Grant Agreement N° 215483

Title:	*State of the Art Report, Gap Analysis of Knowledge on Principles, Techniques and Methodologies for Monitoring and Adaptation of SBAs*
Authors:	*Salima Benbernou (UCBL), Luca Cavallaro (Polimi) Mohand Said Hacid (UCBL), Raman Kazhamiakin (FBK), Gabor Kecskemeti (SZTAKI), Jean-Louis Poizat (INRIA), Fabrizio Silvestri (CNR), Maike Uhlig (UniDue), Branimir Wetzstein (USTUTT)*
Editor:	*Salima Benbernou (UCBL)*
Reviewers:	*Manuel Carro (UPM)*
	Barbara Pernici (Polimi)
Identifier:	*Deliverable # PO-JRA-1.2.1*
Type:	*Deliverable*
Version:	*1.0*
Date:	*10 July 2008*
Status:	*Final*
Class:	*External*

Management Summary

The deliverable presents the state-of-the-art principles, techniques, and methodologies for the monitoring and adaptation of Service-Based Applications. The report also includes an overview of the different kinds of adaptation and monitoring principles and mechanisms, provides a classification of the existing approaches from the literature, and makes a comparison with the state-of-the-art approaches in other related areas of information systems.

Copyright © 2008 by the S-Cube consortium – All rights reserved.

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

Members of the S-Cube consortium:

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

Published S-Cube documents

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

The S-Cube Deliverable Series

Vision and Objectives of S-Cube

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

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

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

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

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

Contents

Chapter 1

Introduction

The advances in modern technology development and future technology changes dictate new challenges and requirements to the engineering and provision of Service Based Applications. These applications should become drastically more flexible; they should be able to operate and evolve in highly dynamic environments, being able to adequately react to various changes in these environments. In these settings, adaptability becomes a key feature of SBA as it provides a way for applications to continuously change itself in order to satisfy new requirements and demands dictated by the environment.

The range of the possible changes that require application adaptation includes the changes in the infrastructural layer of the application, where the quality of service changes or service becomes unavailable; changes of the (hybrid) application context and location, where SBA should be able to replace one service by another possibly having other properties and parameters; changes of the user types, preferences, and constraints that require application customization and personalization as a means to adapt the application behavior to a particular user; changes in the functionalities provided by the component services that requires modifying the way in which services are composed and coordinated; changes in the ways the SBA is being used and managed by its consumers, which should lead to changes in the application requirements and the business processes implementing them. The ability of the application to adapt strongly relies on the presence of monitoring mechanisms and facilities. With monitoring we mean the methods and mechanisms that allow one to identify, detect, and even foresee the critical events and situations that occur in the SBA environment and that require the latter to change its configuration, behavior, presentation, etc. As these events occur at different functional layers of SBAs, namely the business process layer, the service composition layer, and the service infrastructure layer, there is a need for the cross-cutting monitoring techniques and methodologies for SBAs.

The life-cycle of adaptable SBAs[1] with respect to the adaptation and monitoring tasks consists of the following phases:

- identification and agreement on critical requirements and properties of the underlying SBA that should be continuously observed. This may include the relevant behavioral requirements, non-functional constraints, Key Performance Indicators (KPI), contextual properties, etc.
- design of an observation model of the applications, i.e., specifying relevant monitoring information using the corresponding notations and toolkits (design for monitoring).
- deployment of monitors and sensors together with the underlying application.
- monitoring the relevant execution properties of the application instances. Such monitoring may be performed at run-time, in parallel with the application execution, or post-mortem, i.e., after the execution.

