The University of Sheffield
South-East European Research Centre

DEPARTMENT OF COMPUTER SCIENCE

PhD Transfer Report

Towards Engineering Multi-Layer Monitoring and Adaptation of Service-Based Applications

Konstantinos Bratanis

June 2012

Supervisors
Dr. Dimitris Dranidis
Dr. Anthony J.H. Simons

Internal Advisor
Dr. George Eleftherakis
Abstract

This report discusses the topic of monitoring and adaptation of service-based applications and it explains the problem of implementing multi-layer monitoring and adaptation (multi-layer M&A) in such applications. The report presents a comprehensive literature review of the current landscape of monitoring and adaptation approaches for service-based applications, while it discusses the current state of the art in multi-layer M&A. Based on the literature review, the report argues that the current state of the art in multi-layer M&A is significantly limited due to the lack of a concrete framework for architecting flexible, extensible, dynamic, and effective multi-layer M&A solutions. It proposes to address this gap through the investigation of a framework for integrating and coordinating diverse monitoring and adaptation mechanisms. The report sets the concrete research goals and objectives of the proposed PhD project. Finally, it reports the current progress of the author and it proposes plan for pursuing the proposed objectives in the remained of the project.
## Table of Contents

1 **Introduction** ................................................................................................................................. 5  
   1.1 Structure of the Report .............................................................................................................. 7 

2 **Fundamentals of Monitoring and Adaptation of Service-Based Applications** .......... 8 
   2.1 Fundamentals of Service-Oriented Computing ........................................................................ 8  
   2.1.1 Key Characteristics of Service-Oriented Systems .............................................................. 8  
   2.1.2 Web services ....................................................................................................................... 9  
   2.2 Fundamentals of Autonomic Computing .................................................................................. 12 
   2.2.1 Key Characteristics of Autonomic Systems ....................................................................... 12  
   2.2.2 MAPE-K Loop ....................................................................................................................... 13  
   2.2.3 Self-* Properties of Autonomic Systems ........................................................................... 15  
   2.3 Service-Based Applications ..................................................................................................... 16  
   2.4 Adaptable Service-Based Applications .................................................................................... 18  
   2.5 Monitoring Process in Service-Based Applications ............................................................... 20  
   2.6 Adaptation Process in Service-Based Applications ............................................................... 22  
   2.7 Multi-Layer Monitoring and Adaptation Process in Service-Based Applications ............ 24 

3 **State of the Art on Monitoring and Adaptation of Service-Based Applications** .. 27  
   3.1 Monitoring and Adaptation in the Business Process Management Layer ....................... 29 
   3.1.1 Research Efforts on Monitoring Approaches .................................................................... 29  
   3.1.2 Research Efforts on Adaptation Approaches .................................................................... 31  
   3.2 Monitoring and Adaptation in the Service Composition and Coordination Layer .......... 33 
   3.2.1 Research Efforts on Monitoring Approaches .................................................................... 33  
   3.2.2 Research Efforts on Adaptation Approaches .................................................................... 37  
   3.3 Recent Research Efforts on Multi-Layer Monitoring and Adaptation ............................... 41 
   3.3.1 Research Efforts on Diverse Aspects of Multi-layer Monitoring and Adaptation .......... 42  
   3.3.2 Ad-hoc Integrations of Monitoring and Adaptation Processes ....................................... 43  
   3.3.3 Research Efforts on Architectural Aspects of Multi-layer Monitoring and Adaptation ... 44  
   3.3.4 Limitations of Current State of the Art in Multi-layer Monitoring and Adaptation ....... 45 

4 **Towards Engineering the Integration and the Coordination of Monitoring and**  
   **Adaptation Processes** .................................................................................................................... 47 
   4.1 Integrating Diverse Monitoring and Adaptation Processes ................................................ 48  
   4.2 Conveying Information among Monitoring and Adaptation Processes .............................. 49  
   4.3 Coordinating Monitoring and Adaptation Processes............................................................. 49  
   4.4 Research Goal and Objectives ................................................................................................. 50  
   4.5 Novelty of the Proposed Research Directions and Potential Areas of Contribution .. 51 

5 **Current Progress and Future Work** ............................................................................................ 53  
   5.1 Research Activities .................................................................................................................. 53  
   5.2 Dissemination Activities ......................................................................................................... 54  
   5.3 Participation in the Research Community .............................................................................. 55  
   5.4 Timetable for the Remainder of the Research Project ........................................................... 55  

References .......................................................................................................................................... 57  

Appendix I: Classification of Research Efforts on Monitoring and Adaptation of**  
**Service-Based Applications** ............................................................................................................ 68
List of Figures

Figure 1: The simplest form of SOA follows the service request/response paradigm..... 9
Figure 2: SOA triangle inspired by the initial set of Web services standards................. 10
Figure 3: The MAPE-K control loop. ................................................................. 14
Figure 4: The conceptual view of a service-based application.................................. 17
Figure 5: Conceptual framework of monitoring and adaptation process in service-based applications................................................................................. 19
Figure 6: Modified conceptual framework of monitoring and adaptation process to support multi-layered monitoring and adaptation of service-based applications. ....... 25
1 Introduction

Almost a decade has passed since the introduction of service-oriented computing (SOC). Over this period, SOC has received an ever increasing attention from both the academia and the industry, through the availability of Web services technologies and the various standardisation efforts around SOC. In its core, SOC is a set of design principles that foster the implementation of software systems in a loosely coupled, highly reusable, highly interoperable and widely distributed manner. The main idea behind SOC is to make a feature of a system available as a reusable and remote accessible service. Following this approach, a system can be seen as a set of reusable services that can be composed to create new systems or services.

Systems build out of reusable services are named service-based (or service-oriented). Service-based systems are usually complex and harder to manage during the design-time and the run-time as well, due to a number of reasons:

1) Service-orientation fosters the implementation of highly dynamic systems, where services can be discovered, consumed, and removed at run-time;
2) Services are often developed independently from the execution context, that is, a service can be consumed in an application that was never envisaged;
3) Services can evolve without prior notification, that is, services exposing an expected behaviour through an agreed interface can have their behaviour or interface modified;
4) The quality of service is variable at run-time, that is, even if a service was performing well during design-time, it is not guaranteed that will perform the same at run-time;
5) Service-based systems operate in highly heterogeneous and dynamic execution context.

Therefore one of the most critical challenges associated with the dynamics and the complexity of service-based systems is the development of approaches for engineering self-adaptation in service-based systems, since there is high probability that the requirements satisfied during design time will be violated at run-time, ultimately leading to faults or total failure of the service-based system. As such, the ability to detect, predict, diagnose, recover, compensate, and prevent such violations has been widely acknowledged by the SOC research community as a significant and challenging problem. The neighbouring fields of Autonomic Computing and Self-Adaptive Systems target similar challenges.

Over the past decade there has been an increasing interest in incorporating monitoring and adaptation capabilities in service-based systems, motivated by the aforementioned problems. The goal of the adaptation process is to modify the service-based system, based on its own state and the state of its environment, such that it continues to satisfy its specification and requirements. The adaptation process aims to address the adaptation requirements identified through the analysis of problematic situations, which are detected by monitoring the service-based system or its context. In the event of that adaptation is required, i.e. requirements have been violated or are about to be violated, an adaptation strategy, i.e. a set of adaptation actions, has to be devised and enacted, such that the service-based system is led to a state in which it will satisfy again the desired requirements.
A particular instance of a service-based system that calls for engineering self-adaptation is the service-based applications (SBA). An SBA is a layered application that is developed by integrating multiple heterogeneous software-based services, which are usually developed, controlled and provided by different third-party organisations, not necessarily by the organisation that owns the SBA. An SBA usually incorporates a big number of services, often consumed over a network, that are used as the building blocks for performing the desired functionalities of the application. The services are used to instantiate service compositions or business processes that realise business workflows.

Over the years, a wide range of fragmented monitoring and adaptation approaches have been proposed in the literature for partially addressing very specific problems from a very particular perspective at a single layer of the SBA. The high fragmentation of the existing approaches is also confirmed by the literature review included in this report in Section 3. Relatively recently, the research community recognised that the high fragmentation of approaches does not facilitate the development of effective solutions for developing adaptable SBA. The individual monitoring and adaptation mechanisms operate in isolation and they ignore the layered nature of the SBA, therefore, leading to problems, such as the incorrect detection of adaptation requirements, or the enactment of ineffective adaptation actions.

In order to address the aforementioned multifaceted problem, the research community proposed to investigate new directions towards multi-layer (or cross-layer) monitoring and adaptation (multi-layer M&A) of SBA. Multi-layer M&A is concerned with the integration of monitoring and adaptation mechanisms targeting multiple aspects across the layers of an SBA. Despite these proposed directions, the current state of the art in multi-layer monitoring and adaptation of SBA has demonstrated few multi-layer M&A solutions, which have been based on the ad-hoc integration of pre-existing monitoring and adaptation mechanisms. Such efforts realised multi-layer M&A from a particular perspective and they addressed a very specific problem. We argue that the current state of the art in multi-layer M&A is significantly limited due to the hardwired approach to integration of mechanisms and their uncoordinated operation in the existing multi-layer M&A solutions.

Given the above, we propose to investigate an engineering framework for developing multi-layer M&A approaches that are flexible, extensible, dynamic, and effective. The framework will address three main problems that call for a concrete solution. The first problem is how to engineer the integration of diverse mechanisms and to ensure the dissemination of information among them. The second problem is what information needs to be conveyed among the mechanisms and how this information is standardised. The last problem is how to coordinate the operation of the integrated mechanisms, such that they operate in an effective way. Addressing all three problems would provide the baseline for architecting flexible, extensible, dynamic, and effective multi-layer M&A solutions for service-based applications.

The subsequent sections provide a brief introduction to the topics of service-oriented computing and autonomic computing, in order to facilitate a basic understanding of the background of the report. The last section of the introduction explains the contents and the structure of the report.
1.1 Structure of the Report

The report follows a sequential flow comprising four major sections. The report is organised as follows:

Section 2 provides an introduction to the fundamentals of monitoring and adaptation of service-based applications. In the first subsections, the fundamentals of the fields of service-oriented computing and autonomic computing are introduced. Next, it explains the characteristics of service-based applications and the necessity to engineer adaptation capabilities into them. The section explains the concepts associated with the monitoring and adaptation processes, while it discusses about the necessity for realising multi-layer monitoring and adaptation in service-based applications.

Section 3 presents the current landscape of research efforts addressing the subject of monitoring and adaptation of service-based applications, demonstrating the high fragmentation of existing approaches. It discusses the classification of the existing monitoring and adaptation approaches for two layers of a service-based application. The section discusses the current state of the art in multi-layer monitoring and adaptation, while it also explains the current limitations.

Section 4 proposes concrete research directions aiming to advance the existing state of the art in multi-layer monitoring and adaptation, through the development of a framework for the integration and the coordination of diverse monitoring and adaptation mechanisms. This section sets the research goals and objectives of the proposed PhD project.

Section 5 reports the current progress of the author and it proposes plan for pursuing the proposed objectives in the remained of the project.

The report includes in Appendix I a classification of the existing monitoring and adaptation approaches for service-based applications.
2 Fundamentals of Monitoring and Adaptation of Service-Based Applications

Monitoring and adaptation of service-based applications (SBA) aims at the identification of requirements concerned with various aspects of the SBA and the means to address the new requirements by modifying the SBA, preferably in an automated manner. Various concepts for monitoring and adaptation have been developed by different research efforts throughout the past decade. Currently, there exists a more comprehensive and integrated understanding of the concepts and the terminology associated with the field. This understanding has been mainly developed in the frame of European Network of Excellence in Software Services and Systems (S-Cube)\(^1\). Throughout this report, we have adopted the concepts and the terminology defined in the S-Cube Knowledge Model\(^2\).

Before proceeding to the fundamentals of monitoring and adaptation, the next two sections present the fundamentals of service-oriented computing and autonomic computing. This is done to familiarise the reader with the two fields, which are highly relevant. The remaining sections introduce the main concepts and the key terminology from the domain of monitoring and adaptation of SBA, with the intention to assist the reader in understanding the domain. Firstly, the concept of the service-based application is discussed, explaining what is comprised in an SBA and how it is structured. Having a clear understanding of SBA, the concepts associated with the monitoring and adaptation of SBA are explained.

2.1 Fundamentals of Service-Oriented Computing

The service-oriented computing (SOC) paradigm suggests the use of services to support the development of rapid, low-cost, interoperable, evolvable, and massively distributed software systems [1]. According to SOC, software systems should be made of basic services, which can be described, published, discovered, and loosely coupled in novel ways to support interoperability and platform-independence. The utility of each service can range from executing a very simple task to complex business processes, which can involve interaction with other services.

The following sections present in more depth the area of Service-Oriented Computing. They present the key characteristics of service-oriented systems and the enabling technologies such as Web services.

2.1.1 Key Characteristics of Service-Oriented Systems

Service-oriented systems implement a service-oriented architecture (SOA). SOA is not an explicit technical application architecture that defines the structure and the interactions between software components. SOA is an architectural style that defines a set of principles for designing and developing software systems [2]. SOA comprises the following common set of principles [2]:

1) Services are reusable: even if reusability has not been foreseen, services are designed for reuse;

\(^1\) S-Cube website - http://www.s-cube-network.eu/
\(^2\) S-Cube Knowledge Model - http://www.s-cube-network.eu/km/
2) Services share a formal contract: each service has to expose an interface that specifies its functions and defines the communication messages;
3) Services are loosely coupled: services must be designed to interacted without the need for hardcoded cross-services dependencies;
4) Services abstract underlying logic: the only visible information about the services is the contents of its contracts. The underlying implementation is invisible to service requestors;
5) Services are composable: services can compose other services, thus allowing the implementation of more complex activities;
6) Services are autonomous: the logic governed by a service resides within an explicit boundary. The service has control within this boundary and is not dependent on other services for it to execute its governance;
7) Services are stateless: services should not manage information regarding the state of a transaction, as that can increase coupling;
8) Services are discoverable: services should allow their descriptions to be discovered and understood by humans and service requestors that may be able to make use of their logic.

If the architecture of a software system has been designed and implemented with the aforementioned principles, it can then be characterised as SOA [2].

![Diagram](image.png)

**Figure 1: The simplest form of SOA follows the service request/response paradigm.**

Figure 1 shows a SOA in its most simple form. This architecture follows the request/response communication paradigm by exchanging messages between a service consumer and a service provider. The service consumer sends request messages (i.e. invocations of service operations) to the service provider, requesting a particular function to be executed on his behalf. The service provider sends response messages (i.e. the results of the service invocations) to the service consumer, providing the outcome of the execution. Usually, the service consumer and the service provider have established a well-defined contract, which describes the request and the response messages supported by the service. A widely adopted technology for realising this paradigm is Web services, which are discussed in the next section.

### 2.1.2 Web services

Web services have been the most successful technology in realising the aforementioned SOA principles implementing systems that follow the SOC paradigm. Although Web services are not automatically provide service reusability, autonomy, statelessness, and discoverability, and they require cautious modelling and design effort to support these principles, Web services are in full alignment and natively support service abstraction, compositability, loose coupling, and service contracts [2]. This natural alignment of Web services with half of the principles is an indication as to why Web services have been so successful in implementing SOA.
Figure 2 illustrates a SOA which has been inspired mostly from the Web services technology and the set of standards developed around Web services. The illustrated architecture is often referred as the “SOA Triangle” [3] and it comprises three main components, which are the consumer who uses the service, the provider who provisions the service, and the service registry that holds a directory of published services.

This model was realised by the first generation of Web services standards that comprised the Web Service Description Language (WSDL)³ for specifying the interface of a Web service, the Simple Object Access Protocol (SOAP)⁴ for communication between the service requestor and the service provider, usually over the HTTP protocol, and the Universal Description Discovery Integration (UDDI)⁵ registry for building, platform-independent, XML-based registry for discovering Web services, published by service providers. WSDL and SOAP have remained the dominant standards for developing Web services, whereas UDDI was not so successfully because of the lack of qualitative information about the published Web services and the limited text-based searching features for discovery of Web services.

![SOA Triangle](image)

Figure 2: SOA triangle inspired by the initial set of Web services standards.

The popularity of Web services led to the development of the second generation of Web services specifications, known as WS-* family of specifications⁶, which were introduced by different organisations and focused mostly on a standardising functionality required for using Web services in an enterprise context. Some notable specifications are WS-Addressing⁷ providing transport-neutral mechanisms to exchange endpoint references, WS-Policy⁸ providing a general purpose model and the corresponding syntax to describe policies of Web services and service consumers, WS-Notification⁹ that is a set of specifications to standardise the way Web services

---

³ Web Service Description Language - http://www.w3.org/TR/wsdl20/
⁴ Simple Object Access Protocol - http://www.w3.org/TR/soap/
⁵ Universal Description Discovery Integration - http://uddi.org/pubs/uddi-v3.0.2-20041019.htm
⁶ A comprehensive visualisation of Web services standards is available at http://www.innoq.com/resources/ws-standards-poster/
⁷ WS-Addressing – http://www.w3.org/standards/techs/wsaddr
⁸ WS-Policy - http://www.w3.org/TR/ws-policy/
interact using notifications or events, in order to implement event-driven architectures using Web services.

Another focus of the WS-* specifications was the use of service-orientation in Business Process Management (BPM). Two key specifications addressed the process level interactions and provided the framework for decomposing and implementing business processing using Web services. Firstly, the WS-BPEL\(^{10}\) specification introduced the Business Process Executable Language (BPEL) for specifying business processes by orchestrating Web services, which realise an activity of the process. Secondly, the WS-CDL\(^{11}\) specification introduced the Choreography Description Language (CDL) for describing peer-to-peer collaborations of participants by defining, from a global viewpoint, their common and complementary observable behaviour.

The abovementioned process specifications were developed to support the two acknowledged models concerning the flow of service invocations during the execution of a process [4]. WS-BPEL was developed to support the orchestration model, which refers to a process that involves the invocation of both internal and external services of various providers, with only one party being responsible for coordinating the interaction throughout the whole execution of the process. In contrast, WS-CDL was developed to support the choreography model, which focuses on tracking the execution of a process, while there is no control by a single party, but rather a collaboration of all peers that invoke services independently.

A gap that the business process specifications did not address was the involvement of humans in processes. The BPEL language considered the behaviour of business processes as long as the activities are realised by Web services, without considering human interactions. This gap was addressed by the introduction of two specifications, the BPEL4People\(^{12}\) and the WS-HumanTask that extended the scope of activities realised by Web services to activities realised by humans. All these specifications related to business processes provided a language capable of expressing business logic in an executable format, while leveraging service-orientation for encapsulating and abstracting the business logic from the underlying technology used for implementation.

Over the past few years, RESTful Web services have emerged as another popular and widely adopted approach for implementing SOA. RESTful Web services follow the Representational state transfer architectural style, which was introduced by R. Fielding [5]. Unlike Web services that use structured SOAP messages for communication, RESTful Web services follow the architectural style of the Web, meaning that they are identified using a Unique Resource Identifier (URI) and they have a uniform interface based on the HTTP verbs (i.e. GET, POST, PUT, DELETE). RESTful Web services use self-descriptive messages so that the representation of messages can be negotiated and the messages can be exchanged in a variety of formats. The interaction in RESTful Web services is stateless and it is necessary to perform explicit state transfer i.e. either to transfer some information for identifying the state and future states, or to transfer the whole state, in order to implement stateful interactions. RESTful Web services are more lightweight than Web services that use SOAP, because RESTful Web services leverage existing well-known Web standards (HTTP, XML, URI, MIME) and they require

---

\(^{10}\) WS-BPEL - http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=wsbpel

\(^{11}\) WS-CDL - http://www.w3.org/TR/ws-cdl-10/

infrastructure that has become pervasive, similar for building dynamic Web sites, while such services can be invoked and tested through a Web browser, without the need for developing custom client-side software [6].

