of User-Centric Services	Raman Kazhamiakin, Massimo Paolucci, Marco Pistore and Heorhi Raik	Book Chapter	OTM 2010

Table 1

7 References

[1] 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.

[2] Elisabetta di Nitto, editor. State of the art report on software engineering design knowledge and Survey of HCI and contextual Knowledge, July 2008. S-Cube project deliverable: PO-JRA-1.1.1. http://www.s-cube-network.eu/achievements-results/s-cube-deliverables.

[3] Raman Kazhamiakin, editor. 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.

[4] Raman Kazhamiakin, Marco 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.

[5] Andreas Gehlert, Marco Pistore, Pierluigi Plebani, Loredana Versienti, editors. First Version of Integration Framework, December 2009. S-Cube project deliverable: CD-IA-3.1.3. http://www.s-cube-network.eu/achievements-results/s-cube-deliverables.

[6] Andreas Gehlert and Andreas Metzger, editors. Quality Reference Model for SBA, March 2008. S-Cube project deliverable: CD-JRA-1.3.2. http://www.s-cube-network.eu/achievements-results/s-cube-deliverables.

[7] 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.

[8] Branimir Wetzstein, Asli Zengin, Raman Kazhamiakin, Marco Pistore, Dimka Karastoyanova, "Preventing KPI Violations in Business Processes based on Decision Tree Learning and Proactive Service Substitution", SCC 2011 (under submission).

[9] Barbara Pernici and S. Hossein Siadat, "A Fuzzy Service Adaptation for Service Based Applications", CAiSE 2011.

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

[11] OASIS Standard: Web services business process execution language (BPEL) version 2.0, 11 April 2007.

[12] Oriol, Marco, Franch, Ameller. Monitoring Adaptable SOA-Systems using SALMon. In Proc. Of 1st Intl. workshop on Monitoring, Adaptation and Beyond (MONA+), Collocated with ServiceWave'08, 2008.

[13] Luciano Baresi, Sam Guinea, Michele Trainotti, and Marco Pistore. Dynamo + ASTRO: An integrated approach for BPEL monitoring. In ICWS'09, 2009.

[14] Andreas Metzger, Eric Schmieders, Cinzia Cappiello, Elisabetta Di Nitto, Raman Kazhamiakin, Barbara Pernici and Marco Pistore, Towards Proactive Adaptation: A Journey along the S-Cube Service Life-Cycle, Workshop paper MESOA 2010.

[15] Vasilios Andrikopoulos, Salima Benbernou, and Mike P. Papazoglou, "Evolving Services from a Contractual Perspective", CAiSE 2009.

[16] Antonio Bucchiarone, Raman Kazhamiakin, Marco Pistore, and Heorhi Raik, "Context-driven adaptation of Service–Based Business Processes", ICWS 2011 (under submission)

[17] Gehlert et al. Exploiting Assumption-Based Verification for the Adaptation of Service-Based Applications. SOAP@SAC 2010.

[18] Khaled Mahbub and George Spanoudakis, "Proactive SLA Negotiation for Service Based Systems". SERVICES 2010.

[19] Khaled Mahbub and George Spanoudakis, "A Framework for Proactive SLA Negotiation", 5th International Conference on Software and Data Technologies (ICSOFT 2010), 22 - 24 July, Athens, Greece

[20] Luciano Baresi, Mauro Caporuscio, Carlo Ghezzi, and Sam Guinea, "Model-driven Management of Services", In Proc. ECOWS'10, 2010.

[21] Mohamed Anis Zemni, Salima Benbernou, and Manuel Carro, "A Soft-Constraint Based Approach to QoS-Aware Service Selection", In Proc. ICSOC'10, 2010.

[22] M.L. Massie, B.N. Chun and D.E. Culler, The ganglia distributed monitoring system: design, implementation, and experience, Parallel Computing **30** (July) (2004), pp. 817–840

[23] NESSI Roadmap Series Document: NESSI Strategic Research Agenda. NESSI Research Priorities for FP7, May 2009, SRA Vol.3.2, Revision 2.0, 10 May, 2009.

[24] A. Bucchiarone, R. Kazhamiakin, C. Cappiello, E. di Nitto and V. Mazza "A Context-driven Adaptation Process for Service-based Applications", in Proc. PESOS workshop, 2010.

[25] R. Contreras and A. Zisman, "A Pattern-based Approach for Monitor Adaptation". In Proc. ICSTE 2010.

[26] R. Contreras and A. Zisman, " Identifying, Modifying, Creating, and Removing Monitor Rules for Service Oriented Computing", Accepted in Workshop PESOS 2011

[27] J. Ejarque, A. Micsik, R. Sirvent, P. Pallinger, L. Kovacs, R. M. Badia. Semantic Resource Allocation with Historical Data Based Predictions. The First International Conference on Cloud Computing, GRIDs, and Virtualization, CLOUD COMPUTING 2010, November 21-26, 2010 - Lisbon, Portugal

[28] D. Ameller, X. Franch. Service Level Agreement Monitor (SALMon), in 7th International Conference on Composition-Based Software Systems, Madrid, 2008. ICCBSS 2008.

[29] Waikato Environment for Knowledge Analysis, WEKA, http://www.cs.waikato.ac.nz/ml/weka/

[30] B. Pernici and S. H. Siadat. "Selection of Service Adaptation Strategies Based on Fuzzy Logic", ICWS2011 (under submission).

[31] Ivona Brandic, Vincent C. Emeakaroha, Michael Maurer, Sandor Acs, Attila Kertesz, Gabor Kecskemeti, Schahram Dustdar. "LAYSI: A Layered Approach for SLA-Violation Propagation in Self-manageable Cloud Infrastructures", In Proc. COMPSAC 2011.

[32] Raman Kazhamiakin, Massimo Paolucci, Marco Pistore and Heorhi Raik "Modelling and Automated Composition of User-Centric Services". OTM 2010.