[1]as defined and presented in Deliverable PO-JRA-1.1.1 “State of the art report on software engineering design knowledge and Survey of HCI and contextual Knowledge”

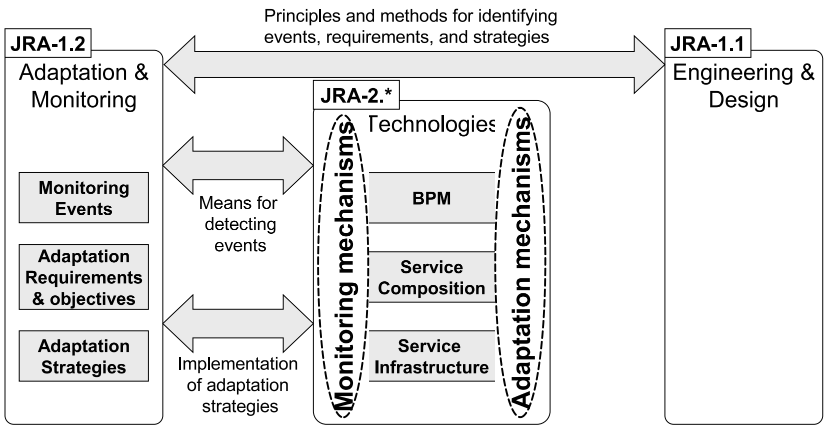

Figure 1.1: Adaptation and Monitoring Research

- identification of critical events and situations in accordance with the monitoring specification. These events may correspond to the contextual changes, violation of requirements and service-level-agreements (SLA), etc.

- identification of adaptation needs and requirements, that is, whether the application should be changed (adapted), and what part of the application is the subject to change.

- definition and selection of the corresponding adaptation strategy that aims to achieve the adaptation need given the current operational state. Such strategy may be defined by the system integrator at design-time or automatically at run-time based on predefined policies and current situation.

- enactment of the adaptation strategy that, upon successful implementation, leads to a new version of the system. Such modification may change only a currently executed application instance or the whole application model.

- re-activation of the modified application in the current operational context. As a result, the application may be completely re-deployed, or the execution may continue from the previous state according to the new model.

Seen from a global perspective, the research activities on monitoring and adaptation may be represented as in Fig 1.1. It relies on a number of key concepts, namely *Monitoring Events*, *Monitoring Mechanisms*, *Adaptation Requirements and Objectives*, *Adaptation Strategies*, and *Adaptation Mechanisms*. These concepts identify the overall vision on the development and provision of adaptable and flexible service-based applications (SBA).

- *Monitoring Mechanisms* provide the actual realization of the methods for continuous observing and detecting relevant Monitoring Events; they identify the "How?" dimension of the monitoring process.

- *Monitoring Events* deliver the information about the application evolution and changes in the environment. These events define the "What?" dimension of the monitoring process: they are used to indicate whether the SBA is executed and evolves in a normal mode, whether there are some deviations or even violations of the desired or expected functionality.

- The events identified during the monitoring process carry the information about potential changes that the system and / or the underlying platform should perform in order to adapt to a new situation. The relation between the monitoring events and the changes of SBA are defined by the

Adaptation Requirements and Objectives. These requirements identify the “What?” dimension of the adaptation process as they describe the goals and needs the adapted SBA should satisfy.

- The *Adaptation Strategies* define the possible ways to achieve these requirements and needs given the adaptation mechanisms made available (independently) in different functional layers of SBA. The adaptation strategies should guarantee coherent and integrated cross-layer management of these adaptation techniques.

- The *Adaptation Mechanisms* provided at various functional SBA layers offer a means to actually implement the adaptation strategies in integrated and coordinated manner.

The research on the problem of monitoring and adaptation of SBA is not performed in isolation; it is strongly related and relies upon the activities performed in other research areas. The research on engineering and design of SBAs provides approaches, principles, methods and notations for identifying and describing relevant changes in the application and the corresponding events (design for monitoring), for defining and representing adaptation requirements and objectives (design for adaptation). State of the art in design and engineering SBAs, and, in particular, adaptable SBAs is presented in Deliverable PO-JRA-1.1.1. The research on the realization mechanisms for SBAs will provide various tools that will be used to implement monitoring and adaptation activities at different functional layers. Deliverables PO-JRA-2.1.1, PO-JRA-2.2.1, and PO-JRA-2.3.1 present state of the art in realization mechanisms for business process management, for service composition and coordination, and for service infrastructure respectively.