### 2.2 Fundamentals of Autonomic Computing

In 2001 Paul Horn from IBM introduced the vision of autonomic computing [7] as a direction towards automating the management of computing systems and software applications, which continuously are becoming dramatically complex, and consequently difficult to manage by only relying on human intervention and administration of such complex systems. The increasing number of components and applications that interact and depend to each other increase the complexity and raise barriers for a human to provide informed decisions in a timely manner. Optimisation becomes challenging, while the diagnosis of failures go beyond the human capabilities. Ganek and Corbi [8] argued that managing complex systems has grown too costly and prone to error, and therefore complexity is becoming a significant inhibitor that threatens to undermine the future growth and societal benefits of information technology.

The idea of autonomic computing stems from the autonomic nervous system, which seamlessly manages and coordinates various vital low-level functions of the human body, such as heart rate, blood flow, and others. This way the human mind is free to focus on what the goal is rather than calculating how to implement the goal. For example, a human that runs from point A to point B, he does not have to calculate how hard his heart will work or how fast he will breathe.

In a similar way, autonomic computing is an approach to constructing computing systems and software applications that are capable to manage themselves, based on some higher-level objectives specified by an administrator [7]. Such systems are able to adapt to different situations and to prepare their resources to cope most efficiently with demanding workloads.

The main body of work in Autonomic Computing is presented in [9–12]. The following sections present in more depth the area of Autonomic Computing. They present the key characteristics of autonomic systems, the MAPE-K reference model for developing autonomic control loops in autonomic systems, and the self-* properties.

#### 2.2.1 Key Characteristics of Autonomic Systems

When Paul Horn [7] introduced the grand challenge of developing computing and software systems able to manage themselves with minimum human administration, he specified eight key characteristics that such systems need to possess. These characteristics are explained below.

1. **“To be autonomic, a system needs to “know itself” and consist of components that also possess a system identity”**. A system cannot monitor and control what it does not know that exists. Thus, an autonomic system needs fine grained knowledge of its components, the interdependencies among its components, as well as the dependencies to external systems. It needs to have detailed knowledge of its current status, capacity, and capabilities, while it needs to know the resources it owns and which resources can be shared or have to be isolated;
2) “An autonomic system must configure and reconfigure itself under varying and unpredictable conditions”. An autonomic system needs to be able to automatically adjust its configuration based on multiple variables and in a timely manner, in order to cope with a dynamic operational environment;

3) “An autonomic system never settles for the status quo—it always looks for ways to optimize its workings”. An autonomic system needs to continuously monitor metrics about itself and its constituent components while it will fine-tune its operation by taking appropriate actions to fulfil system goals and to cope with the complexity and situations presenting conflicting objectives;

4) “An autonomic system must perform something akin to healing—it must be able to recover from routine and extraordinary events that might cause some parts to malfunction”. An autonomic system must be able to identify problems or to foresee potential problems and take actions to keep functioning as required. Root-cause analysis is necessary to identify the causes of failures to provide an indication of what immediate actions need to be taken, based on rules predefined by humans, or discovered by the system itself;

5) “A virtual world is no less dangerous than the physical one, so an autonomic computing system must be an expert in self-protection”. An autonomic system must be able to detect, identify and defend itself against attacks with the aim to maintain its security and integrity;

6) “An autonomic computing system knows its environment and the context surrounding its activity, and acts accordingly”. An autonomic system needs to be able to discover its context and generate rules for how best to respond in a given situation. Thus, a system is able to operate in a wide range of anticipated situations and ultimately it could be able to maintain its operation in unforeseen events;

7) “An autonomic system cannot exist in a hermetic environment (and must adhere to open standards)”. An autonomic system has to be independent for managing itself and operate in a heterogeneous open world, interacting with other self-managed systems based on open standards;

8) “Perhaps most critical for the user, an autonomic computing system will anticipate the optimized resources needed to meet a user’s information needs while keeping its complexity hidden”. An autonomic system hides complexity from the user and it takes actions based on user decisions, without involving her in the enacting process.

Computing and software systems that possess the aforementioned characteristics can qualify as self-managed autonomic systems. P. Horn argued that implementing autonomic systems with such characteristics is very difficult and require significant exploration and advancement of various technologies. He proposed that a direction to realising autonomic systems is to make the individual system components autonomic, while at the same time glue these components together with mechanisms aiming to achieve autonomic behaviour at the system level.

### 2.2.2 MAPE-K Loop

The MAPE-K loop was introduced in 2003 by Kephart and Chess [13] as reference model for developing autonomic systems. It was based on the sense-plan-act control methodology [14] proposed in 1980s by the Artificial Intelligence community for developing autonomous robots. The term MAPE-K refers to the key activities of an autonomic element, i.e. monitoring, analysis, plan, execute, and knowledge.
Towards Engineering Multi-Layer Monitoring and Adaptation of Service-Based Applications

Maintenance and representation. MAPE-K describes the necessary elements for constructing an autonomic component, which is able to perform self-management by collecting information, analysing the collected information to identify a need for change, planning a set of actions to perform the change, and coordinating the execution of the planned actions to address the required change. The MAPE-K loop is an explicit control loop [15], similar to feedback control loops investigated in control theory.

Figure 1 shows the main components of the MAPE-K reference model [13]. An autonomic element refers to a system component having self-managing capabilities. From a structural perspective, an autonomic element comprises the managed element or resource and the autonomic manager. The managed element incorporates sensors necessary for observability and effectors for controllability. The autonomic manager uses the sensors to observe changes in the managed element, while it enacts actions to the managed element using the effectors, such that the managed element is managed by the autonomic manager.

![Figure 1: The MAPE-K reference model.](image)

In order to perform management of the resource, the autonomic manager incorporates a set of functions, as well as, knowledge for various aspects of the managed element. First is the monitor function that collects, aggregates filters and reports the information collected from a managed resource through the sensors. Second is the analyse function that correlates and reasons about the collected information with the aim to infer and identify a need for change. During the analyse function new knowledge is generated about the managed element and it stored in the autonomic manager. Third is the plan function that reasons and decides on the actions needed to perform the change, based on the knowledge concerned with the desired goals. Lastly, the execute function controls the execution of actions on the managed element using the available effectors. As such from the moment that a the monitor will collect some kind of information from a sensor, the MAPE-K control loop initiates, such that the observed information is analysed to detect a need for change, which is
addressed by planning and reason for a set of actions to be enacted using the effectors.

An important aspect in the MAPE-K activities is knowledge produced or consumed during these activities. In the context of MAPE-K, knowledge can concern various aspects of the managed element and the autonomic manager. For instance, the expected behaviour of the managed element or its structure, symptoms identified during analysis, policies configured in the autonomic manager for planning or explicit stating appropriate strategies addressing a required change, past changes applied to the managed element and their effects. Any piece of information that can be leveraged in to the monitor, analysis, plan, and execution functions, or produced by these functions can be assumed as knowledge, which is maintained within the autonomic manager.

The overall aim of the MAPE-K was to introduce a reference model for implementing control loops, which comprise activities that are partially or fully automated, such that a system is capable of exercising self-management. Autonomic elements must have automations for collecting the necessary information from the managed resource, analyse the collected information to identify if a change is required, create a strategy with a sequence of actions, and perform the actions specified by the strategy, in order to address the required changes.

2.2.3 Self-* Properties of Autonomic Systems

The characteristics defined by P. Horn established the concept of autonomic systems. In 2003, Ganek and Corbi [8] elaborated on the four fundamental features, self-configuration, self-healing, self-optimising, self-protecting, required for system to be self-managed and on the four enabling properties, self-awareness, environment-awareness, self-monitoring and self-adjusting, characterising the enabling mechanisms for implementing the fundamental features required in self-managing systems [16].

The four fundamental features or properties of self-managing systems characterise the expected functionality of such systems, which are able to adapt their configuration, to recover from failures, to optimise their operation, and to defend themselves from malicious attacks. The following paragraphs explain each property more analytically.

**Self-Configuring** characterises a system able to adapt its configuration automatically to dynamically changing environments based on higher level objectives, which specify the desired operation, rather than how this operation is accomplished. An autonomic system adapts to the introduction of a new component, such that the autonomic behaviour of the existing components and the overall system is modified appropriately.

**Self-Healing** characterises a system able to detect, diagnose, and recover from failures caused by faulty components. An autonomic system analyses data provided by various information sources such as log files or monitoring components, in order to detect failures. It performs diagnosis to identify the root cause of a failure and when it detects the cause it attempts to correct it, either automatically or assisted by a human.

**Self-Optimising** characterises a system able to continuously improve its operation by tuning parameters and exploiting underutilised resources or releasing ones that are not necessary, in order to meet an operational objective such as performance or cost. An autonomic system tries to leverage every opportunity presented be more efficient with minimum human interleaving.
Self-Protecting characterises a system able to detect malicious attacks and identify potential threats, so that it defends against attacks and it mitigates threats. An autonomic system tries to detect malicious activity, take counter measurements and notify its administrator, in order to sustain its operation.

To realise the four aforementioned features in a self-managing system, the system needs to possess four enabling properties. An autonomic system is aware of its components and its environment, it is able to monitor and analyse information, and it is able to adjust based on these information. The following paragraphs describe each property in more detail.

Self-Awareness characterises a system that has knowledge of the capabilities it can deliver and the components it comprises. An autonomic system is aware of the purpose that it serves and what goals it is expected to achieve.

Environment-Awareness characterises a system able to identify situations that occur during its operation. An autonomic system infers its operational environment or context based on information it receives or observes.

Self-Monitoring characterises a system able to collect and analyse information concerning its operation. An autonomic system uses multiple monitoring components with different monitoring concerns as its main source of information for identifying problems or perceiving its environment.

Self-Adjusting characterises a system able to adjust its behaviour based on the information it knows regarding its operation and its environment. An autonomic system performs adjusting actions in order to adapt in situations occurred in its operational environment or to continue achieving its operational objectives.

Even if the list of self-* properties has become more extensive since 2001 and it now also includes properties such as self-destructing, self-organising and other self-* properties, the two initial sets of fundamental features and enabling properties are enough for characterising an autonomic system [9]. Some self-* properties are challenging to achieve in computing and software systems, while it is desired for a system to possess a combination of properties to realise autonomic behaviour.

2.3 Service-Based Applications

The Service-oriented computing paradigm fosters the development of service-based systems, which utilise the concept of services to encapsulate functionality and to deliver this functionality to heterogeneous systems across networks. Particular instances of a service-based system are service-based applications (SBA). An SBA is developed by integrating multiple heterogeneous services, which are usually developed, controlled and provided by different organisations, not necessarily by the owner of the SBA. The services serve as the building blocks to perform the desired functionalities of the architecture, while they are usually available over a network. An SBA can be seen as a central point or control loop that orchestrates the operation and the interaction with various services, often provided by third-parties. An SBA utilises directly basic services (i.e. services that are not depend on other services) for creating service compositions, which are being leveraged to construct business processes. The service compositions and the business processes of an SBA are usually being executed in a separate run-time platform, than the one used by the basic services, which could
also be provided by third-parties and therefore execute in a platform that is not controlled by the SBA owner.

A service-based application contains three functional (or technology) layers [17], the Business Process Management (BPM) layer, the Service Composition and Coordination (SCC) layer, and the Service Infrastructure (SI) layer. The aforementioned layers do not correspond to a concrete technical architecture, but they merely provide a conceptual view for separating the concerns and functions involved during the engineering and the operation of an SBA.

The Service Infrastructure layer is the bottom level of an SBA and it is concerned with the underlying run-time environment utilised by the SBA. It is worth noting that parts of the run-time environment can be outside the control of the SBA owner. The run-time environment corresponds to the middleware software and the computational resources used by the SBA. The middleware software comprises the software stack (e.g. application container, service bus, service registry, process engine, etc.) required for executing the basic services, the service compositions, the business processes and the SBA as a whole. The computational resources comprise the physical infrastructure (e.g. servers, storage devices, networking equipment, etc.) that provides compute, storage and networking resources for either the execution of the SBA or the execution of the basic services.

The Service Composition and Coordination layer stands between the SI and the BPM layer. The SCC layer is concerned with the organisation and management of the control and the data flow between services, which are composed into business processes to realise a predefined workflow. A developer relies on knowledge about the interfaces and the interaction protocol of the available basic services, which are used to realise the business activities of the process. As such, the characteristics of the selected services have a twofold role, since they affect the way that the composition will be realised, while at the same time they affect the quality of the composition. Therefore, the SCC layer is also concerned with quality of service (QoS) metrics that concerns various quality attributes of the basic services such as response time, cost, availability, reliability and scalability, since quality of the individual basic services constitutes the overall quality of the SBA.

The Business Process Management layer is at the top level of the SBA and it is concerned with the business aspects of the SBA. The business aspects of an SBA
comprise a set of elements such as technology-neutral workflows or business process models to be realised by concrete service composition, process performance metrics for measuring the performance of the realised processes and key performance indicators for overseeing if the measured business performance of processes meets the expected business goals of the SBA. Workflows concern abstract models of the business processes offered in an SBA and they can incorporate business policies and other business-related knowledge. Process performance metrics can be used to measure the various performance aspects (e.g. order shipped per day) of either a specific process instance or multiple process instances by aggregating their measurements. Key performance indicators set the expected value for a metric usually over a period of time and they concern the global business performance of the SBA.

Given the fact that an SBA utilises external services, another perspective to view the abovementioned layers is that each layer concerns a different type of service [18]. The BPM layer concerns business services or business activities realised through software services. For instance, a shipping provider who exposes a Web services API for shipping goods. The shipment of goods is a business activity provided through a Web service. The SCC layer concerns software services that implement a specific functionality or a business activity. For instance, in the case of the shipping provider the Web service is the software service. The SI layer concerns the infrastructure services used by an SBA. For instance, an SBA could be running on a third-party computing infrastructure and rely on shared computing, storage, and networking resources. As such, the SBA made of business services, software-based services and infrastructure services, since it outsources business activities, consumes software services, and uses infrastructure services.

2.4 Adaptable Service-Based Applications

As mentioned throughout the previous section, a service-based application (SBA) usually utilises services provided by third-party service providers. These services are not owned by the owner of the SBA, who does not have control over their evolution and execution. This is a major difference with respect to component-based applications, in which the owner of the component-based application owns and controls its components [19]. As such, the owner of an SBA should incorporate provisions for dealing with the software related failures due to lack of control either on the evolution of the third-party services or quality of services delivered by such services. Also, the fact that part of the business activities realised in an SBA are provided by external services, creates the need for addressing problems related to the business performance of the SBA. An SBA should incorporate mechanisms for both identifying such situations and addressing them by adapting to these situations.

An adaptable service-based application [20] is an SBA enhanced with one or more control loops that aim to systematically monitor the execution and evolution of the application and its operational environment, while they modify the application according to strategies and policies specified by the SBA owner. These control loops can be considered as a realisation of the MAPE-K control loop from Autonomic Computing presented in Section 2.2.2. Such control loops are realised through the combination of monitoring and adaptation processes.

A conceptual framework of monitoring and adaptation of SBA is depicted in Figure 5, which is based on the conceptual framework proposed in [21]. The framework defines the use of monitoring mechanisms to monitor various aspects of the SBA (e.g.
Towards Engineering Multi-Layer Monitoring and Adaptation of Service-Based Applications

functional and non-functional behaviour, business performance, etc.) with the intention to detect problematic situations (e.g. functional and non-functional failures, business performance degrade, etc.). Through analysis of the detected problematic situations, adaptation requirements are being identified (e.g. retry failed actions, correct behaviour, optimise QoS, etc.). In order to address the identified adaptation requirements, it is necessary to generate an adaptation strategy (e.g. substitute faulty services, create a new composition for the problematic workflow, rollback to a previous state, etc.), which defines a sequence of actions to be enacted. The adaptation strategy is executed using the available adaptation mechanisms (i.e. automated service composition, service discovery, etc.), which perform modifications on the SBA.

The modification of the SBA may be necessary to cope with various situations. For instance there are situations which require optimising the technical and business performance. Another example is situations demanding immediate actions to recover from failures. Other situations may require modification of the SBA to adjust to a specific operational context, such that it satisfies the user requirements in a particular manner. Each of the three layers of the SBA is concerned with different situations, which can be individually addressed with different monitoring and adaptation processes.

In the Business Process Management layer, the monitoring process is mostly concerned with either overseeing the business performance of the SBA in terms of KPIs, or checking the conformance to business policies. In case of violations of KPIs or policies occurred, the adaptation process usually aims to reconfigure the workflow of the business process to compensate for the violations. The monitoring and the adaptation processes could also be concerned with the opportunistic business optimisation of the workflows, such that the SBA yields the best possible performance.

As far as the Service Composition and Coordination layer is concerned, the monitoring process aims to reveal problematic situations that refer to software failures, which can be due to violation of functional or non-functional. The adaptation process aims to address such problematic situation by modifying the SBA. The modification can refer to
changes in the control flow or the data flow of a problematic service composition. Additionally, the adaptation process can try to ensure a given quality of service.

Regarding the monitoring process in the Service Infrastructure layer, it is mostly concerned with the detection of problematic situations due to either failures in the underlying infrastructure or the changes in the configuration of the infrastructure used by the SBA. Such failures often refer to either the unavailability of infrastructure resources or faults in the communication infrastructure. The adaptation process in the context of this layer is concerned with the modification of the SBA in order to either recover from failures or adjust to an evolving infrastructure.

Both the monitoring the adaptation processes usually address a specific aspect in one layer of the service-based application. The subsequent sections explain more precisely the monitoring and the adaptation processes, while the last subsection discusses the problem of multi-layer monitoring and adaptation processes.

2.5 Monitoring Process in Service-Based Applications

The process of monitoring can be found in many domains and under varying contexts. In a broad sense, the process of monitoring is associated with the observation of an artefact with the intent to collect data about it. Part of the monitoring process can be the analysis and interpretation of the collected data. In the context of service-oriented computing, the general definition of monitoring is the process of collecting and reporting relevant information about the execution and evolution of service-based systems [22].

Despite the abovementioned definitions, the implementation of a monitoring process has different interpretations depending on the usage of the monitoring process in a service-based application. According to [22], some instances of the monitoring process are the conformance checking of the system’s execution against a given specification, the measurement of non-functional quality of service properties and comparing the measurements to expected values, the measurement and reporting of the business performance and others. Some of these instances are not only concerned with the collection of relevant information, but they incorporate analysis activities as well with the aim to detect problematic situations and to identify adaptation requirements. Henceforth in the rest of the report both the monitoring process and the monitoring mechanisms refer to the collection, reporting and the analysis of relevant information, unless otherwise is mentioned.

The monitoring process targets the collection and the analysis of data for a specific artefact, which is referred as the monitored subject. In the context of an SBA, the monitored subject can be the instance or the class of a business workflow, a service composition, or a basic service. The operational context of either the SBA or its constituent elements can be considered also as the monitored subject. In case that it is necessary to acquire feedback regarding the way the system is monitored and modified, the monitoring and the adaptation mechanisms can be monitoring subjects themselves, in order to improve the operation of such mechanisms based on the provided information. Depending on the usage of the monitoring process, the process can be intrusive to either the monitored subject or its execution platform, such that the observability of the monitored subject is increased.

Various stakeholders can be concerned with the monitoring process. Among them three main roles or perspectives can be explicitly identified. The consumer perspective
Towards Engineering Multi-Layer Monitoring and Adaptation of Service-Based Applications

cconcerns the enactment of the monitoring process from external consumers, who aim to check if the SBA delivers what is expected to them. The provider perspective concerns the enactment of the monitoring process from the provider of the SBA, who wants to oversee if the system satisfies specific requirements. A monitoring process can be also enacted by a third-party, who can provide an independent view of the monitored information.

As monitored information is being referred the outcome of the collection and the analysis of data provided by various information sources during the monitored process. The monitored information is often collected from information sources provided either by the monitored subject, or by its operational context. The information sources comprise execution traces or values of performance metrics corresponding to the functional or non-functional behaviour, events concerning an instance or a class, and contextual information corresponding to changes in the operational context.