These techniques will be used for the definition of the integrated, cross-layer monitoring and adaptation approaches. On the other side, the continuous evaluation and validation of the obtained research results will allow us to identify potential problems and gaps in the implementation of cross-layer approaches, and therefore to devise new requirements and challenges that should lead to the development of the new engineering principles and methods, and new realization mechanisms in the corresponding activities. In this survey we will study and review the relevant monitoring and adaptation approaches. We will provide a comprehensive classification of the corresponding concepts, principles, techniques, and methodologies; we will identify the overlaps between various research activities and reveal the gaps and problems that the research community should address in this area. The structure of the survey is organized as follows. Chapter 2 will present the classification and the state-of-the-art works in monitoring of service-based applications. Chapter 3 will provide a view on the adaptation problem and survey the relevant work in this area.

Chapter 2

Monitoring

2.1 Preliminaries

The term “monitoring” has been widely used in many disciplines and in particular in service-oriented design and engineering. Depending on a particular purpose of the designed system, on the role the monitoring process plays in the system life-cycle, and the kind of information being collected, the definition of the monitoring problem has different interpretations. In a broad sense, monitoring may be defined as *a process of collecting and reporting relevant information about the execution and evolution of service-based applications*. This general definition becomes more concrete and clear when the monitoring goals are considered. Monitoring may be used to discover problems in the application execution. In this case monitoring may be defined as a problem of observing the behavior of a system and determining if it is consistent with a given specification [1]. Monitoring may support the run-time optimization of a system, when the monitored quality of service properties of the involved services is used to drive resource allocations. It can also be used to detect relevant contextual changes and events in order to support context-driven run-time adaptation of SBA; to collect information relevant for the application evolution, such as, histories of adaptations, decisions made in different executions, in order to support the evolution and governance of SBA; perform business activity monitoring, etc.

Another relevant issue for the description of monitoring is what kind of data is being observed and reported, that is, the *monitored information*. In these regards one can consider monitoring of functional or non-functional properties, instance or class-based events, external or internal aspects, etc.

Finally, *monitoring approaches* used to address one or another monitoring problem, are implemented with the help of various *monitoring techniques*. These approaches and techniques vary on the basis of such aspects as the way the monitoring information is specified, degree of integration with the application, timing of collecting information and information sources.

A generic monitoring framework may be defined as follows. Following the definition proposed in [2], a *monitor* is a program that observes the execution of another program. The monitor is being defined and implemented on the basis of the *monitoring specification* that describes the properties and events to be observed. The actual information is being perceived and delivered to the monitor by a set of *sensors*. This information is being processed by the monitor in order to decide whether this information corresponds to the monitored event or property (e.g., certain quality-of-service (QoS) parameter value exceeded expected bound, a correctness requirement, is violated, or a relevant contextual change has occurred). In this case the monitor reports the event to the associated *event handler*, which performs some associated action or merely writes the event to the corresponding application log.

In the following section we will give the more detailed classification of the monitoring problem and review the state of the art in the research on monitoring of service-based applications.

2.2 Classification of Monitoring Aspects

In the classification of the existing approaches to monitoring service-based applications, we will focus on the following aspects of the monitoring problem: (i) the *monitoring task*, i.e., what is the purpose of the monitoring activity within the life-cycle of the SBA and what kind of information is being observed; (ii) the *specification of the monitoring information*, that is, how the monitoring problem is modeled and how the monitoring instructions are specified; (iii) *implementation methods and techniques* exploited by the approach.

We remark that this classification is based on and extends the taxonomies suggested for the classification of fault monitoring tools and approaches for software systems [1] and service-oriented architectures [3].

2.2.1 Monitoring Task

As we already discussed above, the monitoring approaches may be classified according to the monitoring task addressed by the approach. The monitoring task aims at answering two key questions, namely “why monitor?” and “what to monitor?”.