There are three main information gathering modes for the collecting monitored information. Polling mode, i.e. to query or directly interact with the monitored subject on regular intervals. Push mode, i.e. propagate events or other data from the information sources of the subject to the monitoring mechanism. Simulation mode, in which a model of the monitored subject evolves based on the monitored information collected using either of push or polling modes. A monitoring process can either use one of these modes or use a combination of them.

Concerning the execution mode of the monitoring process, it can be either synchronous or asynchronous with the execution of the monitored subject. In the synchronous execution, the monitoring process blocks the execution of the monitored subject, until the process completes its activities. Synchronous execution usually takes place when polling mode is used for information gathering due to the direct interaction with the monitored subject. In contrast the asynchronous execution means that the monitoring process runs in parallel with the monitored subject without blocking the execution of the monitored subject. Asynchronous execution takes place when push mode is used for information gathering, because events are being produced as the monitored subject is being executed.

As far as the analysis activity is concerned, there are three types of monitoring processes with respect to the time that a situation is detected. A monitoring process is referred as reactive when it detects a situation as soon as the situation has occurred. In contrast a monitoring process is proactive when it is capable to predict situations before they occur. A monitoring process can also be post-mortem when it detects situations much after they have occurred. The post-mortem monitoring processes usually take place in the form of offline analysis after the execution of the monitored subject. In either case, the analysis activity can be automated, interactive or manual. In automated analysis not human interleaving is necessary, while in contrast in interactive analysis there is substantial human involvement. In manual analysis there is no automation and a human enacts the detection of either situations or adaptation requirements.

The monitoring process is the trigger for the adaptation process in an SBA by providing as the adaptation requirements as inputs to the adaptation process. It is worth noting that in some cases the identification of adaptation requirements i.e. what needs to change to fix a problematic situation can be part of the adaptation process,
since the capabilities of the monitoring process could be limited only to the detection of problematic situations. Ultimately, the adaptation process decides the adaptation strategy, i.e. the sequence of actions needed to address the adaptation requirements, while the adaptation process enacts also adaptation actions that modify the SBA. The next section discusses in more detail the adaptation process in a service-based application.

2.6 Adaptation Process in Service-Based Applications

In service-based application (SBA), a monitoring process can be leveraged to collect monitoring information, detect problematic situations, and to identify adaptation requirements, which will trigger an adaptation process. The adaptation process will then modify the SBA, usually at run-time, in order to address the adaptation requirements. In a broad sense, the process of adaptation is associated with the act of adapting i.e. modifying an artefact, such that it is adjusted to specific situations. In the context of service-oriented computing, adaptation is defined as the process of modifying a service-based system, in order to fulfil new requirements and to fit new situations occurred in its operational environment [22].

According to [22], an adaptation process is required to cope with the uncontrolled evolution of services (often provided by third-parties) utilised in an SBA. Some instances of such problematic situations that require an adaptation process are changes in structures (e.g. a change in the signature of a method), changes in behaviour (e.g. a change in the output of a method), changes in quality (e.g. an increase in the response time of a service), and changes in policies (e.g. a change in a business policy). Additionally, an adaptation process is necessary due to changes in the operational context of an SBA, such as changes in the availability of computational resources, changes in the configuration of the underlying middleware and infrastructure, and changes in the availability of third-party systems utilised by the SBA. The adaptation process comprises activities such as reasoning for deciding adaptation strategies and execution of the adaptation action dictated by the selected strategy, in order to address the abovementioned problematic situation. Henceforth in the rest of the report both the adaptation process and the adaptation mechanisms refer to the decision and the execution of adaptation strategies, unless otherwise is mentioned.

The adaptation process is also concerned with both the decision of strategies and the enactment of adaptation actions for modifying a specific artefact, which is referred as the adaptation subject. In the context of an SBA, the adaptation subject can be the instance or the class of a business workflow, a service composition, or a basic service. The operational context (e.g. the execution platform) of either the SBA or its constituent elements can be considered also as the adaptation subject. However, in the case of such an adaptation subject, the subject has to expose the necessary means in order to be controllable. Additionally, in case that it is necessary to modify the way that the system is monitored and adapted, the monitoring and the adaptation mechanisms can be adaptation subjects themselves, in order to improve the operation of such mechanisms based on a detected situation. Depending on the usage of the adaptation process, the process can be intrusive to either the adaptation subject or its execution platform, such that the controllability of the adaptation subject is increased.

Various adaptation requirements can be identified through the monitoring process. The adaptation requirements describe the desired situation, state, or functionality, to
which the SBA should be brought to [20]. Some instances of adaptation requirements can be to avoid an unavailable service, retry a failed invocation with a specific service, and optimise the quality of service in a business workflow. Such requirements can be identified through the analysis of a problematic situation and they can be addressed by deciding a suitable adaptation strategy. The decision of the adaptation strategy can be either manual requiring a human to make the decision, interactive with minimal human interleaving, or fully automated.

The sequence of actions for resolving an adaptation requirement is referred as an adaptation strategy. Adaptation strategies can be statically decided during design-time, such that they address a specific adaptation requirement. A strategy can be also dynamically decided during run-time based on predefined configuration. According to [19] there are implicit and explicit approaches for specifying an adaptation strategy. In Implicit approaches the adaptation strategy is implied by the adaptation mechanism and the specification cannot change without altering the mechanism. An instance of an implicit approach is when a utility function is used to decide which action will take place. In contrast in explicit approaches the adaptation strategy is explicitly defined through configuration without modification of the adaptation mechanism. Explicit approaches can be further classified into action-based that define which actions to perform in certain situations, goal-based that declaratively define the goals to be fulfilled, and explicit variability that define variation points, which are associated with certain behaviour variants.

Relevant to the time that adaptation requirements are identified by the monitoring process, is the time that an adaptation strategy is decided and executed. The decision can be either reactive i.e. to perform an adaptation action after an adaptation requirement has been identified, or proactive i.e. to perform an adaptation action based on the prediction of an adaptation requirement in order to avoid a future problematic situation. However, the decision can also be refereed as preventive i.e. to perform an immediate adaptation action after detecting a problematic situation, which will eventually lead to an observable failure, which has not been manifested yet. However, for the sake of simplicity throughout this report only the term proactive is used to refer to both preventive and proactive. One could argue that another type of decision is post-mortem, which usually refers to re-engineering of the SBA to address and adaptation requirement identified in the past.

An adaptation strategy can be realised by various mechanisms, which provide the features used in the sequence of actions described in an adaptation strategy. For instance, in order to replace an unavailable service, a mechanism has to be used for discovering an equivalent and bind to the newly discovered service. Other actions that could be used in an adaptation strategy could be to ship a specific invocation, modify the values in an SLA, alter the service selection logic, and others. According to [21], adaptation actions can be classified into service instance adaptation actions (e.g. retry, negotiate SLA, duplicate service, substitute service), flow instance adaptation actions (e.g. substitute flow, redo, choose alternative behaviour, undo, skip / skip to, compensate), service class actions (e.g. change SLA, and suggestion for service re-design), flow class actions (e.g. re-design/re-plan, change service selection logic, change service registry, change platform).

A monitoring and adaptation process usually targets the detection of specific problematic situations, while it is able to identify a limited set of adaptation requirements and support specific adaptation actions (e.g. functionality, quality, business performance) in a specific layer of the SBA. Consequently, this can lead to ineffective adaptations of the SBA that are not adequate to address adaptation requirements, which can be identified due to multiple problems in different layers. As such, both the monitoring and adaptation processes should target multiple aspects of multiple layers, such that the adaptation requirements are addressed effectively and accurately. The next section discusses the challenges associated with the implementation of multi-layer monitoring and adaptation processes in service-based applications.

2.7 Multi-Layer Monitoring and Adaptation Process in Service-Based Applications

Service-based applications (SBA) are highly complex layered applications. A single monitoring and adaptation process often targets a specific concern in the SBA, and thus it is not adequate to address the multifaceted problem of implementing adaptable SBA, since the process does not acknowledge the layered nature of an SBA and the cross-cutting dependencies among its layers.

Multi-layer (or cross-layer) monitoring and adaptation (multi-layer M&A) [17], [23] is concerned with the integration of monitoring and adaptation processes across the functional layers of an SBA. Multi-layer M&A aims to properly diagnose the source of a problematic situation leading to identification of truly needed adaptation requirements, while it aims also to properly generate a holistic adaptation strategy to coordinate the modification of different elements of the SBA, such that it prevents conflicting and ineffective adaptations of the SBA. Multi-layer M&A requires the integration of diverse monitoring and adaptation processes, which are currently realised by isolated monitoring and adaptation approaches that fall in the scope of a specific SBA layer.

The aforementioned perspective has been further elaborated in [17], where the authors argued that the isolated operation of the diverse adaptation and monitoring mechanisms suffers from a number of issues, such as the lack of alignment of monitored events, the lack of adaptation effectiveness, the lack of compatibility of adaptation actions, and the lack of integrity.

The lack of alignment of monitored events refers to the fact that critical information is not propagated across layers and it does not conveyed to all mechanisms, leading to wrong diagnosis or ineffective and possible conflicting adaptations. As such, it is necessary to propagate monitored events across layers, such that the actual source of the problem is identified and uncoordinated adaptation activities are avoided, based on the correlation of the conveyed information across layers.

The lack of adaptation effectiveness means that adaptation mechanisms do not consider desired properties and goals of other layers, leading to ineffective adaptation actions. As such, it is necessary to take into account the desired properties and goals of the whole SBA stack when the adaptation requirements are identified and the adaptation strategy is decided.
The lack of compatibility refers to the isolated adaptation actions performed in each layer, while they are not compatible with the requirements and the constraints posed by other layers. The isolated actions lead to violations of the requirements and the constraints of other layers. As such, it is necessary to consider how the identified adaptation activities impact the SBA as a whole.

The lack of integrity means that the uncoordinated adaptation actions in one layer may not be sufficient to address an adaptation requirement, putting the integrity of the SBA at risk. As such, it is necessary to plan and enact wide range of adaptation actions at different functional layers in a coordinated manner.

In order to implement a multi-layer monitoring and adaptation process for realising an adaptable service-based application, it is necessary to overcome the abovementioned problems. As it has been proposed in [17], there are particular needs to consider for realising multi-layer M&A. Firstly it is necessary to provide ways to express the monitored properties for a specific layer and mechanism in a uniform manner. Secondly, an infrastructure is required to propagate the observed monitored information across monitoring mechanisms used in each layer, such that the observed information is aligned. Thirdly, an integrated adaptation process should aim to the aggregation and the coordination of adaptation actions, while validating different adaptation activities. Complementary mechanisms are required to identify adaptation requirements across layer and to generate adaptation strategies to address these requirements. The aforementioned elements should operate, on the bases of a cross-layer representation of the monitored information, the adaptation strategies, and the SBA model, which captures various aspects of the SBA (e.g. Service models, SLAs and contracts, End-to-end quality models).

Figure 6: Modified conceptual framework of monitoring and adaptation process to support multi-layered monitoring and adaptation of service-based applications.

In order to illustrate the aforementioned requirements, Figure 6 shows a modified version of the conceptual framework of monitoring and adaptation process (originally presented in Section 2.4). The modified version of the framework positions the issues
that need to be addressed, in order to afford multi-layer monitoring and adaptation of service-based applications. Monitoring needs to take place across layers following an integrated and aligned manner, such that it is able to detect problematic situations that concern multiple layers. The problematic situations should be diagnosed sufficiently and considering the cross-layer dependencies, such that the real source of the problem is diagnosed and the appropriate adaptation requirement is identified. The decision of the adaptation strategy needs also to consider the cross-layer dependencies and examine the impact of the strategy to the SBA. Finally, the execution of the modifications needs to be integrated and coordinated across layers.

As it will be discussed in the literature review, the current state of the art on multi-layer monitoring and adaptation of service based applications is rather limited. The recent research efforts towards addressing the multi-layer monitoring and adaptation of SBA are rather scattered.
3 State of the Art on Monitoring and Adaptation of Service-Based Applications

Over the past decade there has been an increasing interest in engineering self-adaptation capabilities in service-based systems, motivated by the high level of complexity involved in such systems and the dynamic, often heterogeneous, operational environment of service-based systems, and also the fact that they comprise a variety of types of services (e.g. business, software, infrastructure) usually controlled by different stakeholders. The result of this complexity is that the requirements satisfied during design time can be trivially violated at run-time, leading to failures or degradation of the service-based system.

As such, the research community of service-oriented computing has recognised the need to abstract this complexity by introducing automations into the everyday operation of service-based systems. Such automations include the monitoring and the analysis of operation of a service-based system with the aim to detect new requirements or adaptation needs, as well as, the planning and the execution of the actions necessary to adapt the system, such that the newly detected requirements are addressed in parallel with the execution of the service-based system. The research community has widely acknowledged that the implementation of such mechanisms in service-based systems is a significant and challenging problem. The two relevant research fields, Autonomic Computing [11] and Self-Adaptive Systems [24], investigate the challenges associated with the self-management and the self-adaptation in the greater context of computing and software systems, and therefore serve as the foundations towards addressing the challenges associated with the self-adaptation in service-based systems.

Research in adaptive service-based systems is outlined in [25, 26]. The main body of research work has focused on monitoring approaches for the collection and the analysis of information to detect adaptation needs at run-time, as well as, adaptation approaches for the reasoning, or the planning, and the execution of modification actions to address the newly detected adaptation needs. In most approaches, the monitored information is associated with the operational aspect, i.e. the business activities realised by a service-based system, or the technical aspect, i.e. the functional and non-functional requirements. The modification actions are associated with the structural modification of business processes, replacement of individual services, changes in service-level agreements, and others.

The majority of the various monitoring and adaptation approaches usually focus on partially addressing a very specific problem from a very particular perspective. The current landscape of monitoring and adaptation approaches for service-based systems is very fragmented. As such and given the layered nature of a service-based system, it is necessary to develop complete solutions for self-adaptation that work together in harmonisation. Such solutions should leverage information and capabilities made available by diverse monitoring and adaptation approaches. Integrated and coordinated solutions should overcome the problems of producing and consuming conflicting information, or executing conflicting actions that will ultimately lead to a not effective solution. This observation is supported in [25, 26] and it is confirmed by the literature presented in this section.
The motivation for conducting the literature review was twofold. Firstly and most importantly to illustrate the necessity for addressing the problem of the integration and coordination of the existing approaches within a concrete framework, and secondly to distinguish what has been done from what needs to be done.

The goal of this literature review was to map the landscape of research efforts concerned with the monitoring and the adaptation of service-based applications in either the Business Process Management (BPM) or the Service Composition and Coordination (SCC). Approaches concerned with the Service Infrastructure layer were not included, because they mostly concern monitoring and adaptation of computation resources (e.g. memory capacity, cpu load, network capacity, etc.), which are beyond the scope of this project. Additionally, this literature review has considered recent research efforts concerned with multi-layer monitoring and adaptation (multi-layer M&A) of service-based applications.

The literature review was conducted in a systematic way. The collection of publication was performed throughout the past period using multiple online resources, such as Scopus, IEEE Xplore, ACM Digital Library, SpringerLink, and Google Scholar. Also, important dissemination channels, such as journals and conferences, were closely monitored for new publications. A classification framework was established, in order to organise the publications and to gain a clear understanding of the existing landscape of research efforts. The classification framework was based on the taxonomy of monitoring and adaptation principles and mechanisms for service-based applications introduced in [21]. Although other taxonomies were also considered, such a taxonomy of software-fault monitoring [24] and a taxonomy of self-adaptation [26], the taxonomy from [21] was selected, mainly because it was more relevant and complete. The motivation for classifying the research efforts in the classification framework was to examine how the existing fragmented approaches for monitoring and adaptation are positioned with respect to the methodology, architecture, and the activities realised by each approach. The complete classification of the approaches is included in Appendix I: Classification of Research Efforts on Monitoring and Adaptation.

The subsequent sections follow a thematic organisation for reporting the literature review. Firstly, it discusses the research efforts concerned with the monitoring and the adaptation in the BPM layer. Secondly, it discusses research efforts addressing monitoring and adaptation in the SCC layer. Thirdly, the literature review discusses the current state of the art in multi-layer monitoring and adaptation of service-based applications. The first and the second thematic sections begin with an overview of the concerns addressed by the monitoring and adaptation research efforts in the specific layer. Next, they provide a summary of the existing research efforts and continue by discussing notable research efforts. The two thematic areas conclude with a brief discussion of the research efforts classification. The third thematic area has been organised differently, since the relevant research efforts are rather scattered. However, an effort has been made to organise these research efforts in ones addressing diverse aspects of multi-layer M&A, others presenting multi-layer M&A solutions based on ad-hoc integration of mechanisms, and the small number of efforts focusing on the architectural aspects of multi-layer M&A.
3.1 Monitoring and Adaptation in the Business Process Management Layer

3.1.1 Research Efforts on Monitoring Approaches
The monitoring approaches in the Business Process Management (BPM) layer of a service-based application (SBA) are concerned with the collection and the analysis of information related to business aspects of an SBA. The business aspects may comprise the business processes delivered through the SBA, the performance of these processes in terms of key performance indicators, and their compliance to the various policies defined by the organisation that operates the SBA or external authorities imposing the policies (e.g. data protection). The monitoring of such concerns is the primary subject of study in the research fields of Business Activity Monitoring (BAM) and Process Analytics. In the next paragraphs, we focus only on approaches for monitoring business concerns that have been applied in the context of service-oriented computing. As such, research works focusing on BAM or Process Analytics fall outside of the scope of this survey.

The main body of work focusing on the monitoring in the BPM layer is concerned with the detection of interesting interaction patterns occurred during the execution of the process [27], the monitoring of the business performance of a processes executed within one organisation [28] or spanning across multiple organisations [29], as well as the analysis of the observed performance to identify the influential factors causing low performance [30] and the detection of service-level agreements violations [31]. Additionally, few works are concerned with monitoring the compliance of the execution of a process to detect violations of policies [32], [33]. The aforementioned concerns focus on the business realised in a process and not on the technical characteristics of the process (e.g. functional or non-functional requirements) that fall in the scope of the Service Composition and Coordination layer. The following paragraphs present in more detail few of the key research works addressing the aforementioned concerns.

Lazovik et al. [32] described an approach for monitoring the compliance of BPEL processes to business rules during run-time. The business rules are defined using different types of assertions provided by the XML Service Assertion Language (XSAL). The types of assertions include simple reachability conditions, conditions that need to be satisfied through the execution and conditions that check the evolution of a particular business property. The monitoring approach has been realised in the context of a middleware for service-based systems that is able to plan and monitor the execution of processes according the predefined assertions.

Beeri et al. [27] introduced a framework for analysing event logs produced during the execution of a BPEL process with the intention to identify interesting patterns related to business policies and statistics. For example, the framework can be used to identify process instances where a user has tried to perform an activity, without completing a pre-required activity, or to collect statistics for a specific process class (e.g. find the age of users). The monitoring framework comprises a visual editor that allows a user to express visually queries. Directed Acyclic Graphs (DAG) are used to model the execution traces collected during the execution of the process. Finally, DAGs are used for pattern matching, such that they consume the execution traces to identify patterns.
Wetzstein et al. [28] described an approach to monitor the business performance of BPEL processes by measuring various performance metrics. For example, using this approach one can measure the duration of a process or an activity within the process, find the time that an activity occurred, and count the number of executions of the same activity within a loop of the process. Even if the aforementioned measurements can be done for a process instance, the approach supports measurements for a process class by aggregating measurements from the individual instances. The approach uses XML for representing the process performance metrics in a platform-independent manner. An event-based monitor can be automatically generated based on the specified process performance metrics for a specific BPEL process engine. In a follow-up work [30], the authors presented an approach for discovering the main factors influencing the performance of business process. The approach applies machine learning techniques on the historical metrics, collected during monitoring, to create a dependency tree, which represents the process performance metrics that have influenced a key performance indicator. The dependency tree can be leveraged by a developer acting as the process designer to manually modify the process, therefore avoiding future violations by removing the indicated parts of the process that are led to a violation.