Monitoring goals and objectives

For what concerns the goal of the monitoring activity, the approaches may be tailored to the following purposes identified in the literature.

- Fault monitoring addresses the problem of *run-time analysis* of the system correctness, i.e., checking the system execution and its parameters against certain specifications that distinguish “correct” and “failing” situations. This amounts, for example, to the run-time verification of certain behavioral or non-functional properties expected from the system, run-time conformance analysis, when the actual system traces is compared with the system specification, analysis of service-level agreements, requirements analysis, etc.

- Monitoring is instrumental to perform the *diagnosis and recovery (repair)* of the identified faults, i.e., to identify the cause of the problem and drive the actions necessary to return the system to the “normal” execution.

- Monitoring of quality characteristics of the involved services and components is often used to instantiate and guide the *optimization* activities regarding resource allocation in grid computing.

- In a similar way, monitoring of various contextual properties may be required in order to support *dynamic adaptation* of the application to the relevant change in its environment. Depending on the adopted definition of context, this may relate to the monitoring of service registries in order to discover new useful services or changes in the characteristics of the existing ones; monitoring of the run-time environment of the application, such as an underlying device or infrastructure; monitoring of the user-related characteristics (e.g., preferences or accessibility parameters); monitoring current location and time, etc.

- *Business Activity Monitoring* (BAM) provides near real-time monitoring of business activities, measurement of key performance indicators (KPIs), their presentation in dashboards, and automatic and proactive notification in case of deviations. A “business activity” thereby can be implemented as a service orchestration in a BPMS, or, more general, as part of a business process consisting of a series of activities implemented across workflow systems, enterprise resource planning (ERP) systems and legacy applications, possibly across organizational boundaries. The goal of BAM is to provide timely information about the status and the performance of business processes, and to proactively alert users if business rules are violated or KPI values deviate from target values.

- Monitoring activities may be also used to support the long-term adaptation of the service-based system, also referred to as *evolution*. In this case the evolutionary aspect speaks about monitoring of different data that represent certain trend in the execution of the process, i.e., the decisions taken, the relevant values of the process-related data, and the history of this information.

Relevant to this issue, is the concept of the stakeholder, which is interested in the monitoring activity [3]. Run-time analysis of various correctness properties over involved service amounts to the viewpoint of the service requester: the service should deliver the expected property. The analysis of SLA conformance may correspond to the third-party viewpoint, where the expectations and commitments of both the requester and provider are considered. Finally, the issues related to the application adaptations and optimizations correspond to the perspective of the service provider, who is willing to improve the system and to align it with the new situation and context.

Monitoring objectives may also be characterized by the aspect of the SBA model addressed by the activity. Many run-time approaches deal with the behavioral properties of the application, such as conformance to the abstract interaction protocol, various safety properties or complex temporal constraints, and in certain cases quality of service parameters. Other approaches focus on the contextual information or business activity and management characteristics, such as business metrics and KPIs, security-related aspects, etc.

Monitoring Information

For what concerns the definition of the monitoring information, the following key issues may be considered: the kind of properties it represents (i.e., functional or non-functional, instance-based or aggregated), the occurrence and distribution of the information (internal or external, centralized or distributed, functional SBA layer).

Functional properties refer to the definition of service operations, their signatures and technical parameters (e.g., binding information), as well as to the behavioral specifications of the service-based applications, such as service composition models. The monitored properties, therefore, often reflect the functional requirements on the underlying system. These properties may refer to the run-time verification of functional and behavioral requirements, such as pre- and post-conditions on service calls, signature conformance, properties over the transmitted data, etc. Monitoring *non-functional* properties, on the other side, aims at measuring certain characteristics of the system [3]. The relevant non-functional properties that are subject of this quantitative analysis are availability, accessibility, performance, and reliability. The monitored properties may also be divided into *instance-based*, i.e., related to the current instance of the application (e.g., current instance of a composition), or *aggregated* (historical), where the property represents the accumulative information over the relevant property (e.g., average duration of the business process, or weighted number of times when the certain event occurred).

The location and distribution of the monitored information may be characterized in several dimensions. First, the events may be *internal* to the application (e.g., internal faults, states changes), thus characterizing the application execution, or *external* (e.g., contextual), thus characterizing the application environment. Second, the information may depend on the collaboration paradigm adopted by the composition model of the application [3]. In case of orchestration-based compositions, the application, and therefore the monitor, is tailored to the *centralized* execution environment. In case of choreography-based applications, the components of the system act independently performing their own part of the business process. The information about the state of the application and the performed task is, therefore, *distributed*, thus requiring an extra effort for defining and performing overall monitoring activities. Finally, the monitored properties may refer to the information that corresponds to different functional layers of the service-based applications. Business activity monitoring addresses the information defined at the *business process management layer*. Monitoring of the QoS properties usually targets the *infrastructural layer*, where the services and underlying middleware protocols are defined. The monitoring of behavioral properties and interactions corresponds to the *composition layer*, where the service composition and message exchanges are specified.

2.2.2 Specification of Monitoring Information

From the design perspective, an important aspect of the classification of the monitoring approaches refers to the ways the monitoring properties are specified. Even when dealing with the same kind of monitored information, the approaches may differ in the features of the specification language (e.g., formalism used and its expressiveness, modeling methodology), level of abstraction, interleaving with the application specification, etc.

An important aspect of the specification language is characterized by the set of *monitoring primitives*, on top of which the properties are specified. Depending on the kind of monitored data, these primitives may refer to various events (e.g., reception/emission of a message, faults, management operations), state/configuration predicates (execution point of the composition specification, values of the relevant parameters or variables), current configuration parameters (QoS metrics of the involved services), and so on. These primitives may be simple (such as, message reception) or structured (aggregated events, global configuration composition configuration).

Related to this is the *abstraction level* of the monitoring specification. Low-level specifications allow for monitoring very specific information related to a particular implementation language. High-level languages may be based on the business-level concepts and events, thus abstracting from the implementation-specific information and details.

The languages rely on different *formalisms* and *modeling notations*. Logic-based notations allow for specifying a property as a logical formula of a special form. Depending on the formalism and its expressiveness, these formulas may characterize only a state of a system at certain moment of time (e.g., in approaches that monitor pre- or post-conditions of the service invocations) or express properties over execution trace or its fragment (e.g., the approaches relying on temporal logics).

Another important aspect of the monitoring specification is how much this specification is *interleaved with the application code*. In many approaches the monitoring problem is described separately from the application specification in order to ensure separation of concerns in the development process. In some approaches, however, the information related to the monitoring is built into the application code. This is done, in particular, in certain adaptation approaches, where the monitored information is provided together with the event handler specification, so that the event detected while performing the corresponding application code immediately triggers the execution of the event handler.

2.2.3 Methodologies and Techniques for Monitoring Implementation

From the implementation perspective, monitoring approaches may be classified according to the features of the techniques and architectures exploited for the realization of the monitoring activity. In this report we do not aim to describe different implementation techniques and specific monitoring architectures in details, due to a very broad scope of the problems and solutions addressed in different SBA-related areas and the corresponding approaches. Instead, we will focus only on a selected set of aspects that characterize one or another approach both from the architectural and from the implementation points of view. These aspects include methodology used to observe the relevant information, degree of invasiveness of the monitors and monitoring architecture with respect to the application and the platform, timing of information gathering, sources of the information and their placement, and adopted techniques specific to the presented approach.

There exist a number of the methodologies used within the monitoring approaches. In *simulation-based* approaches the monitoring property is automatically converted into a special program (automaton) that is being executed (simulated) in parallel with the monitored application. The state of this program is updated upon reception of a certain event. When the program reaches certain state that corresponds to the violation/satisfaction of the monitored property, this fact is reported and the monitor terminates. In *event-based* approaches the monitoring properties are explicitly associated to certain events (e.g., assertions, pre-/post-condition checks), and when the event is fired, the corresponding property is evaluated or metric variable is updated. *Situation-based* approaches, on the contrary, periodically access the information sources and perform the evaluation on the basis of the currently available information.