Leitner et al. [31] presented an approach for predicting SLA violations in BPEL processes at run-time. During the design-time of the BPEL process, checkpoint activities, invoking the prediction mechanism, are inserted in the BPEL process to be monitored. The prediction is based on machine learning regression techniques, and trained using historical process instances. The prediction approach requires QoS information that has been already collected, estimated QoS values for QoS attributes that are not known, and process instance data (i.e., data specific to the process, for example, a customer or an order). The aforementioned information converted into the format expected by the prediction model (WEKA Attribute-Relation File Format ARFF2). Finally, the prediction is performed by passing the gathered input to the prediction model, which produces a numerical estimation of the SLO value.

Wetzstein et al. [29] presented a framework for monitoring of service choreographies comprised of BPEL processes, which span across different organisations, in order to measure the business performance of the processes. The framework suggests the use of a monitoring agreement between the organisations. The monitoring agreement is expressed in XML and it specifies the events that each party will have to provide for enabling the cross-organisational monitoring of a business process. The agreement supports two types of events: Resource events, i.e., concern process instances, activities, variables etc., and complex events i.e. events derived from the correlation of basic events. The correlation of basic events to infer complex events is performed using a complex event processing engine operated by a third-party organisation, which oversees the execution of the service choreography.

Dustdar et al. [33] introduced a framework for monitoring of BPEL processes to check whether they comply with various policies. The framework uses a complex event processing engine to infer complex business events based on basic events emitted by the process engine during the execution of the monitored process. The basic events contain unique identifiers, which are used to map the events to a model of the monitored process. The inferred business events are fed to a component that checks the compliance of the process against the compliance meta-model, while it also
probes the running process instance using the API provided by the underlying process engine for retrieving more information about the monitored process.

The classification of the abovementioned approaches together with the classification of other monitoring approaches for the BPM layer is presented in Appendix I: Classification of Research Efforts on Monitoring and Adaptation. The classification presents how the various monitoring approaches for the BPM layer are positioned with respect to the methodology used for data collection, the architecture adopted in the approach and the characteristics of the analysis activity suggested by the approach. The following paragraphs discuss the classification of the monitoring approaches for the BPM layer.

The methodology of the approaches is characterised by the mode used for collecting information and the effect that the execution of the monitoring approach has on the monitored subject. The analysis reveals that most approaches use push mode for collecting the monitored information, while they do not block the execution of the process and therefore they do not introduce a significant overhead in the execution of the monitored subject.

The distribution of components in the architecture adopted in the most approaches is centralised, because some sort of integration is required with the execution platform, which has to have mechanisms for pushing events to the monitoring component, such that the monitoring subject (e.g. a process) is observable. Additionally, most approaches address the monitoring activity from the perspective of the provider, who is the main stakeholder interested in the performance and compliance of the processes delivered.

The analysis performed in most approaches uses the events collected by the monitoring mechanism to identify execution patterns, perform measurements, or non-compliance of the monitored process. Most approaches are able to identify such concerns reactively, i.e. relatively close to the time of their occurrence, while their operation does not require human interleaving.

Monitoring approaches are particularly useful for triggering an adaptation process. The next section discusses the existing research efforts on adaptation approaches for the Business Process Management layer.

### 3.1.2 Research Efforts on Adaptation Approaches

The adaptation approaches in the Business Process Management (BPM) layer of a service-based application are concerned with the modification of running business processes, usually realised using BPEL, in order either to address operational problems or to accommodate changes in the organisational environment. The former concerns problems caused by deviations from the expected performance at the business level (e.g. failure to fulfil a particular number of orders within a period of time) or non-compliance to business policies (e.g. the maximum number of items in an order should not exceed 50), while the latter concerns the availability of better business services (e.g. a better shipping service is available), updates in business policies and legal regulations, or modified workflow models that improve the realisation of business processes from an operational perspective (e.g. new activities should be included in a process). Such changes can be accommodated by taking the running processes offline, modifying them, and redeploying them. However, this approach to re-engineering the process does not allow for immediate response to changes in the running instances,
since it requires waiting for the completion of all running instances without addressing the new requirements. As such, adaptation is necessary to accommodate the new changes by modifying the running instances temporarily or permanently changing the process.

The main body of work focusing on adaptation approaches for the BPM layer is concerned with the structural modification of a business process, in order to accommodate changes occurred in the organisation environment (e.g. [34–36]) and to cope with deviations from the expected quality of service (e.g. [37], [38]). There are also few approaches that concentrate on verifying the correctness of modifications (e.g. [39], [40]). These works approach the adaptation of a BPEL process from the business perspective and focus on how to address issues related to the business realised in such processes. The following paragraphs present in more detail few of the key research works addressing the aforementioned concerns.

Reichert and Dadam [34] described an approach to supporting the structural changes of running workflow instances by enacting manual modifications. The approach uses a formal workflow model, based on Petri-nets, to define a set of modification actions, such as task deletion, serialisation, changes in the order of the operations, and others that can be used to modify a running workflow instance. The framework is able to determine whether a specific change can be applied to a particular workflow instance or not, by checking verifying the correctness and consistency of the modifications. In case that a modification violates the correctness it is either rejected or it can be applied by handling the exceptions resulting from the change, in order to restore the correctness. The proposed approach is aimed to facilitate change management of running workflow instances by providing guidance to process developers concerning the applicability of the available modification actions to the running workflow instance. In a latter work, the authors present the implementation of a workflow management system [41], which incorporates the aforementioned approach.

Hallerbach et al. [35] presented an approach for managing and deriving process variants to accommodate changes (e.g. changes of legal regulations), which affect multiple process variants. The approach has been based on explicit variability techniques used to define the variation points and the process variants at design-time, while the use context-rules to select the appropriate process variant at run-time. The authors use one process model that contains all process variants. A specific variant is then derived from the unified process model by applying a set of well-defined change actions to the process model. The actions can be are insert, delete or move process fragment, and modify process attributes. The process variants are associated with the process context in which a variant will be used is based on context variables and rules, which are explicitly defined at design-time. During run-time a process variant is selected based on the contextual information.

Leitner et al. [37] proposed an approach for preventing violations of service-level agreements concerned with the business performance of BPEL process. The approach combines aspect-oriented programming with process fragment in order to substitute fragments in a running process instance. The substitution of a process fragment can be proactively triggered by predicting SLA violations using the approach described in [31]. The best suited fragment is selected to prevent the predict SLA violation, by evaluating the available fragments with respect to their expected impact on the performance of the BPEL process. However, the authors do not consider that an adaptation could lead to another violation from the one that it has prevented. The authors have
implemented a prototype using the Windows Workflow Foundation technology provided in the .NET framework.

The classification of the abovementioned approaches together with the classification of other adaptation approaches for the BPM layer is presented in Appendix I: Classification of Research Efforts on Monitoring and Adaptation. The classification presents how the various adaptation approaches for the BPM layer are positioned with respect to the methodology used for achieving adaptation, the architecture adopted in the approach and the characteristics of the decision implemented in the adaptation approach. The following paragraphs discuss the classification of the adaptation approaches for the BPM layer.

The methodology used in adaptation approaches is characterised by the type of strategy that is used for the adaptation and the type of the modification actions that are supported in the approach. The strategy used in the adaptation approaches is based upon explicit variability techniques, goal-based and action-based techniques. In the majority of the approaches, the adaptation strategy is specified at design-time and it supports modifications of the process instance’s flow. An exception is [36], which addresses modifications applied to the process class and affects all instances of a particular process.

The architecture of the adaptation approaches is characterised by the distribution of the components used in the approach and the invasiveness of the approach. In all approaches the components are centralised, because they mainly target the modification of a process instance that is being executed within a process engine. Additionally, few approaches (e.g. [34], [39], [40]) are independent of the process being modified or the platform used for the execution of the process, while most of the approaches require integration with the execution platform (i.e. the process engine).

Concerning the decision in adaptation approaches, the approaches that require a human to decide the modifications manually are dynamic with respect to the decision dynamicity, since the decision of applying an adaptation action is left to the human to decide (e.g. [34], [36], [39], [40]). In contrast, automated approaches are mostly static. However, there are automated approaches that are dynamic (e.g. [37], [38]), because they consider the impact of the applicable adaptation actions to a given situation and therefore the adaptation strategy may vary depending on a particular situation. Finally, most approaches are reactive with the exception of [37].

Thus far the literature review focused on the existing research efforts on monitoring and adaptation approaches for the Business Process Management layer. Such approaches focus more on enacting monitoring and modifications of service-based application, in order to preserve the business level goals of the application. The subsequent sections discuss the research efforts on monitoring and adaptation approaches for the Service Composition and Coordination layer.

### 3.2 Monitoring and Adaptation in the Service Composition and Coordination Layer

#### 3.2.1 Research Efforts on Monitoring Approaches

The monitoring approaches in the Service Composition and Coordination (SCC) layer of a service-based application (SBA) are concerned with the collection and the analysis of
Towards Engineering Multi-Layer Monitoring and Adaptation of Service-Based Applications

information related to the functional and non-functional aspects of an SBA. The functional aspects may comprise the functional correctness of the processes composing software-based services or the correctness of the individual services participating in a process, while the non-functional aspects may comprise the quality attributes (from a software engineering perspective) of the processes or the individual services. The main difference between the BPM and the SCC layers is that the SCC sees the SBA from the perspective of software engineering, while the BPM layer views the SBA from the perspective of the business realised and delivered through the SBA. The next paragraphs focus the monitoring approaches for the functional and non-functional concerns that have been proposed in the context of service-oriented computing.

The main body of work on monitoring approaches for the SCC layer can be classified into three groups. The first group concerns BPEL processes and service compositions focusing on the run-time correctness of process instances to detect violations of requirements [42–44], the measurements of the quality of service in processes to detect problems related to quality degradation [45], [46], and the detection of failures occurred in processes [47–50]. The second group concerns the individual Web services participating in a service composition, for example a BPEL process, and the interaction between these services within the context of the composition. This group focuses on the behavioural correctness of the individual Web services and their interactions according to a set of pre-specified requirements, which are in the form of assertions that check if a desired property is satisfied [47], or they are model-based using a state-based model to specify the control flow of the interactions [51–53]. The third and last group concerns the interaction with individual Web services focusing on the behavioural correctness, i.e. the control flow and data flow of the interaction with the services [51], [54–58] and the conformance of services to service-level agreements [59–64], that describe the expected quality characteristics from the monitored services. The aforementioned groups of approaches address the concerns related to the technical perspective of the SBA. The following paragraphs present in more detail few of the key research works addressing the aforementioned concerns.

Baresi et al. [47] proposed an approach to monitor the correctness of Web services that participating in a BPEL process with respect to predefined assertions. The proposed approach requires a developer to annotate a BPEL process with comments that can contain assertions for three classes of undesirable behaviours, i.e. timeouts, run-time errors, and violations of functionality. The annotations are automatically translated into BPEL activities and included within the monitored BPEL process. The newly included activities invoke external Web services that act as dedicated monitors, in order to evaluate the assertions. The authors present two alternative implementations for evaluating the assertions forwarded to the dedicated monitors. The first approach uses the C# programming language in the .NET environment, while the second one uses xLinkIt, an assertion validation engine. In a follow-up work [48], the authors incorporate the aforementioned approach in a monitoring framework for BPEL processes. The framework allows specifying monitoring rules, which can be dynamically associated with the monitored BPEL process instance. Also, the assertions are specified in the Web Service Constraint Language. The novelty of this work is that the assertions are decoupled from the monitored process, which still has to invoke the monitoring framework. However, the selection and execution of the various monitors is decided at run-time based on the pre-defined monitoring rules.
LI et al. [55] described a framework for monitoring and validating the interaction behaviour of SOAP-based Web services based on predefined interaction constraints. The interaction constraints are represented using Finite State Automata (FSA), based on the Specification Pattern System [65] that describes patterns for specifying the restrictions on both the occurrence and sequence of invocations of operations. The framework is able to synchronously intercept SOAP messages exchanged between Web services and/or their consumers, extract the key elements from the message of the body associated with the operation invocation, and generate events forwarded to the constraints validation component. The validation component performs the validation asynchronously and generates a validation report. The authors have implemented the framework using the Tomcat application server for deploying the monitoring framework, and the Apache Axis Web services framework for deploying the monitored Web services.

Ameller and Franch [60] proposed a generic architecture, namely SALMon, for monitoring Web services to detect violations of service-level agreements and to decide a treatment as a response to SLA violations. SALMon architecture follows a SOA style, since most of its elements are realised as services, which are loosely coupled and can be replaced by other services with variable functionality. The monitors store the measures in a database, which is accessed by the analyser component in order to process the measures and to detect SLA violations. In the event that a violation is detected, the analyser components notify the Decision Maker component, which will have to perform corrective actions to satisfy SLA objectives again. Additionally, the authors introduce a quality model for software-based services that describes various quality criteria (e.g. availability, accessibility, etc.) and metrics for measuring the quality criteria. Such measurements are performed in SALMon by measure instruments, which are being coordinated by various monitors.

Aalst et al. [52] described an approach to post-mortem analysis of message logs produced during the interactions of SOAP-based Web services within a service choreography expressed in BPEL, in order to identify unexpected behaviours based on a choreography process model. The approach considers two notions of conformance, fitness, i.e. whether the observed process behaviour complies with the control flow specified in the process model, and appropriateness, i.e. whether the model describes the observed process in a suitable way. The process model, expressed in BPEL, is transformed automatically to a Petri-nets formal process model. During the execution of the process, the SOAP messages exchanged are recorded to a message log. Later, a conformance checking component named ProM, analyses the message log and produces an event log, which describes the event sequences (e.g. sequences of messages exchanged). Finally, ProM uses the Petri-net process model to “parse” every event sequence extracted from the message logs. If the model is unable to “parse” an event sequence, the implemented choreography does not conform to the model.

Baorsi et al. [46, 47] presented an integrated framework for detecting behavioural patterns in BPEL processes, measuring numeric and time-related properties, and monitor the behaviour of a single process instance or all instances of a process by aggregating the monitored information. The integrated framework comprises two previous monitoring approaches for BPEL processes, namely Dynamo [48] and Astro [68]. The authors present the integration for both the language used to express the properties to be monitored, as well as for the overall architecture of the monitoring framework. The architectural integration is realised by using a simple event bus that
collects the relevant events on the process of interest and forwards them to the correct analysers. The authors argue that the main challenge associated with the language integration is to avoid conceptual redundancy, and to provide a solution that is as cohesive as possible for creating monitoring specifications. Concerning the architectural integration, the authors state that the main challenge is to maximise code reuse, while at the same time accommodate a solution that is efficient.

Michlmayr et al. [69] presented a framework for event-based QoS monitoring of Web services that is able to detect SLA violations occurring during the operation of the services. The framework operates as a third-party entity in a service-oriented environment and it is part of the Vienna Runtime Environment for Service-oriented Computing (VRESCo) [70], a middleware for service-oriented systems. The framework supports both monitoring at the server-side and the client-side. At client-side monitoring is done by sending probe request to the monitored services, while measuring various QoS attributes (e.g. response time). At the server-side monitoring is done by using vendor-specific techniques, in this case Windows Performance Counters provided by the Windows Communication Foundation of the Microsoft .NET framework. The observed information from both the client-side and the server-side (providing the Web services) are sent in the form of events to a host running the monitoring framework, which uses an event engine and a QoS manager that both are used to detect SLA violations. The actual detection of violations is done using complex event processing techniques.

Lorenzoli and Spanoudakis [64, 65] described a framework for detecting potential violations of quality of service (QoS) properties by predicting the values that such properties will have in a future point in time. The framework has been developed as a run-time environment for executing QoS predictors and storing their predictions. The framework provides the mechanisms for collecting monitored events, which are serve as the basis for predicting future QoS attributes. The actual prediction is performed by using Event Calculus specifications to identify relevant monitored information, which are used to infer QoS property prediction models. The prediction models are then used to detect potential violations of QoS properties by estimating the probability of such violations.

The classification of the abovementioned approaches together with the classification of other monitoring approaches for the SCC layer is presented in Appendix I: Classification of Research Efforts on Monitoring and Adaptation. The classification presents how the various monitoring approaches for the SCC layer are positioned with respect to the methodology used for data collection, the architecture adopted in the approach and the characteristics of the analysis activity suggested by the approach. The following paragraphs discuss the classification of the monitoring approaches for the SCC layer.

Concerning the methodology, the most approaches follow the push mode for information gathering, which is realised mostly by collecting the events generated during the execution of BPEL process or by intercepting the messages exchanged during the interaction with the individual Web services. Even if the majority of the approaches do not block the execution of the monitored process or service, almost half of the approaches block the execution of the monitored subject. Few of these approaches block the execution in order to have control over the execution of the monitored subject, for instance, to recover as soon as possible from a failure (e.g. [73]).
Towards Engineering Multi-Layer Monitoring and Adaptation of Service-Based Applications

The majority of the approaches use a centralised architecture by integrating the monitoring and the analysis components with the execution platform used by the monitored BPEL process and Web service. It is noteworthy that there are many approaches that even if they investigate the monitoring and the analysis activities from the provider perspective, they actually focus on the individual Web services participating in a BPEL process that can be provided by a different entity (e.g. [47], [52], [53]). Therefore, it can be argued that such approaches concern also the consumer perspective, since the services addressed by such approaches are consumed in a BPEL process. There are also few approaches that use a decentralised architecture with the aim to have a global view of the failures occurred (e.g. [49], [54], [59]), while allowing the various components (for monitoring and analysis) to be hosted at different physical locations (e.g. [60]). Such approaches usually investigate the monitoring and the analysis activities from the perspective of a third-party entity, which is responsible for detecting violations.

Regarding the analysis of the monitored information, the majority of the approaches use events generated by the execution platform or messages intercepted during the interaction with the monitored services. There are also few approaches that parse execution logs to detect issues occurred during run-time (e.g. [62], [52]). Although most of the existing analysis approaches are able to detect problems after they have occurred, being able to predict potential problems has been acknowledged as a promising direction with recent research works demonstrating the first research results in predicting QoS attributes (e.g. [64], [71], [72], [74]) or detecting potential problems related to the functionality of a service (e.g. [44], [58]).

Monitoring approaches presented thus far provide the half solution for engineering adaptation in the Service Composition and Coordination layer of a service-based application. The next section discusses the existing research efforts on adaptation approaches for the same layer.

3.2.2 Research Efforts on Adaptation Approaches

The adaptation approaches in the Service Composition and Coordination (SCC) layer of a service-based application (SBA) are concerned with the decision and enactment of appropriate adaptation actions in an SBA. The adaptation actions aim to address adaptation requirements concerned with software engineering perspective. The adaptation actions are targeted to alter either the functional or non-functional behaviour of an SBA. Adaptation actions usually involve the modification of the flow (control and data) and the modification of service interfaces. More particularly, adaptation actions usually comprise the replacement of the constituent software-based services in a business process based on various criteria, the modification of the predefined way of interaction to use new services, or the reconfiguration of the SBA to adapt to new requirements. The next paragraphs focus on the adaptation approaches that suggest solutions for addressing the aforementioned concerns and they have been proposed in the context of service-oriented computing.

The main body of research efforts on adaptation approaches for the SCC layer can be clustered into three groups based on the generic goal that each group of research efforts attempts to address. The first group targets the optimisation of service compositions, particularly BPEL processes. The second group focuses on the recovery from failures occurred either in the execution of a BPEL process or during the interaction with the constituent services. The third group addresses mismatches in the
interface or the interaction protocol between Web services, which are being substituted. The adaptation approaches among the three groups often use custom mechanisms (which in some approaches involve the use of formalism) for implementing the reasoning and the execution of the adaptation actions. It is worth to note that a frequently used mechanism in adaptation approaches is Aspect-Oriented Programming (AOP), which provides the facilities to either interleave with the execution or implement explicit variability techniques. The following paragraphs present in more detail each group of approaches, together with few of the notable research efforts in each group.