The *degree of invasiveness* characterizes the monitoring approaches in two directions. First, it depends on the way the monitoring framework is integrated with the application architecture. In some approaches, monitoring facilities and sensors become (usually at deployment time) an integrated part of the application, and therefore abstract from the underlying infrastructure and platform. Other approaches, instead, make use of the platform and middleware APIs in order to observe relevant information, without affecting the realization of the application. Second, it depends on whether the monitors are executed *synchronously* with the application code, blocking it until the property evaluation is done, or *asynchronously* in a independent parallel thread, or even on another machine. The example of the first approach is the assertion checking, while the example of the second approach is contextual information gathering. The first approaches permits more operative reaction to the critical events, blocking the application until the handling is done. The second approach allows for better performance and scalability, since the execution of the application is independent from monitoring.

For what concerns the timing of information gathering, one can distinguish the approaches, where the monitoring information is updated *immediately* when the relevant events are triggered, and the *post-mortem* approaches, where the information is gathered after the events occur. This is the case, for instance, of many process mining approaches, where the monitoring activities are performed on the basis of event logs. It is possible also to consider the proactive monitoring, where special techniques are used to *predict* the future events, such as failures on the basis of the current situation and the historical information, i.e., on the basis of a certain “trend”.

The classification of the monitoring techniques may consider also the sources of information used to extract data and relevant events, i.e., the placement of the corresponding sensors. While application logs, context sensors, message queues, process containers, and the corresponding data storages are the typical sources of information, there exist approaches defining ad-hoc information repositories. For example in [4] special reputation repositories are introduced and dynamically updated. The applications monitor the reputation values in order to suggest new service candidates with better reputation.

Finally, an important aspect for the implementation of the monitoring approaches, deals with the particular techniques adopted within the approach. A remarkable candidate is, for example, process mining, which is widely used for extracting from the process logs the patterns that describe the actual process in comparison to the process specification. Another important technique for implementing monitors is the aspect-oriented approach that allows for embedding the monitoring code without affecting the program and in many cases the platform code. Automated planning techniques are exploited in order to extract the monitor programs from the behavioral models of the involved services and complex property specification [5], taking into account non-determinism and partial observability of the composition behavior.

2.3 Approaches to Monitoring of Service-Based Systems

In this section we survey the state-of-the-art works in monitoring of service-based applications. We group the approaches according to the main characteristics of the research and application areas addressed in the approaches. Section 2.3.1 presents the approaches and solutions towards monitoring different types of requirements and policies for Web services and service compositions. Section 2.3.2 presents the works in the area of business activity monitoring, where the focus is on the run-time analysis of business process characteristics and metrics. Section 2.3.3 also describes the process monitoring works, targeting, however, post-mortem monitoring, based on the analysis of various process execution logs. Finally, Section 2.3.4 presents the approaches and tools that aim to monitor grid services and the corresponding grid architectures and infrastructures.

2.3.1 Monitoring Web Services and Service Compositions

In Web services and service compositions, wide range of run-time monitoring approaches address the problem of checking whether certain predefined properties are satisfied when the system is executed. Such properties may have different forms and target different monitoring objectives. Typical monitoring scenarios include run-time verification of behavioral requirements (e.g., assertions, pre-/post-conditions, temporal requirements), conformance of the actual service execution to some predefined specifications (contracts, SLAs), correspondence to some expected characteristics (QoS bounds), etc. The results of the monitoring, i.e., identified deviations of the expected model of executions, may be used not only for signaling the anomalous behavior of the system, but also to activate some additional actions, which target the recovery to a normal execution (in case of identified requirements violation) or optimization of the system (e.g., in case of deviations of the service quality properties).