The central focus of the first group of research efforts (e.g. [45], [75–81]) is the dynamic selection and binding of the best services to fulfil a particular function within a service composition, which is usually realised as a BPEL process. Adaptation approaches, such as the ones proposed in [75], [78], [79], use an abstract description of the workflow that does not specify concrete services, but it defines either the quality of service requirements or the desired functionality that has to be provided by concrete services at run-time. During the deployment or the execution of the abstract workflow, service discovery and selection mechanisms are employed to choose the most appropriate service to fulfil the desired functionality. The service is usually selected from a registry that maintains more comprehensive description of the functional and non-functional characteristics of the available services, in order to facilitate the discovery and selection of a service. A common issue due to the dynamic binding of services in these approaches is the service interface and the interaction protocol mismatches, which occur between the expected interface and protocol defined in the abstract workflow and the concrete services selected at run-time. Although this issue is better addressed by the third group of adaptation approaches, there are also approaches in the first group, such as [45], [79], [80], [82], which consider the mismatch issues. The following paragraphs present in more detail few notable research efforts that address the aforementioned concerns.

Zeng et al. [75] proposed an approach to optimally select services during the execution of a service composition, based on constraints set by the user on various quality characteristics, such as cost, duration, reliability, availability, and reputation. The main goal is to select the optimal execution path that uses the appropriate services. The authors argued that a service composition can be described using statecharts, such that each state of the statechart represents the invocation of an external service. Each execution path of the statechart, involving different services, can be represented as a Direct Acyclic Graph (DAG) under the assumption that the statechart does not contain cycles. The authors described a global planning approach, which based on the available services, assigns a service for each task in every execution plan, in order to generate the set of execution plans. The selection of the optimal execution path is performed based on a Multiple Attribute Decision Making (MADM) approach.

Chafle et al. [78] presented an approach for adapting service compositions during deployment and run-time, based on a set of functional and non-functional requirements. During deployment multiple workflows are being generated that realise the service composition. Each workflow is an execution plan that comprises different services, which have varying QoS. An optimal plan is selected for executing the composition. In the event of a failure, alternative workflow candidates are explored to replace the faulty one. Semantic annotations are leveraged for describing the characteristics of the candidate services.
Ardagna and Pernici [79], [80] presented a framework supporting the execution, optimisation, and semiautomatic adaptation of BPEL processes. The framework allows for the selection of concrete services at run-time based on quality of service constraints. The underlying mechanism for the service selection uses mixed integer linear programming. The services are selected from a semantic registry, which contains descriptions of the quality characteristics of services. The service invocations defined in the process are performed by a mediation engine, which redirects the invocations in the process to the selected services. Additionally, the framework is capable to perform recovery actions in the event of a failed service invocation. Recovery actions can be to retry a service invocation or to select a new service, which will substitute a faulty one.

The research efforts of the first group mentioned thus far concentrate mostly on the optimisation of BPEL processes. The central focus of the second group of research efforts, such as [73], [81], [83–97], is failure recovery either during the execution of a service composition, often caused either due to faults presented in the constituted services, or changes in the operational environment. These approaches are implementing recovery actions, which often aim either at replacing a problematic service, or retrying a failed invocation. The following paragraphs present in more detail few notable research efforts that address the aforementioned concerns.

Charfi and Mezini [83], [89], [92] introduced an extension to the standard BPEL language in order to enable the support aspect-oriented programming. The support of aspect weaving in BPEL was a major advancement, which added support for evaluating conditions before or after the invocation of an external service. Each aspect can incorporate BPEL code, which upon its activation it enact various tasks, such as counting activities, modifying the process by replacing a service and other actions. The great benefit of this extension is that aspects can be hot-deployed to a running process instance, without requiring to take the instance offline.

Baresi et al. [73], [88], [98], [99] extended their previous work [47], related to monitoring of service compositions, to a complete monitoring and adaptation framework for BPEL processes. The authors introduced the Web Service Recovery Language that can be used to create adaptation strategies specifying recovery actions. The adaptation strategies are defined explicitly during the deployment of the monitored process. There are three sets of actions, service actions, such as retry a method invocation, process actions, such as substitute a service, and administration actions, such as send a notification. The recovery actions execute synchronously with the execution of the monitored process and therefore the monitored process is blocked until the recovery action completes. The framework makes use of aspect-oriented techniques to weave monitoring and adaptation functionality during the execution of the BPEL process.

Modafferi et al. [87] introduced a self-healing framework for BPEL processes. The framework has implemented as a plug-in, which can be integrated with a BPEL engine, in order to enhance the ability of a standard engine to provide process-based recovery actions. The framework introduces the necessary constructs to annotate a BPEL process with recovery actions. The annotated process is then transformed by the plug-in, such that it inserts the necessary BPEL activities that realise the recovery actions. As such, at design-time a process designer is able to create more complex recovery strategies, based on a combination of recovery actions.
The research efforts of the second group mentioned thus far concentrate mostly on the failure recovery in service compositions. The central focus of the third and last group of research efforts is to address the mismatches between the expected interfaces and interaction protocols that usually occur either when dynamic binding of services takes place at run-time, or during the manual adaptation of running services. Interaction protocol mismatches refer to the case that a semantically equivalent service is used to implement a required activity, but the sequence of invocations needed to complete a given task is different from the sequence of actions supported by the service consumer. Similarly, interface mismatches refer to the case that there are syntactic and semantic differences on the interface exposed by a service and the interface expected by the service consumer. Research efforts such as [100–105] address interaction protocol mismatches, while efforts such as [104], [106–109] target both interface and interaction protocol mismatches. The following paragraphs present in more detail few notable research efforts that address the aforementioned concerns.

Benatallah et al. [106] proposed a framework for developing Web service adapters through a semi-automated process, in order to address interface and interaction protocol mismatches. The authors classified into the proposed framework the ways that the interface or the interaction protocol may differ. They captured these differences as mismatch patterns. For each pattern, the framework provides code templates that can be used for semi-automated generation of an adapter.

Brogi and Popescu [107] presented an approach for automatically generating adapters in order to address interaction protocol mismatches between interacting BPEL processes. The authors show method for transforming the interacting BPEL processes into the Yet Another Workflow Language (YAWL). Based on the YAWL model they generate service execution trees that represent execution scenarios. Each scenario is analysed to check properties such as lock-freedom, reachability and liveness. If a scenario is found that satisfies such properties, it is used to automatically synthesise an adapter.

Cavallaro et al. [108] presented an approach for invoking services that have a different interface or protocol from those originally expected. This is particularly useful in the context of dynamic service composition, i.e. services are selected at run-time based on an abstract description of the desired services. The approach provides few basic mapping functions for solving simple mismatches. The functions can be combined in a mapping script that can be used to solve complex mismatches. In a latter work [105], the authors assume the existence of Label Transition Systems (LTS) that model the interaction protocol of a service. Using the LTS model the authors demonstrate an approach for identifying the interaction protocol mapping between two services. In their most recent work [110], the authors improve the aforementioned approach, such that it automatically synthesises adapters by analysing the dataflow dependencies on the service interface and without the need for an LTS model to be provided.

The classification of the abovementioned approaches together with the classification of other adaptation approaches for the SCC layer is presented in Appendix I: Classification of Research Efforts on Monitoring and Adaptation. The classification presents how the various adaptation approaches for the SCC layer are positioned with respect to the methodology used for achieving adaptation, the architecture adopted in the approach and the characteristics of the decision implemented in the adaptation approach. The following paragraphs discuss the classification of the adaptation approaches for the SCC layer.
As far as the methodology used in the adaptation is concerned, most adaptation approaches are either implicit (e.g. [77], [86], [91], [96], [100], [101], [103–108], [110–112]), i.e. the adaptation is hardwired into the approach giving limited flexibility, or action-based (e.g. [73], [84], [87], [88], [90], [93–95], [102]), i.e. they provide a set of action that can be used to resolve an adaptation requirement. Implicit approaches are mostly concerned with either the optimisation of the quality of service (QoS), or the resolution of interface and interaction protocol mismatches. Additionally, approaches such as [45], [78], [79], [97] are also concerned with QoS optimisation and they are based on an utility function, which is an implicit approach as well. In contrast action-based approaches are concerned with the recovery of failures in service compositions. There are also less approaches such as [76], [83], [92], [109], [113] based on explicit variability, while there are fewer goal-based approaches such as [75], [114]. Most of these approaches support service instance actions (e.g. substitute a service in a service composition instance) and flow instance actions (e.g. substitute a flow in a service composition instance), while the approaches addressing mismatches support service and flow class actions.

Regarding the architecture of the approaches, the majority of the approaches are centralised, with the exception of [84], [90], [113] that are concerned with the distributed diagnosis and repair of service compositions. Additionally, almost all approaches require some sort of integration with the platform used for executing the adaptation subject. However, the level of intrusiveness to the platform ranges from a simple plug-in to be installed, to a whole customised platform, which supports a specific adaptation approach.

Concerning the decision in the reviewed adaptation approaches, most approaches are automated with few exceptions (e.g. [96], [100], [102], [104], [106], [107], [109]), which are interactive and therefore they require an iterative interaction with a human in order to complete their function. These interactive approaches are mostly concerned with the generation of adapters for addressing interface and interaction protocol mismatches. Most approaches that are action-based or based on explicit variability they are static with respect to the dynamicity of decision, while approaches that are goal-based are dynamic. Also, there are both static and dynamic implicit approaches. The majority of the approaches are reactive, while there are few approaches that are proactive. The proactive approaches are mostly concerned with the optimisation of service compositions.

The monitoring and adaptation approaches discussed thus far are concerned with a specific layer of the service-based application. The next section discusses the state of the art in multi-layer monitoring and adaptation of service-based applications.

3.3 Recent Research Efforts on Multi-Layer Monitoring and Adaptation

Thus far the literature review focused on the landscape of research efforts on monitoring and adaptation that target specific problem either in the Business Process Management (BPM) layer or in the Service Composition and Coordination (SCC) layer. As it is evident from the literature review, the existing research efforts are highly fragmented, while they address very specific problems from particular perspectives. The research efforts often focus on activities of either the monitoring process or the adaptation process, and therefore they are not adequate to address the multifaceted
problem of implementing adaptable service-based applications. As such, there is need to integrate the various monitoring and adaption approaches for realising multi-layer monitoring and adaptation (multi-layer M&A) solutions.

As explained in Section 2.7, in order to realise multi-layer monitoring and adaptation (multi-layer M&A) of a service-based application, the research community needs to address a number of issues, such as the lack of alignment of monitored events, the lack of adaptation effectiveness, the lack of compatibility of adaptation actions, and the lack of integrity due to adaptation. However, the Initial research efforts towards addressing the multi-layer monitoring and adaptation of SBA are rather scattered. Small number of existing research efforts associated with this topic can be clustered in (i) research efforts addressing diverse aspects of multi-layer M&A [18], [115–117], (ii) initial efforts presenting ad-hoc integrations of pre-existing adaptation and monitoring approaches [118], [119], and (iii) efforts targeting the architectural perspective of multi-layer M&A [120–123]. The subsequent sections discuss in more detail these research efforts.

3.3.1 Research Efforts on Diverse Aspects of Multi-layer Monitoring and Adaptation

Research efforts such as the ones presented in [18], [115] are concerned with the role of service-level agreements (SLA) in multi-layer M&A, while efforts in [116], [117] present approaches targeting the problem of deciding adaptation strategies. The following paragraphs present in more detail these recent research efforts addressing these aspects.

Fugini and Siadat [115] presented an approach for capturing user goals in SLA, which could be employed in multi-layer M&A. The authors introduced the Key Goal Indicators (KGI), which state how well an SBA achieves user goals. They argued that the existing relevant literature mostly deals with the technical service level aspects, therefore an SLA should not only capture key performance indicators and IT infrastructure parameters, but also indicators associated with user goals and the dependencies of the user goals with other parameters. The authors exemplify how to use linking rules to map conditions to conclusions for the user-level (e.g. if the actual availability of a service is greater than 90%, then the user goal for availability is high).

In the context of SLA for multi-layer M&A, we presented [18] an approach for identifying third-party services in each layer of the SBA, in order to establish the appropriate SLA for these services. We suggest a new perspective for the third-party services that concerns the type of external services utilised in each layer. The types are as follows, business services in the BPM layer, software services in SCC layer, and infrastructure services in SI layer. We argued about the necessity to define SLA for all types of services. We exemplified the approach to the definition of SLAs using a case study of a platform as a service framework, which has been developed as a service-based system.

Concerning the monitoring aspects of multi-layer M&A, Mos et al. [121] presented a platform that aims at monitoring and analysing of large-scale layered service-based systems, which have been deployed in distributed run-times. The platform incorporates a distributed service bus for propagating infrastructure level and application level events to the monitoring components, which use complex event processing to aggregate information (e.g. calculate aggregated metrics) collected at
the same level. The authors use various visualisation components to report the information collected at various levels. The main utility is to provide analytics to human operators regarding the operation of the service-based system.

Regarding the adaptation aspects of multi-layer M&A, Popescu et al. [85] proposed an approach for adaptation of multi-layer applications. The authors defined taxonomies of organisational, behaviour, and service mismatches. The approach uses adaptation templates as predefined patterns, which are solutions to a specific adaptation need. Monitored events trigger the process of matching adaptation templates, which expose adaptation logic as BPEL processes, using the three mismatch taxonomies. A matchmaking algorithm is used to verify the degree of match between an event and a mismatch, in order to select one of the adaptation templates for the identified mismatch. However, the approach can only resolve one mismatch at the time, since it is not able to cope with multiple mismatches, i.e., to decide on a sequence of adaptation templates and coordinate their execution, in order to address the mismatches.

Zengin et al. [117] presented an adaptation manager component for coordinating adaptations across layers. The adaptation manager can use existing analysis and adaptation tools, which target specific system concerns, to assess the impact of an adaptation at the different levels. The adaptation manager incorporates an analysis algorithm that incrementally constructs a tree with all strategies, starting from an initial adaptation trigger originated at any level. Each path in the tree represents an adaptation strategy, while the nodes of the path represent the adaptation actions, each addressing a specific adaptation need. However, the effectiveness of this approach is in question, since it selects an adaptation strategy by considering only the number of adaptation actions indicated by the strategy.

The abovementioned research efforts address diverse aspects of multi-layer M&A. The next section discusses recent research efforts on integrating various pre-existing monitoring and adaptation approaches.

3.3.2 Ad-hoc Integrations of Monitoring and Adaptation Processes

Research efforts such as [118], [119], [124] are the first to realise the integration of diverse adaptation and monitoring approaches for implementing multi-layer M&A solutions. The following paragraphs present in more detail these recent research efforts.

Schmieders et al. [118] present a framework for cross-layer adaptation that uses SLA prediction to avoid SLA violations. The main idea behind the framework is an approach that uses assumptions on the SBA's context. The framework employs SALMon [60] as a monitoring engine, which is integrated with QoS monitors for both Service Composition and Business Process Management layers. In case that a violation is detected, SALMon notifies the specification and assumption based detection (SPADE) component. SPADE is used to evaluate the impact of the violated assumption on the corresponding requirement. In case that adaptation is necessary, the adaptation strategy engine, implemented as a multi-agent system, decides on the adaptation strategy. Having decided an adaptation strategy, the adaptation enactment engine applies the decided adaptation strategy by performing service replacement, SLA re-negotiation, or service-infrastructure adaptation.
Guinea et al. [119] present a framework for multi-layer adaptation and monitoring of service-based applications. The framework integrates existing mechanisms for (i) monitoring of business processes [73] and the underlying service infrastructure [125], an (ii) event correlation mechanism and event bus for aggregating metrics such as reliability, average response time, rate, or domain-specific metrics [126], (iii) a quality of service analysis technique based on machine learning for constructing decision trees and identifying adaptation requirements [127], (iv) an approach to adaptation strategy generation based on the creation of graphs that represent adaptation strategies [117], and (v) a dynamic BPEL engine that allows the modification of BPEL processes at runtime.

Zeginis et al. [124] outlined a proposal for realising a cross-layer adaptation and monitoring framework. The proposed framework is based on an event monitoring and logging, event-pattern detection, and mapping between event patterns and appropriate adaptation strategies. The authors suggest employing the ASTRO monitoring framework [68] for collecting monitored events, which are then used by ESPER\(^\text{14}\), an event processing framework, for detecting patterns of events. For realising the reasoning process required to decide an appropriate adaptation strategy, the authors argue that this function can be efficiently accomplished by an event reasoning component [128].

The abovementioned recent research efforts have managed to realise an ad-hoc integration of monitoring and adaptation approaches, that is, they integrate a specific monitoring and analysis mechanisms with specific decision and adaptation enactment mechanisms. The next section discusses recent research efforts addressing the architectural aspects of multi-layer M&A.

### 3.3.3 Research Efforts on Architectural Aspects of Multi-layer Monitoring and Adaptation

A very small number of research efforts, such as [120], [122], [123], explicitly address the integration of diverse monitoring and adaptation mechanisms for multi-layer M&A. The existing research efforts addressing the architectural perspective are rather scarce. The following paragraphs present in more detail these recent research efforts.

Gjorven et al. [120] presented a server-side architecture integrating SOA-based adaptation mechanisms such as dynamic service selection. The architecture is based on a technology-agnostic adaptation middleware named QUA. QUA can be use adaptation mechanisms exposed by the application platform. The architecture is realised in the form of a middleware that can integrate with adaptation mechanism addressing service interfaces and application logic modifications. It is not clear if the authors target service-based systems or traditional applications that interact with Web services.

In [129], we presented an extensible architecture for integrating diverse service monitoring mechanisms in the same middleware. We have classified monitoring mechanisms into two classes, heavy-weight monitors able to observe and analyse multiple concerns (e.g. interaction protocol, failures, QoS), and light-weight monitors focusing on a single concern. Taking into consideration these two classes, we identified two solutions to enable dynamically deploy and remove monitors, by either (i) relying

\(^{14}\) Esper Event Processing Framework - http://esper.codehaus.org/
a single component acting as an information gateway for forwarding all intercepted messages from the monitored services to multiple monitors, or (ii) having a pool of diverse monitoring services attached directly to the monitored services at run-time. We presented three patterns for message interception that can be employed in both cases.

Similarly, Popescu et al. [123] briefly outlined an extensible architecture for integrating monitoring and adaptation units that may comprise a mixture of software-based mechanisms and human-based activities. The architecture is centered around an enterprise service bus (ESB), which serves as the main medium for propagating events from the service-based application to the monitoring mechanisms. In case that the monitoring mechanisms detect an adaptation requirement, they produce events that trigger adaptation patterns, which are defined using the Yet Another Workflow Language (YAML). Through YAML developers can invoke external adaptation mechanisms. Human operators can also produce or consume events, in order to either be notified of problems or enact adaptation actions manually.

3.3.4 Limitations of Current State of the Art in Multi-layer Monitoring and Adaptation

The previous sections presented the three groups of the small number of recent existing research efforts centered around the topic of multi-layer monitoring and adaptation (multi-layer M&A). These efforts focused on addressing diverse aspects of multi-layer M&A, ad-hoc integrations of pre-existing adaptation and monitoring approaches, and on the architectural perspective of multi-layer M&A. The first group of research efforts concerned with diverse aspects of multi-layer M&A focused on the definition of SLA, the monitoring of different layers and the adaptation across the layers. The small amount of research efforts is a sign of the immaturity of the field, which should set as a top priority the development of methods for integration of diverse monitoring and adaptation mechanisms, and their integrated coordination, in order to tackle the challenge of multi-layer M&A of service-based applications. We argue that the current state of the art in multi-layer M&A is significantly limited due to the lack of such a framework, which addresses the integration and coordination of diverse monitoring and adaptation processes.

Our argument is supported by the fact that all three research efforts [118], [119], [124], which constitute the current state of the art in multi-layer M&A frameworks follow an ad-hoc integration of pre-existing monitoring and adaptation mechanisms. The hardwired approach to integration employed by these efforts limits their capabilities in terms of extensibility, dynamism, and effectiveness. These efforts address multi-layer M&A from a very specific perspective, while they focus more on the operation of the individual mechanisms, without considering the effectiveness of the integrated mechanisms as a whole and across the layers of an SBA. Additionally, they leave aside the extensibility aspect concerned with the integration of other adaptation and monitoring mechanisms, which can be used for addressing other concerns of the cross-layer setting.

Moreover, the research efforts [120], [122], [123], which constitute the current state of the art concerning the architectural aspects of MLM, outline initial ideas towards addressing the challenges associated with the architectural aspects of multi-layer M&A. However, these efforts have only managed to scratch the surface of the problem, since they only presented insights and partial solutions, which address a very
narrow aspect of the architecture required for the integration and the coordination of diverse monitoring and adaptation mechanisms. Finally, these efforts ignore completely the issue of dynamism in multi-layer M&A, i.e. the ability to deploy or remove monitoring and adaptation mechanisms during run-time.
4 Towards Engineering the Integration and the Coordination of Monitoring and Adaptation Processes

As discussed in the previous section, the state of the art in multi-layer monitoring and adaptation (multi-layer M&A) of service-based applications has demonstrated few multi-layer M&A solutions, which have been based on the ad-hoc integration of pre-existing monitoring and adaptation mechanisms. Such efforts realised the multi-layer M&A of service-based applications from a particular perspective and they addressed a very specific problem. We argue that the current state of the art in multi-layer M&A is significantly limited due to the hardwired approach to integration of mechanisms employed in the existing multi-layer M&A solutions.

As such, a key challenge associated with multi-layer M&A of service-based applications lies in engineering multi-layer M&A solutions that are flexible, extensible, dynamic, and effective. The flexibility of such solutions consists in their support for extending their capabilities through the integration of mechanisms for monitoring, analysis, decision, and adaptation, while also supporting dynamism through the deployment and removal of such mechanisms at run-time. The effectiveness of such dynamic solutions would rely on an additional control loop for coordinating the parallel operation of the integrated mechanisms.

The abovementioned challenge comprises three main problems that call for a concrete solution. The first problem is how to engineer the integration of diverse mechanisms and ensure the dissemination of information among them. The second problem is what information needs to be conveyed among the mechanisms and how this information is standardised. The last problem is how to coordinate the operation of the integrated mechanisms, such that they operate in an effective way. Addressing all three problems would provide the baseline for architecting flexible, extensible, dynamic, and effective multi-layer M&A solutions for service-based applications.

In order to advance the existing state of the art in multi-layer M&A, this report puts forward the investigation of a framework for engineering the integration and the coordination of monitoring and adaptation mechanisms, in order to facilitate the implementation of flexible, extensible, dynamic, and effective multi-layer M&A solutions for service-based applications.

The proposed framework comprises three main pillars, each addressing one of the aforementioned problems, while contributing as a whole to the integration and the coordination of monitoring and adaptations mechanisms. The first pillar of the framework is concerned with the architectural aspects of the integration, i.e. how to enable the loosely coupled integration of the interacting monitoring and adaptation mechanisms, while at the same time ensuring the propagation of information between them. The second pillar is concerned with the conveyance of information among mechanisms, i.e. what kind of information needs to be conveyed among mechanisms, such that their coordination is possible. The last pillar is concerned with the coordination of the monitoring and the adaptation mechanisms, i.e. how to achieve near real-time operational behaviour in the multi-layer M&A that will be capable of detecting situations occurred in monitoring and adaptation mechanisms, such that coordination actions are performed.
The exploration of such a framework is proposed as a timely and a relevant subject for research at the PhD level. The subsequent sections discuss the main pillars of the framework. They set the specific research goal and objectives, while they argue about the novelty of the proposed research directions and the potential areas of contribution.

4.1 Integrating Diverse Monitoring and Adaptation Processes

The mechanisms used in a monitoring and adaptation process usually target a specific layer of a service-based application (SBA). The monitoring process uses a monitoring mechanism for detecting problematic situations, which are further analysed by analysis mechanisms to identify adaptation requirements. The adaptation process uses a decision mechanism to select an appropriate adaption strategy, which is enacted through an adaptation mechanism. The mechanisms involved in a monitoring and adaptation process usually address a very specific problem.

As such and in order to enable the loosely coupled integration of different monitoring and adaptation processes, it is required to devise an architecture that will support the interaction of diverse mechanisms utilised by these processes. The architecture has to be flexible enough to accommodate the integration complexity of mechanisms, which were not designed to interoperate. In this perspective, the mechanisms to be integrated are treated as black boxes, which do not have to be aware of each other’s existence and functions, since they are only expected to produce and consume information through a standardised medium. Based on this perspective, diverse monitoring and adaptation mechanisms could be used as building blocks for implementing multi-layer M&A solutions.

Necessary ingredients of an architecture for realising the aforementioned approach to integration, are provisions for information dissemination among mechanisms through standardised channels, a central repository for maintaining the knowledge associated with the operation of the mechanisms, as well as, the capabilities offered by the mechanisms.

By relying on such an architecture, it could be possible to construct multi-layer M&A solutions that are flexible, extensible, and dynamic. The flexibility of multi-layer M&A solutions build in such a way consists in the ability to reconfigure the capabilities of the solution. Such solutions would be highly extensible, since additional mechanisms can always be integrated. Moreover, such flexibility will be facilitated also by deploying or removing mechanisms during the normal operation of the multi-layer M&A solution, such that the required mechanisms are utilised when there is a particular need, or they are suspend their operation to minimise the performance overhead.

In order to develop the architecture sketched thus far, a direction that we will investigate is concerned with the convergence of the service-oriented architecture and the event-driven architecture, often referred as event-driven SOA [130–132]. We think that in the context of an event-driven SOA, diverse monitoring and adaptation mechanisms can be treated as loosely coupled services, which act as event producers and event consumers, in order to disseminate information and react to events when it is necessary.
The abovementioned architecture is only part of a solution for realising multi-layer M&A solutions. By having only a flexible architecture does not address neither integration of the information conveyed among mechanisms, nor the coordination of their operation. The next sections elaborate more precisely on these issues.

4.2 Conveying Information among Monitoring and Adaptation Processes

The previous section sketched the architecture required for realising flexible multi-layer monitoring and adaptation (multi-layer M&A) solutions for adaptable service-based applications. The architecture aims to facilitate the integration of diverse monitoring and adaptation processes, while also propagating information among these processes to facilitate information dissemination. However, a particular problem with the information dissemination between diverse processes consists in the heterogeneity of information, which are produced and consumed by the diverse mechanisms used to realise the processes.

Every mechanism usually has its own particular representation of inputs and outputs that use specific syntax and semantics. Therefore, the monitoring and the adaptation mechanisms, which have not been designed to interoperate, are not able to convey information associated with problematic situations that they have detected, adaptation requirements that they have identified, adaptation strategies that they have selected, and adaptation actions that they need to be executed.

As such and in order to enable the conveyance of information among diverse monitoring and adaptation processes, it is necessary to define a common set of information produced and consumed by different mechanisms in such processes, such that it is used as the common language among these mechanisms. The common set of information could be realised though as an expressive event schema (or model) specifying various events expected to be produced and consumed by the monitoring and adaptation mechanisms. The event schema has to define the information conveyed by a specific event, as well as, define the relationships between events. Additionally, such an event schema should facilitate its extension through the inclusion of new events, since the event schema cannot foresee all possible events produced and consumed by mechanisms to be integrated.

The event schema is considered as the second important pillar of the proposed framework, which aims to enable the realisation of flexible multi-layer monitoring and adaptation solutions. We think that the event schema will not only solve the problem of information conveyance, but it will also serve as the basis for coordinating the operation of the diverse monitoring and adaptation processes, by providing the required means to convey coordination information. The next section discusses in more detail the insights concerning the coordination of monitoring and adaptation processes within the proposed framework.

4.3 Coordinating Monitoring and Adaptation Processes

Thus far it was argued that by relying on a flexible architecture and an expressive event schema, it can be possible to address the problem of integration of diverse monitoring and adaptation processes. However, overcoming the problem of integration is not enough for realising multi-layer monitoring and adaptation (multi-layer M&A) solutions, since the integrated monitoring and adaptation processes will still operate in
isolation. The mechanisms utilised in these processes still react in an uncoordinated fashion, often producing conflicting information or enacting conflicting actions.

As such and in order to enable the effective operation of diverse monitoring and adaptation processes, it is necessary to develop a method for coordinating the operation of the integrated mechanisms, such that the mechanisms operate in the frame of a predefined set of goals. The coordination should happen on the basis of the information conveyed by the integrated mechanisms through feedback channels, such that situations requiring the coordination of actions are detected. In the event of such situations, coordination information should be disseminated to the integrated mechanisms through control channels.

Such a coordination method should correlate the observed information, in order to be capable of detecting a situation that will lead to an undesirable behaviour of the integrated mechanisms. The coordination method should also incorporate some sort of reasoning to decide the appropriate coordination information, which will be disseminated to the integrated mechanisms in order to prevent the undesirable behaviour. Upon the occurrence of a situation, the coordination process could evaluate a number of conditions, in order to decide the appropriate coordination actions.

The coordination method should abstract away the coordination logic from the integrated mechanisms, while supporting the modification of the logic without requiring the modification of mechanisms. The coordination process should be enacting the coordination through a central point of control within the architecture, which will also provide the medium for disseminating coordination information.

In order to develop the coordination method, we will investigate the use of event processing technologies [133]. We think that event processing is a promising direction, since it is capable of detecting event-patterns, as well as, relationships between events. However, how such features can be exploited in the coordination process require further investigations.

4.4 Research Goal and Objectives

This PhD project targets the challenge of multi-layer adaptation and monitoring of service-based applications, more particularly the need for integration and coordination of diverse monitoring and the adaptation mechanisms, and proposes to address this challenge through a framework for the integration and the coordination of monitoring and adaptation mechanisms.

This PhD project aims to develop a framework for the integration and the coordination of existing monitoring and adaptation mechanisms. The framework will be based upon a flexible architecture for the loosely coupled integration of mechanisms; an expressive event schema for facilitating the conveyance of information between the mechanisms; and a coordination method for controlling the operation of the integrated mechanisms.

The PhD project must fulfil five objectives to achieve its goal:

1) To develop an architecture that is extensible enough to facilitate the integration of diverse monitoring and adaptation mechanisms, while also enabling the propagation of information between these mechanisms. Such an architecture aims to accommodate the complexity of an environment that
comprises multiple, possible heterogeneous, monitoring and adaptation mechanisms, which focus on a great variety of activities, from different perspectives, and with varying concerns. The architecture should account for the dynamicity through the appropriate provisions for enabling the deployment and removal of mechanisms during run-time.

2) To develop an event schema that is expressive enough to facilitate the conveyance of information between monitoring and adaptation mechanisms, while at the same time supporting the conveyance of feedback and coordination information. The event model will explicitly specify the events, which are expected to be produced and consumed by the integrated mechanisms, while also facilitating its extension through the inclusion of new events when new mechanisms are being integrated.

3) To develop a method for coordinating the operation of the monitoring and the adaptation mechanisms, in order to avoid undesirable behaviour of the integrated mechanisms as a whole. The coordination process should comprise the detection of situations calling for coordination, reasoning about the action required for coordinating the operation of the mechanisms involved in the situation, and disseminate the coordination information to the involved mechanisms.

4) To implement a prototype of the integration and coordination framework in an open-source service-oriented middleware, such that a proof of concept implementation is made available. The prototype should realise all three parts of the framework, i.e. the sketched architecture, the event schema, and the coordination method.

5) To evaluate the prototype implementation of the framework, demonstrating a proof of concept for supporting the integration and the coordination of diverse monitoring and adaptation mechanisms, in order to realise multi-layer monitoring and adaptation of service-based applications. The evaluation will be carried mostly through experimental simulation, while also the integration of concrete mechanisms will be explored, based on their availability.

The following sections discuss the novelty of the proposed research directions and mention the potential areas that these directions could contribute.

**4.5 Novelty of the Proposed Research Directions and Potential Areas of Contribution**

As it has been covered in the literature review (see Section 3.3.4), existing research works in the subject of multi-layer monitoring and adaptation (multi-layer M&A) of service-based applications, either present static ad-hoc integration between different monitoring and approaches, or partial solutions to a very narrow aspect of the overall architecture required for the integration of diverse monitoring and adaptation mechanisms.

The proposed project builds on the existing literature of multi-layer M&A and proposes to address the lack of a concrete framework for integrating and coordinating monitoring and adaptation mechanisms, through the development of such a framework. To the best of the author’s knowledge, the investigation of such a framework has not been the subject of any known research studies so far.
As such, this project could potentially contribute to research communities, such as the Service-Oriented Computing community and particularly the sub-community investigating the challenges of self-adaptive service-oriented systems, as well as, the Autonomic Computing community.
5 Current Progress and Future Work

This section outlines the current progress and the envisaged plan for future research activities. The author was registered on 1st February 2010 as a part-time MPhil candidate, who aims for a PhD. Since 15th November 2010 he is employed as a research associate at the South East European Research Centre, where he is involved in various funded research projects, some of which are particularly relevant to his own research. Since the registration in 2010, he carried a number of activities related to research, dissemination and participation in the research community. These activities are described in the subsequent sections.

5.1 Research Activities

Thus far, a number of research activities have been carried. The activities comprise an extensive literature review of various topics, involvement in relevant research projects, and experimental implementation. This section summarises briefly these activities.

The literature review activity encompassed investigating a number of topics, which are summarised below.

- Service-oriented computing: Fundamentals of service-oriented architecture, service-orientation, service science;
- Autonomic Computing: Fundamentals and challenges for engineering autonomic systems;
- Cloud computing: Fundamentals, challenges, and technologies for engineering cloud-based systems;
- Event-driven computing: Fundamentals of event-driven architecture, preliminary investigation of event processing;
- Web services technologies: SOAP, REST, BPEL, and various standards for Web services;
- SOA Middleware: Fundamentals, OSGI-based architectures, SOA offerings;
- Service-based applications: Fundamentals and challenges for engineering such applications;
- Adaptable service-based applications: Fundamentals of monitoring and adaptation, thorough investigation of techniques for the whole monitoring and adaptation life-cycle (monitoring, analysis decision, and adaptation) of service-based applications.

The author has been involved in research projects relevant to his own research. These projects and the activities carried out in their context is summarised below.

- S-Cube\textsuperscript{15} - the European Network of Excellence in Software Services and Systems that aimed to establish an integrated, multidisciplinary, vibrant research community focusing on key research areas of Software Services and Systems, such as engineering and design, adaptation and monitoring, as well as quality definition, negation and quality assurance. The author was actively involved in a wide range of joint research activities such as brain storming sessions and project meetings, while he also provided contributions for the

\textsuperscript{15} S-Cube - http://www.s-cube-network.eu/
Towards Engineering Multi-Layer Monitoring and Adaptation of Service-Based Applications

...project deliverables and he presented insights from his own research. This project helped the author to engage with the research community and provided the basis for on-going and future research collaborations.

- CAST\textsuperscript{16} - Enabling Customisation of SaaS Applications by Third parties that aimed to the development of a framework for customisation of SaaS applications. The author contributed to the overall development of the governance registry & repository system, which is part of the overall framework developed within the project. The author was also the main contributor to the development of a monitoring engine, which aimed at monitoring the compliance of third-party services (utilised by SaaS applications deployed on the platform) to service-level agreements. The developed was based on the extension of WSO2 Governance Registry\textsuperscript{17}.

Except from the experimental software development in the scope of relevant research projects, the author carried out his own experimental implementation activities, which were concerned the development of a prototype [129] for monitoring the behaviour of conversational Web services.

The research activities described in this section led to a number of dissemination activities reported in the next section.

5.2 Dissemination Activities

Thus far the author performed a number of dissemination activities in the form of academic publications and presentations in various academic events. The list of relevant publications is provided below.


\textsuperscript{16} CAST - http://www.cast-project.eu/
\textsuperscript{17} WSO2 Governance Registry - http://wso2.org/library/governance-registry
5.3 Participation in the Research Community

Throughout the duration of the PhD project, the author pursues actively his participation in the research community by assisting in the peer-review process of academic publications, while also contributing to the organisation of events.

The author has served as a reviewer in the following:

- Software: Practice and Experience (SPE Journal)
- 6th IEEE/ACM International Workshop on Automation of Software Test
- 1st International Workshop on Adaptive Services for the Future Internet
- 6th South East European Doctoral Student Conference
- 7th South East European Doctoral Student Conference

Moreover, the author was invited to participate in a panel discussion in the International Workshop on Quality Assurance for Service-based applications (QASBA 2011), where he also gave an invited presentation about the S-Cube project.

Finally, the author has also actively contributed to the organisation of the series of the South East European Doctoral Student Conferences.

5.4 Timetable for the Remainder of the Research Project

It is assumed that the PhD thesis will be submitted by December 2014. The number of research activities that have been envisaged until then comprise the following:

March 2012- May 2012

1. The main task for this period is the authoring of publication, because a lot of submission deadlines of several important conferences have been scheduled.
2. In parallel, background reading on event processing is envisaged. The aim is to familiarise with the concepts of the field and identify key literature on the intersection of event processing and autonomic computing. Such literature can be quite helpful for the proposed research.

June 2012- July 2012

1. The main task for this period is the definition of concrete examples and case studies that will describe the operation of the framework.
2. The definition of the examples will involve the development of a small service-based application, which will be used as a pilot study for multi-layer M&A.

August 2012- November 2012

1. During this period, it is envisaged to focus on the coverage of service-orientated architecture (SOA) with event-driven architecture (EDA). Since, there is already a very good understanding of SOA, more attention will be given to EDA. The aim is to build a solid understanding of the concepts and the various messaging patterns used in EDA and also explore synergies between SOA and EDA.
2. The second task envisaged for this period is a brief evaluation and an extensive experimentation with various eventing infrastructures provided by middleware, such as WSO2 Carbon and JBoss AS. The aim is to investigate how such middleware can be leveraged for developing a communication channel for integrating diverse M&A mechanisms.
3. The last task for this period is the implementation of an experimental prototype using an appropriate messaging pattern (e.g., publish/subscribe) simulating the communication of M&A mechanisms.
4. Refinement of the examples or case studies from the point of view of the architecture

December 2012 – January 2012
1. During this period it is envisaged to focus on the identification of the information required to be conveyed among integrated mechanisms. Analysis of a sample of existing M&A mechanisms to identify the events, which need to be communicated.
2. Analysis of the selected events with the aim to identify relationships among them. Specification of the event schema.
3. Implementation of the schema into a concrete representation, on the basis of the selected middleware.
4. Refinement of the examples or the case studies from the point of view of the event schema.

February 2012 – May 2012
1. During this period it is envisaged to focus on the development of the coordination method. Definition of scenarios that involve conflicting actions. Brief investigation of decentralised coordination approaches inspired from multi-agent systems.
2. Development of the core coordination approach on the basis of complex event processing for detecting conflicting situations and coordination rules, which produce coordination events.
3. Experimental implementation of scenarios that represent situations that involve conflicting actions and experimentation with the coordination method.
4. Refinement of the examples or the case studies from the point of view of the coordination.

June 2013 – August 2013
1. During this period, the core implementation of a prototype middleware is envisaged. The prototype will be developed based on WSO2. The aim of the prototype is to act as a proof of concept for the integration architecture, the event schema and the coordination method developed earlier.
2. Integrate third-party M&A mechanisms, based on their availability.

September 2013 – December 2013
1. During this period, critical evaluation of the prototype will be performed. The aim of the evaluation will be to demonstrate the flexibility, extensibility, dynamicity and effectiveness of the proposed framework.

January 2014 – December 2014
• Authoring of final thesis
References


Towards Engineering Multi-Layer Monitoring and Adaptation of Service-Based Applications


Towards Engineering Multi-Layer Monitoring and Adaptation of Service-Based Applications


Towards Engineering Multi-Layer Monitoring and Adaptation of Service-Based Applications


Towards Engineering Multi-Layer Monitoring and Adaptation of Service-Based Applications


Towards Engineering Multi-Layer Monitoring and Adaptation of Service-Based Applications


Towards Engineering Multi-Layer Monitoring and Adaptation of Service-Based Applications


Towards Engineering Multi-Layer Monitoring and Adaptation of Service-Based Applications


# Appendix I: Classification of Research Efforts on Monitoring and Adaptation of Service-Based Applications

## Classification of monitoring approaches for the BPM layer

<table>
<thead>
<tr>
<th>Work</th>
<th>Goal</th>
<th>Solution</th>
<th>Methodology</th>
<th>Architecture</th>
<th>Analysis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lazovik et al. [32]</td>
<td>Monitor the compliance of BPEL processing to business policies</td>
<td>The authors introduced the XML Service Assertion Language (XSAL) used for defining assertions, which can be simple reachability conditions, conditions that need to be satisfied through the execution, and conditions that check the evolution of a particular business property</td>
<td>Push mode</td>
<td>Centralised</td>
<td>Provider</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Blocking</td>
<td>Platform</td>
<td>Messages</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Reactive</td>
<td>Automated</td>
<td></td>
</tr>
<tr>
<td>Beeri et al. [27]</td>
<td>Monitor interesting patterns in a BPEL process from a business perspective</td>
<td>The authors presented a visual editor that is used to express patterns that are translated to Directed Acyclic Graphs (DAG), which are then used for pattern matching of the execution traces, in order to detect if the specified patterns occurred during the execution of the process</td>
<td>Polling mode</td>
<td>Centralised</td>
<td>Provider</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Non-blocking</td>
<td>Platform</td>
<td>Logs</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Reactive</td>
<td>Post-mortem</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Automated</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Wetzstein et al. [28]</td>
<td>Monitor the performance of a BPEL process from a business perspective</td>
<td>The authors presented an event-based monitoring approach. Events are pushed to the monitoring engine that analyses the events to measure various performance metrics such as duration of an activity, number of exactions, time the activity occurred etc</td>
<td>Push mode</td>
<td>Centralised</td>
<td>Provider</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Non-blocking</td>
<td>Platform</td>
<td>Events</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Reactive</td>
<td>Automated</td>
<td></td>
</tr>
<tr>
<td>Authors</td>
<td>Monitoring Activity</td>
<td>Description</td>
<td>Mode</td>
<td>Blocking</td>
<td>Platform</td>
</tr>
<tr>
<td>-------------------------</td>
<td>--------------------------------------------------------------------------------------</td>
<td>-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
<td>--------------</td>
<td>------------</td>
<td>-------------</td>
</tr>
<tr>
<td>Wetzstein et al. [30]</td>
<td>Monitor the performance of a BPEL process from a business perspective and analyse the causes of low performance</td>
<td>The authors present an approach that based on the historical measurements collected through events, uses machine learning to generate a dependency tree, which represents the process performance metrics that have influenced a key performance indicator</td>
<td>Push mode</td>
<td>Non-blocking</td>
<td>Centralised</td>
</tr>
<tr>
<td>Leitner et al. [31]</td>
<td>Monitor and predict SLA violations</td>
<td>The authors use historical monitored information to train a prediction model using machine learning techniques. The observed and the estimated QoS information and the process instance data are passed to the prediction model, which produces a numerical estimation of the SLO value.</td>
<td>Push mode</td>
<td>Non-blocking</td>
<td>Centralised</td>
</tr>
<tr>
<td>Wetzstein et al. [29]</td>
<td>Monitor the performance of a BPEL process that spans across different organizations from a business perspective</td>
<td>The authors use a monitoring agreement is used between the organisations that describes the events that should be exposed by each organisation and these events are then pushed to a global monitoring component, which uses complex event processing to correlate the events</td>
<td>Push mode</td>
<td>Non-blocking</td>
<td>Decentralised</td>
</tr>
<tr>
<td>Dustdar et al. [33]</td>
<td>Monitor the compliance of BPEL processing to business policies</td>
<td>The authors combine complex event processing with a model expressing information about the observed events.</td>
<td>Push mode</td>
<td>Non-blocking</td>
<td>Centralised</td>
</tr>
</tbody>
</table>
## Classification of adaptation approaches for the BPM layer

<table>
<thead>
<tr>
<th>Work</th>
<th>Goal</th>
<th>Solution</th>
<th>Methodology</th>
<th>Architecture</th>
<th>Decision</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reichert and Dadam [34]</td>
<td>Adaptation of processes for incorporating structural changes</td>
<td>The authors defined a set of actions for transforming a running process by modifying its control flow. The approach uses Petri Nets to check the correctness of the transformations</td>
<td>Action-based</td>
<td>Centralised</td>
<td>Independent</td>
</tr>
<tr>
<td>Rinderle and Reichert [39]</td>
<td>Adaptation of access control in business processes</td>
<td>The authors defined a formal semantics for reasoning about the correctness of model and rule changes for access control in business process. They demonstrate how they can be used for manual adaptation of a BPEL process</td>
<td>Goal-based</td>
<td>Centralised</td>
<td>Independent</td>
</tr>
<tr>
<td>Hallerbach et al. [35]</td>
<td>Management of process variants and adaptation of BPEL process instances</td>
<td>The authors used explicit variability techniques to define the variation points and the process variants at design-time, while the use context-rules to select the appropriate process variant at run-time</td>
<td>Explicit variability</td>
<td>Centralised</td>
<td>Platform</td>
</tr>
<tr>
<td>Ly et al. [40]</td>
<td>Verification of the semantic correctness of the changes performed by adaptation actions</td>
<td>The authors developed a formal framework for defining mutual exclusion constraints and dependency constraints in processes and verify if this constraints are satisfied from the adaptation actions</td>
<td>Goal-based</td>
<td>Centralised</td>
<td>Independent</td>
</tr>
<tr>
<td>Authors</td>
<td>Approach</td>
<td>Description</td>
<td>Variables</td>
<td>Flow</td>
<td>Platform</td>
</tr>
<tr>
<td>---------</td>
<td>----------</td>
<td>-------------</td>
<td>-----------</td>
<td>------</td>
<td>----------</td>
</tr>
<tr>
<td>Koning et al. [134]</td>
<td>Support variability in a BPEL process</td>
<td>The authors presented a BPEL extension that introduces variability concepts in the standard BPEL language. This approach enables both the switching between variants with different QoS and the ability to capture a family of processes within one process definition</td>
<td>Explicit variability</td>
<td>Flow instance</td>
<td>Centralised</td>
</tr>
<tr>
<td>Kazhamiakin et al. [38]</td>
<td>Adaptation of BPEL process based on KPIs</td>
<td>The authors used decision trees to represent the influential factors of KPIs and they use the same tree to extract a set of adaptation requirements and find an adaptation strategy consisting of a set of adaptation actions which takes into account both positive and negative effects of adaptation actions on KPIs</td>
<td>Action-based</td>
<td>Flow instance</td>
<td>Centralised</td>
</tr>
<tr>
<td>Leitner et al. [37]</td>
<td>Adaptation of BPEL process to prevent SLA violations</td>
<td>The authors used aspect-oriented programming and process fragments to substitute parts of a BPEL process, based on predictions for the QoS of the process and the impact of the available fragments</td>
<td>Explicit variability</td>
<td>Flow instance</td>
<td>Centralised</td>
</tr>
<tr>
<td>Hermosillo et al. [135]</td>
<td>Adaptation of a BPEL process based on context changes</td>
<td>The authors presented a framework that comprises a BPEL extension for defining adaptation strategies and referencing these strategies from within a BPEL process. Complex event processing is used for identifying context changes, which are mapped to specific adaptation strategies</td>
<td>Action-based</td>
<td>Flow instance</td>
<td>Centralised</td>
</tr>
<tr>
<td>Weidmann et al. [36]</td>
<td>Adaptation of BPEL processes based on a reference process model</td>
<td>The authors introduced the concept of a standardised variable process model, which is used as the basis for individualisation of BPEL processes according to organisational policies</td>
<td>Explicit variability</td>
<td>Flow class</td>
<td>Centralised</td>
</tr>
</tbody>
</table>
### Classification of monitoring approaches for the SCC layer

<table>
<thead>
<tr>
<th>Work</th>
<th>Goal</th>
<th>Solution</th>
<th>Methodology</th>
<th>Architecture</th>
<th>Analysis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Keller and Ludwig [59]</td>
<td>Monitoring Service Level Agreements for Web Services</td>
<td>The authors presented an XML-based language that is used to define SLAs. They used various types of monitoring services (e.g., measurement and condition evaluation) to monitor the compliance of the monitored Web services to the SLAs</td>
<td>Push mode and Polling mode</td>
<td>Decentralised Platform</td>
<td>Messages and Measurements Reactive Automated</td>
</tr>
<tr>
<td>Robinson [54]</td>
<td>Monitoring the conformance of a Web service to functional and non-functional requirements</td>
<td>The authors used specifications expressing the requirements to be monitored, based on temporal logic. Multiple monitoring services evaluate messages from the monitored services to check if they satisfy the expressed requirements</td>
<td>Push mode Non-blocking</td>
<td>Decentralised Platform</td>
<td>Messages Reactive Automated</td>
</tr>
<tr>
<td>Baresi et al. [47]</td>
<td>Monitor the correctness of the constituent Web services in a BPEL process</td>
<td>The authors used assertions that are expressed as comments within BPEL processes. The comments are translated to BPEL activities, which invoke external Web services acting as dedicate monitors that evaluate the assertions</td>
<td>Push mode Blocking</td>
<td>Centralised Subject Provider</td>
<td>Messages Reactive Automated</td>
</tr>
<tr>
<td>Mahbub and Spanoudakis [42]</td>
<td>Monitor the correctness of a service composition according to requirements</td>
<td>The authors presented an approach that records events produced by the process engine are to an event database, which is used by the monitoring component to analyse the events using event calculus, in order to detect if the execution of the composition is consistent</td>
<td>Push mode Non-blocking</td>
<td>Centralised Platform Provider</td>
<td>Events Reactive Automated</td>
</tr>
<tr>
<td>Lazovik et al. [32]</td>
<td>Monitor the correctness of BPEL processes</td>
<td>The authors used the XML Service Assertion Language (XSAL) for defining assertions, which can be simple reachability conditions, conditions that need to be satisfied through the execution, and conditions that check the evolution of a particular functional property</td>
<td>Push mode</td>
<td>Blocking</td>
<td>Centralised Platform</td>
</tr>
<tr>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
</tr>
<tr>
<td>Barbon et al. [43]</td>
<td>Monitor the correctness of a service composition</td>
<td>The authors presented an approach to check the flow and temporal properties of the composition. The former is done by automatically translating BPEL to State Transition Systems, while the latter is performed based on expressions formulated using Linear Temporal Logic</td>
<td>Push mode</td>
<td>Non-blocking</td>
<td>Centralised Platform</td>
</tr>
<tr>
<td>Li et al. [55]</td>
<td>Monitor the correctness of the conversational protocol of a Web service</td>
<td>The authors presented an approach that specifies interaction constraints, which are checked using Finite State Automata (FSA) according to the Specification Pattern System</td>
<td>Push mode</td>
<td>Non-blocking</td>
<td>Centralised Platform</td>
</tr>
<tr>
<td>Ardissono et al. [49]</td>
<td>Monitor and analysis of the causes of the exceptions in BPEL processes</td>
<td>The authors presented a framework that uses diagnostic services deployed on the individual service providers participating in the process and this local diagnostic services perform local diagnostics of the exception occurred, while the send the results of the local diagnosis to the global diagnostic service, which generated a global hypothesis about the failure</td>
<td>Push mode</td>
<td>Blocking</td>
<td>Decentralised Platform</td>
</tr>
<tr>
<td>Bianculli, and Ghezzi [51]</td>
<td>Monitor the correctness of the conversational protocol of a stateful web service within a BPEL process</td>
<td>The authors specified the expected behaviour of the monitored service using the Algebraics specifications formalism and monitoring takes place within the BPEL process engine by using AOP to intercept messages at predefined pointcuts in order to check if the flow of the messages matches the expected</td>
<td>Push mode</td>
<td>Blocking (?)</td>
<td>Centralised Platform</td>
</tr>
</tbody>
</table>
Towards an integration and coordination framework for architecting multi-layered M&A of SBA

<table>
<thead>
<tr>
<th>Authors</th>
<th>Monitoring Service Level Agreements for Web Services</th>
<th>The authors presented SOA architecture for measuring various QoS properties of Web services. The collected measurements are analysed using rules to check for violations</th>
<th>Push mode</th>
<th>Non-blocking</th>
<th>Decentralised</th>
<th>Platform</th>
<th>Provider</th>
<th>Events</th>
<th>Reactive</th>
<th>Automated</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ameller and Franch [60]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Authors</th>
<th>Monitoring Service Level Agreements for Web Services</th>
<th>The authors used BPEL to define the choreography model that is automatically transformed to Petri-nets, which are then used to parse the execution log to check if the Petri-net model is able to “parse” every execution sequence</th>
<th>Push mode</th>
<th>Non-blocking</th>
<th>Centralised</th>
<th>Independent</th>
<th>Provider</th>
<th>Logs</th>
<th>Post-mortem</th>
<th>Interactive</th>
</tr>
</thead>
<tbody>
<tr>
<td>Aalst et al. [52]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Authors</th>
<th>Monitoring Service Level Agreements for Web Services</th>
<th>The authors used SLAng to specify latency, reliability, and throughput requirements that are then transformed to Timed Automata, such that the language accepted by the automaton characterises exactly the violations of the specifications and each automation is used by a checker component that feeds the automation with the intercepted messages</th>
<th>Push mode</th>
<th>Blocking</th>
<th>Centralised</th>
<th>Platform</th>
<th>Provider, Consumer, and Third-party</th>
<th>Messages</th>
<th>Reactive</th>
<th>Automated</th>
</tr>
</thead>
<tbody>
<tr>
<td>Raimondi et al. [61]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Authors</th>
<th>Monitoring QoS of BPEL processes</th>
<th>The authors predefined few formulas for various QoS attributes (e.g. response time, availability, accuracy, etc.), which are being measured at predefined pointcuts using AOP</th>
<th>Push mode</th>
<th>Non-blocking</th>
<th>Centralised</th>
<th>Platform</th>
<th>Provider</th>
<th>Messages</th>
<th>Reactive</th>
<th>Automated</th>
</tr>
</thead>
<tbody>
<tr>
<td>Moser et al. [45]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Authors</th>
<th>Monitor dependencies between SLAs to analyse the cause of failures</th>
<th>The authors presented an approach that combines dependency models for cost and response time with an impact analysis model. The combination is used to evaluate execution logs to identify the problematic services</th>
<th>Push mode</th>
<th>Non-blocking</th>
<th>Centralised</th>
<th>Platform</th>
<th>Provider</th>
<th>Logs</th>
<th>Post-mortem</th>
<th>Interactive</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bodenstaff et al. [62]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Authors/References</td>
<td>Detection and diagnosis of failures in BPEL processes</td>
<td>The authors used the Workflow Graph formalism to specify a coverability graph for the monitored BPEL process, which is checked using the coverability graph to check the correctness of the process execution flow.</td>
<td>Push mode</td>
<td>Blocking</td>
<td>Centralised</td>
<td>Platform</td>
<td>Provider</td>
<td>Messages</td>
<td>Proactive</td>
<td>Automated</td>
</tr>
<tr>
<td>--------------------</td>
<td>------------------------------------------------------</td>
<td>--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
<td>-----------</td>
<td>-----------</td>
<td>-----------</td>
<td>---------</td>
<td>---------</td>
<td>----------</td>
<td>-----------</td>
<td>-----------</td>
</tr>
<tr>
<td>Hielscher et al. [136]</td>
<td>Detect failures in Web services proactively</td>
<td>The authors presented a framework that exploits online testing approaches to proactively trigger adaptations. Testing of the constituent Web services happens in parallel with the execution of the SBA, in order to reveal potential problems.</td>
<td>Polling mode</td>
<td>Blocking</td>
<td>Centralised</td>
<td>Platform</td>
<td>Consumer</td>
<td>Messages</td>
<td>Proactive</td>
<td>Automated</td>
</tr>
<tr>
<td>Dranidis et al. [56]</td>
<td>Monitor the correctness of the conversational protocol of a stateful web service</td>
<td>The authors modelled the conversational protocol using the Stream X-Machines formalism and the specified model is animated in parallel with the service execution to service as an oracle for the expected behaviour of the conversational protocol.</td>
<td>Push mode</td>
<td>Blocking</td>
<td>Centralised</td>
<td>Platform</td>
<td>Consumer, and Third-party</td>
<td>Messages</td>
<td>Reactive</td>
<td>Automated</td>
</tr>
<tr>
<td>Simmonds et al. [53]</td>
<td>Monitor the correctness of the Web service interactions in a BPEL process</td>
<td>The authors used UML 2.0 Sequence Diagrams (SD) as a visual property specification language for expressing safety and liveness properties. SDs are transformed to finite-state automata to check the conformance of execution traces against to a specification of such properties.</td>
<td>Push mode</td>
<td>Non-blocking</td>
<td>Centralised</td>
<td>Platform</td>
<td>Provider</td>
<td>Messages</td>
<td>Reactive</td>
<td>Automated</td>
</tr>
<tr>
<td>Alodib and Bordbar [50]</td>
<td>Detection and diagnosis of failures in BPEL processes</td>
<td>The authors used the Workflow Graph formalism to specify a coverability graph for the monitored BPEL process, which is checked using the coverability graph to check the correctness of the process execution flow.</td>
<td>Push mode</td>
<td>Blocking</td>
<td>Centralised</td>
<td>Subject</td>
<td>Provider</td>
<td>Messages</td>
<td>Reactive</td>
<td>Automated</td>
</tr>
<tr>
<td>Kallel et al. [44]</td>
<td>Monitor whether the composition satisfies various temporal constraints</td>
<td>The authors used the XTUS-Automata formal language to define temporal properties in a specification that is being translated to AO4BPEL aspects, which are used to check if the properties are satisfied.</td>
<td>Simulation mode</td>
<td>Blocking</td>
<td>Centralised</td>
<td>Platform</td>
<td>Provider</td>
<td>Messages</td>
<td>Proactive</td>
<td>Automated</td>
</tr>
<tr>
<td>Momm et al. [46]</td>
<td>Monitoring QoS of BPEL processes</td>
<td>The author adopted a model-driven approach by introducing three models with different level of abstraction in order to monitor process QoS in a platform-independent manner.</td>
<td>Push mode</td>
<td>Non-blocking</td>
<td>Centralised</td>
<td>Platform</td>
<td>Provider</td>
<td>Events</td>
<td>Reactive</td>
<td>Automated</td>
</tr>
<tr>
<td>Authors</td>
<td>Monitoring Service Level Agreements for Web Services</td>
<td>The authors combined event-based monitoring at server-side and perform various measurements at the client-side. The observed information at both sides is combined sides in order to detect SLA violations.</td>
<td>Push mode</td>
<td>Non-blocking</td>
<td>Centralised</td>
<td>Platform</td>
<td>Provider, Consumer, and Third-party</td>
<td>Events</td>
<td>Reactive</td>
<td>Automated</td>
</tr>
<tr>
<td>---------</td>
<td>-----------------------------------------------------</td>
<td>-------------------------------------------------------------------------------------------------</td>
<td>---------</td>
<td>-------------</td>
<td>---------</td>
<td>---------</td>
<td>----------------------------------</td>
<td>--------</td>
<td>----------</td>
<td>-----------</td>
</tr>
<tr>
<td>Baresi and Guinea [73]</td>
<td>Monitor undesirable behaviours in BPEL processes</td>
<td>The authors presented an approach that injects assertions in the BPEL process, such that various checks can be done at run-time</td>
<td>Push mode</td>
<td>Blocking</td>
<td>Centralised</td>
<td>Subject</td>
<td>Provider</td>
<td>Messages and Events</td>
<td>Reactive</td>
<td>Automated</td>
</tr>
<tr>
<td>Halle et al. [57]</td>
<td>Monitor the behaviour of the services against data and control flow constraints</td>
<td>The authors used the Interface grammars formalism for expressing constraints on sequences of messages, which are intercepted during the interaction with of the monitored Web service</td>
<td>Polling mode</td>
<td>Blocking</td>
<td>Centralised</td>
<td>Independent</td>
<td>Provider, Consumer, and Third-party</td>
<td>Messages</td>
<td>Reactive</td>
<td>Automated</td>
</tr>
<tr>
<td>Baresi et al. [126]</td>
<td>Monitoring of services used in BPEL processes in a platform-neutral manner</td>
<td>The authors used three different models for modelling quality attributes, management objectives, and KPIs at different level of abstraction and they use these models to generate event correlation statements for an event processing middleware</td>
<td>Push mode</td>
<td>Non-blocking</td>
<td>Centralised</td>
<td>Independent</td>
<td>Provider</td>
<td>Events</td>
<td>Reactive</td>
<td>Automated</td>
</tr>
<tr>
<td>Dranidis et al. [58]</td>
<td>Determining proactively when to adapt during the interaction with conversational (aka. stateful) services in order to avoid costly recovery</td>
<td>The authors modelled the conversational protocol using the Stream X-Machines formalism and test cases are generated during deployment, such that online testing reveals faults before the actual invocation of the monitored Web service</td>
<td>Pooling mode</td>
<td>Blocking</td>
<td>Centralised</td>
<td>Independent</td>
<td>Provider, Consumer, and Third-party</td>
<td>Messages</td>
<td>Proactive</td>
<td>Automated</td>
</tr>
<tr>
<td>Lorenzoli and Spanoudakis [64, 65]</td>
<td>Monitoring Service Level Agreements for Web Services</td>
<td>The authors used historical monitored information collected by the EVEREST+ monitoring framework to compute the probability that a QoS property will be satisfied at a particular point in time.</td>
<td>Push mode</td>
<td>Non-blocking</td>
<td>Centralised</td>
<td>Platform</td>
<td>Provider</td>
<td>Events</td>
<td>Proactive</td>
<td>Automated</td>
</tr>
</tbody>
</table>
Towards an integration and coordination framework for architecting multi-layered M&A of SBA

<table>
<thead>
<tr>
<th>Authors</th>
<th>Title</th>
<th>Description</th>
<th>Push mode</th>
<th>Non-blocking</th>
<th>Platform</th>
<th>Provider</th>
<th>Messages</th>
<th>Reactive</th>
<th>Automated</th>
</tr>
</thead>
<tbody>
<tr>
<td>Goel et al. [63]</td>
<td>Monitoring Service Level Agreements for Web Services</td>
<td>The authors specified SLAs as formulas using a DSL based on temporal logic of safety and the formulas are translated to automata utilised in the monitors</td>
<td>Push mode</td>
<td>Non-blocking</td>
<td>Centralised</td>
<td>Platform</td>
<td>Provider</td>
<td>Messages</td>
<td>Reactive</td>
</tr>
<tr>
<td>Ivanovic et al. [64]</td>
<td>Monitoring Service Level Agreements for Web Services</td>
<td>The authors presented an approach for prediction of SLA conformance and violation scenarios by deriving constraints, which represent that scenarios</td>
<td>Push mode</td>
<td>Non-blocking</td>
<td>Centralised</td>
<td>Platform</td>
<td>Provider</td>
<td>Events</td>
<td>Proactive</td>
</tr>
</tbody>
</table>
## Classification of adaptation approaches for the SCC layer

<table>
<thead>
<tr>
<th>Work</th>
<th>Goal</th>
<th>Solution</th>
<th>Methodology</th>
<th>Architecture</th>
<th>Decision</th>
</tr>
</thead>
<tbody>
<tr>
<td>Zeng et al. [75]</td>
<td>Optimise QoS of a BPEL process</td>
<td>The authors proposed the design of an abstract workflow to define basic activities, which are fulfilled by Web services selected at run-time according to their quality characteristics. The workflow is specified using Statecharts, which are then transformed to Directed Acyclic Graphs, in order to generate potential execution plans. Finally, Linear Programming techniques are used to select the optimal execution plan.</td>
<td>Goal-based Flow instance</td>
<td>Centralised Platform</td>
<td>Automated Dynamic Reactive</td>
</tr>
<tr>
<td>Charfi and Mezini [83], [89]</td>
<td>Introduce adaptation logic in BPEL engines</td>
<td>The authors presented an extension of BPEL, namely AOBPEL. The extension introduces support for aspect-oriented programming into the standard BPEL language.</td>
<td>Explicit variability Service and Flow instance</td>
<td>Centralised Platform</td>
<td>Automated Static Reactive</td>
</tr>
<tr>
<td>Friese et al. [84]</td>
<td>Recovery of failures in a BPEL process</td>
<td>The authors introduced a middleware that provides a robust execution layer for BPEL processes. The middleware is able to substitute a problematic Web service by looking up services from a P2P service repository. The authors defined an approach to discover functionality equivalent services using interface-based matching and they select the appropriate Web service from the discovered candidates based on several types of formal reasoning (e.g. goal elaboration, abduction, and path transformation)</td>
<td>Action-based Flow instance</td>
<td>Decentralised Platform</td>
<td>Automated Dynamic Reactive</td>
</tr>
<tr>
<td>Siljee et al. [76]</td>
<td>Optimise QoS of a BPEL process</td>
<td>The authors used explicit variability techniques to define variants with different QoS. A component is able to look for better configurations continuously, switch to different configuration when danger for QoS failure appears, or choose a new configuration in case of low QoS.</td>
<td>Explicit variability Flow instance</td>
<td>Centralised Platform</td>
<td>Automated Static Reactive and Proactive</td>
</tr>
<tr>
<td>Spanoudakis et al. [85]</td>
<td>Recovery of failures in a BPEL process</td>
<td>The authors used an approach to discover functionality equivalent services using interface-based matching and they select the appropriate Web service from the discovered candidates based on several types of formal reasoning (e.g. goal elaboration, abduction, and path transformation)</td>
<td>Utility function Service instance</td>
<td>Centralised Platform</td>
<td>Automated Dynamic Reactive</td>
</tr>
<tr>
<td>Verma et al. [77]</td>
<td>Optimise QoS of a service composition</td>
<td>The authors used ontologies to describe declarative constraints concerning the functional and non-functional requirements of</td>
<td>Implicit Flow instance</td>
<td>Centralised Platform</td>
<td>Automated Dynamic Reactive</td>
</tr>
</tbody>
</table>
Towards an integration and coordination framework for architecting multi-layered M&A of SBA

<table>
<thead>
<tr>
<th>Author(s)</th>
<th>Focus Area</th>
<th>Description</th>
<th>Resolution Approach</th>
</tr>
</thead>
<tbody>
<tr>
<td>Benatallah et al. [106]</td>
<td>Address interface and interaction protocol mismatches between Web services</td>
<td>The authors described a set of mismatch patterns, which are used in combination with service composition and a semi-automated toolset to assist developers in generating adapters for addressing interface and interaction protocol incompatibilities</td>
<td>Implicit Service class Centralised Platform Interactive Dynamic Proactive</td>
</tr>
<tr>
<td>Brogi and Popescu [107]</td>
<td>Address interaction protocol mismatches between interacting BPEL processes</td>
<td>The authors presented a method for transforming the interacting BPEL processes into the Yet Another Workflow Language (YAWL). Based on the YAWL model they generate service execution trees that represent execution scenarios and it analyses these scenarios to automatically synthesise a full or partial adapter</td>
<td>Implicit Service class Centralised Platform Interactive Dynamic Proactive</td>
</tr>
<tr>
<td>Dumas et al. [100]</td>
<td>Address interface mismatches between interacting BPEL processes</td>
<td>The authors introduced a declarative approach to service interface adaptation based on an algebra of six operators and a visual language for creating mapping between the incompatible interfaces using the provided operators. Additionally, the authors present an execution engine, which consumes such mapping in order to transform and propagate the intercepted messages, exchanged between the interacting services</td>
<td>Implicit Service class Centralised Platform Interactive Dynamic Proactive</td>
</tr>
<tr>
<td>Chafle et al. [78]</td>
<td>Optimise QoS of a BPEL process</td>
<td>The authors presented an approach for adapting service compositions during deployment and run-time. The approach uses composition templates used to generate service composition instances by selecting concrete Web services. The services are selected according to semantic annotations that describe functional requirements and utility functions that evaluate QoS properties</td>
<td>Utility function Service Instance Centralised Platform Automated Dynamic Reactive</td>
</tr>
<tr>
<td>Colombo et al. [86]</td>
<td>Failure recovery and reconfiguration</td>
<td>The authors introduced a complete execution environment that extends a BPEL engine to support the definition of rules, which are</td>
<td>Implicit Service instance Centralised Platform Automated Static Reactive</td>
</tr>
</tbody>
</table>
Towards an integration and coordination framework for architecting multi-layered M&A of SBA

<table>
<thead>
<tr>
<th>Authors</th>
<th>Description</th>
<th>Approach Type</th>
<th>Platform Type</th>
<th>Action Type</th>
<th>Recovery Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Denaro et al. [101]</td>
<td>Address interaction protocol mismatches between Web services. The authors demonstrated an approach for synthesising Finite State Machines models by observing the interactions between a consumer and a provider. The model is then exploited by the consumer to dynamically configure an adapter, which translates the consumer requests to the interaction protocol of a provided service.</td>
<td>Implicit</td>
<td>Centralised</td>
<td>Platform</td>
<td>Dynamic</td>
</tr>
<tr>
<td>Kongdenfha et al. [102]</td>
<td>Address interaction protocol mismatches between Web services. The authors described an approach that uses aspect-oriented techniques to address incompatibilities between interacting Web services. The authors introduce aspect-based templates, which are instantiated at run-time to handle mismatches.</td>
<td>Action-based</td>
<td>Centralised</td>
<td>Platform</td>
<td>Interactive</td>
</tr>
<tr>
<td>Williams et al. [103]</td>
<td>Address interaction protocol mismatches between Web services. The authors used ontologies to reflect the abstract interaction protocols of the communicating services. The ontology is then used to mediate the interaction of the communicating services by deciding to enact the appropriate invocations at run-time.</td>
<td>Implicit</td>
<td>Centralised</td>
<td>Platform</td>
<td>Automated</td>
</tr>
<tr>
<td>Modafferi et al. [87]</td>
<td>Recovery of failures in a BPEL process. The authors introduced a plug-in for BPEL process engines to incorporate recovery mechanisms. The plug-in is able to perform recovery actions by leveraging annotation incorporated in the process. The supported action are concerned both with the replacement of the constituent services or the structure of the process.</td>
<td>Action-based</td>
<td>Centralised</td>
<td>Subject</td>
<td>Automated</td>
</tr>
<tr>
<td>Ardagna and Pernici et al. [79]</td>
<td>Optimise QoS of a BPEL process. The authors presented an approach concerned with the selection of the most appropriate Web services in a BPEL process, in order to fulfil the best QoS. The expected QoS is comprises the user preferences, the operational context, and the QoS constraints for the process. The approach uses mixed integer linear programming as the underlying mechanism for the service selection.</td>
<td>Utility function</td>
<td>Centralised</td>
<td>Platform</td>
<td>Automated</td>
</tr>
<tr>
<td>Baresi et al. [88]</td>
<td>Recovery of failures in a BPEL process. The authors introduced the Web Service Recovery Language used for expressing adaptation strategies, which can comprise</td>
<td>Action-based</td>
<td>Centralised</td>
<td>Platform</td>
<td>Automated</td>
</tr>
</tbody>
</table>
Towards an integration and coordination framework for architecting multi-layered M&A of SBA

<table>
<thead>
<tr>
<th>Authors</th>
<th>Description</th>
<th>Approach</th>
<th>Platform</th>
<th>Automation</th>
<th>Reactivity</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bianculli et al. [111]</td>
<td>Prevent failures from external services in a BPEL process</td>
<td>Implicit</td>
<td>Service instance</td>
<td>Centralised</td>
<td>Platform</td>
</tr>
<tr>
<td>Console et al. [90]</td>
<td>Recovery of failures in a service composition</td>
<td>Action-based</td>
<td>Service and Flow instance</td>
<td>Decentralised</td>
<td>Platform</td>
</tr>
<tr>
<td>Motahari et al. [104]</td>
<td>Address interface and protocol mismatches between Web services</td>
<td>Implicit</td>
<td>Service instance</td>
<td>Centralised</td>
<td>Platform</td>
</tr>
<tr>
<td>Narendra et al. [113]</td>
<td>Recovery of failures in a service composition</td>
<td>Explicit variability</td>
<td>Service instance</td>
<td>Decentralised</td>
<td>Platform</td>
</tr>
<tr>
<td>Cavallaro and Di Nitto [108]</td>
<td>Address interface and protocol mismatches between Web services</td>
<td>Implicit</td>
<td>Service instance</td>
<td>Centralised</td>
<td>Platform</td>
</tr>
<tr>
<td>Authors</td>
<td>Title</td>
<td>Description</td>
<td>Utility function</td>
<td>Service and Flow instance</td>
<td>Coordination</td>
</tr>
<tr>
<td>--------------------</td>
<td>------------------------------------------------------------------------</td>
<td>-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
<td>------------------</td>
<td>----------------------------</td>
<td>--------------</td>
</tr>
<tr>
<td>Moser et al. [45]</td>
<td>Optimise QoS of a BPEL process and address interface mismatches</td>
<td>The authors presented a middleware for monitoring and adaptation of BPEL processes using aspect-oriented programming. Various selectors are used to bind to the appropriate Web service. In case that there are interface level mismatches transformers are used to address these.</td>
<td>Implicit</td>
<td>Service instance</td>
<td>Centralised</td>
</tr>
<tr>
<td>Fredj et al. [91]</td>
<td>Recovery of failures in a BPEL process</td>
<td>The authors presented a middleware that is able in the event of a failure to substitute Web services in a BPEL process. The middleware has been based on Semantic Web technologies, since it uses a SA-WSDL to describe services. A semantic registry is used for discovering suitable services to replace the failed services.</td>
<td>Implicit</td>
<td>Service instance</td>
<td>Centralised</td>
</tr>
<tr>
<td>Cavallaro et al. [105]</td>
<td>Address interaction protocol mismatches between Web services</td>
<td>The authors introduced an approach that uses Label Transition Systems (LTS) to represent the interaction protocol of a Web service. Based on the LTS a mapping script is generated that is used by a generic adapter component to perform translation of the interaction protocol at run-time.</td>
<td>Implicit</td>
<td>Service instance</td>
<td>Centralised</td>
</tr>
<tr>
<td>Charfi et al. [92]</td>
<td>Introduce adaptation logic in BPEL engines</td>
<td>The authors presented an architecture based on aspect-oriented techniques for introducing adaptation logic in the form of aspects. Aspects can be executed before or after a specific activity in the process and therefore enact proactive or reactive adaptation actions.</td>
<td>Explicit variability</td>
<td>Service and Flow instance</td>
<td>Centralised</td>
</tr>
<tr>
<td>Kareliotis et al. [114]</td>
<td>Optimise QoS of a BPEL process and recovery from failures</td>
<td>The authors presented a framework that uses BPEL process scenarios, which are fulfilled with concrete Web services, based on the QoS characteristics of the services. In a case of a fault during the invocation of a service, the problematic service is substituted with another equivalent services discovered through a registry.</td>
<td>Goal-based</td>
<td>Service instance</td>
<td>Centralised</td>
</tr>
<tr>
<td>Kongdenfha et al. [109]</td>
<td>Address interface and protocol mismatches between Web services</td>
<td>The authors introduced an approach that uses aspect-oriented techniques for assisting in the generation of service adapters, based on predefined mismatch patterns.</td>
<td>Implicit</td>
<td>Service instance</td>
<td>Centralised</td>
</tr>
<tr>
<td>Authors</td>
<td>Recovery of failures in a BPEL process</td>
<td>The authors presented a refined framework that uses aspect-oriented techniques to foster separation of concerns between the process and the adaptation logic</td>
<td>Action-based</td>
<td>Service instance</td>
<td>Centralised</td>
</tr>
<tr>
<td>----------------------</td>
<td>---------------------------------------</td>
<td>------------------------------------------------------------------------------------------------------------------------------------------------</td>
<td>--------------</td>
<td>-----------------</td>
<td>------------</td>
</tr>
<tr>
<td>Baresi and Guinea et al. [73]</td>
<td>Address protocol mismatches between Web services</td>
<td>The authors introduced an approach to automated generation of interaction protocol adapters. The approach is able to infer the data type dependencies between operations in the service interface and based on the inferred dependencies to create an automation, which is used to test the conformance of the new service and to generate a mapping adapter as well</td>
<td>Implicit</td>
<td>Service instance</td>
<td>Centralised</td>
</tr>
<tr>
<td>Cavallaro et al. [110]</td>
<td>Recovery of failures in a BPEL process</td>
<td>The authors introduced an approach that uses a process model to generate repair plans. The model defines information about the process, such as structure, data dependencies, process results, and available repair actions</td>
<td>Action-based</td>
<td>Service and Flow instance</td>
<td>Centralised</td>
</tr>
<tr>
<td>Friedrich et al. [93]</td>
<td>Recovery of failures in a BPEL process</td>
<td>The authors described a framework that is able to detect undesired delegation behaviour in SOA systems, which comprise both software-based service and human-provided services. The framework is able to detect delegation patterns from event logs, in order to modify the delegation behaviour based on predefined rules</td>
<td>Action-based</td>
<td>Flow instance</td>
<td>Centralised</td>
</tr>
<tr>
<td>Psaier et al. [94]</td>
<td>Recover from failures in delegation behaviour of services</td>
<td>The authors presented a framework that is able to detect undesired delegation behaviour in SOA systems, which comprise both software-based service and human-provided services. The framework is able to detect delegation patterns from event logs, in order to modify the delegation behaviour based on predefined rules</td>
<td>Action-based</td>
<td>Flow instance</td>
<td>Centralised</td>
</tr>
<tr>
<td>Hermosillo et al. [95]</td>
<td>Optimise QoS of a BPEL process and recovery from QoS failures</td>
<td>The authors presented a framework that uses complex event processing for detecting situations requiring QoS optimisation of a process instance. The framework uses aspect-oriented techniques to modify the process and restore the desired QoS</td>
<td>Action-based</td>
<td>Flow instance</td>
<td>Centralised</td>
</tr>
<tr>
<td>Staikopoulos et al. [96]</td>
<td>Recovery of failures in a BPEL process</td>
<td>The authors presented a model-driven framework to support modification of service and process models. Additionally, The framework uses Semantic Web Technologies to reason about the parameterization of service templates, in order to generate and deploy an adapted service</td>
<td>Implicit</td>
<td>Service and Flow class</td>
<td>Centralised</td>
</tr>
<tr>
<td>Aschoff and Zisman [81]</td>
<td>Prevent failures from external services in a BPEL process</td>
<td>The authors presented a framework that uses function approximation and failure spatial correlation techniques to predict QoS faults. In the event of a predicted fault, services are started to prevent the service from failing</td>
<td>Utility function</td>
<td>Service instance</td>
<td>Centralised</td>
</tr>
</tbody>
</table>
Towards an integration and coordination framework for architecting multi-layered M&A of SBA

<table>
<thead>
<tr>
<th>Service Composition</th>
<th>Replaced According to QoS Constraints</th>
<th>Implicit</th>
<th>Service Instance</th>
<th>Centralised</th>
<th>Platform</th>
<th>Automated</th>
<th>Static</th>
<th>Reactive</th>
</tr>
</thead>
<tbody>
<tr>
<td>Taher et al. [112]</td>
<td>Address interface and protocol mismatches between Web services</td>
<td>The authors introduced an approach that uses complex event processing and event patterns to modify the request messages send to a service</td>
<td>Utility function</td>
<td>Service instance</td>
<td>Centralised</td>
<td>Platform</td>
<td>Automated</td>
<td>Static</td>
</tr>
</tbody>
</table>
| Pernici and Siadat et al. [97] | QoS optimisation and recovery of failures in a service composition | The authors introduced an approach to the selection of adaptation actions, based on a fuzzy reasoning mechanism about the QoS of a particular service | }