Assumption-based monitoring of service compositions [6, 5]

In [6, 5] the approach to monitoring of service compositions implemented in BPEL is presented. The main focus of the approach is the run-time checking of the assumptions under which the component services are supposed to participate to the composition, and the conditions that the composition is expected to satisfy. Moreover, the proposed approach aims at checking not only whether the property holds or not, but also at collecting statistical and timing information over such properties, as well as the aggregated historical information about such properties over all the instances of the given business process. In order to address the problem, the authors propose the specification framework and run-time monitoring environment that extends the standard BPEL engine with the monitoring and auditing capabilities. The devised architecture clearly separates the business logic of a web service from its monitoring functionality.

The specification framework relies on an expressive monitoring language, namely Run-Time Monitoring specification Language (RTML), that allows for expressing behavioral properties of service composition instances (e.g., payment should not start before acknowledgement is received), together with the timing (e.g., duration of payment procedure) and statistical information (e.g., number of retries on negative acknowledgment), and of composition classes (e.g., total number of violations or average number of retries). Apart from explicitly specified assumptions it is possible also to check whether the protocol associated to partner interactions is satisfied at run-time. Based on the monitoring specification and the composition model, the corresponding monitor program (in java) is generated using specific automata-theoretic techniques [5].

At run-time the monitor programs, deployed together with the composed services, run in parallel to them, observe their behavior by intercepting the input/output messages that are received / sent by the processes, and evolve on the basis of such event. When the state of the monitor that corresponds to a particular evaluation of the property is reached, the monitor signals misbehaviors or, more in general, information of interest. The run-time environment is implemented as an extension to a BPEL engine. The information is collected from the corresponding message queues and from the process manager in order to observe creation/termination of process instances. The results of the monitoring are presented through a special monitoring console.

The authors also demonstrate the applicability of the presented approach on a set of sophisticated examples, and perform the experimental evaluation of the performance of the monitor extraction and execution.

Monitoring conversational Web services [7]

In [7] the authors propose an approach to check at run-time that the actual behavior of a service conforms to the expected behavior of that service. Conversational services, whose behavior depends on the local state resulting from client-service interaction, raise new challenges both to specification languages and to the way the actual behavior may be checked against the specification. Indeed, the behavioral specification should formally describe what the monitored service ought to do on behalf of its client, and includes not only the syntactic aspects but also the business logic of the service.

This paper explores a specification style that is based on algebraic representation of conversational web services. In this approach the service operations are classified into constructors (that create new data objects), observers (retrieve information on the data) and transformers. The algebraic specification allows not only to specify pre- and post-conditions on the service operations, but also the conditions and axioms that give the semantic to the operations and make the specification state-based. The evaluation of the specification at run-time may be performed by symbolic execution (simulation). The authors present a term-rewriting schema that is needed for this execution.

The architecture of monitoring approach adopts an aspect oriented programming approach. Aspects are dynamically attached to BPEL engine to monitor service composition. AspectJ language is used to add monitoring facilities to ActiveBPEL. The solution in the paper extends the standard implementation of the ActiveBPEL engine with three main additional components: interceptor to intercept the execution of a process within the engine (using aspect oriented programming) in order to pass to the monitor the relevant information (i.e., about external interactions); a specification registry that contains the specifications, against which services are checked for conformance; and a monitor, which is the actual conformance checker. The monitor component is a wrapper for the evaluation of algebraic specifications. It keeps a machine-readable description of the state of the process instances and updates this description when the new information from the interceptor is received. The symbolic state is then evaluated and the result, representing actual information regarding the state of the conversation is compared against the expected one. In case of mismatch, certain specific action may be performed (e.g., logging).

Smart monitors for composed services [8]

In [8] the authors proposed an approach towards monitoring of service compositions specified as BPEL processes against contracts expressed as assertions on service. In particular, the assertion defines the pre- and post-conditions on service invocations, and characterizes functional expectations of the correct service behavior.

The authors propose the process annotations as a way of specifying the contract assertions. These annotations are then automatically translated into the corresponding code extensions that interact with the dedicated monitor components in order to perform the assertion checking. In their proposal these monitors are web services themselves, which receive the instructions from the transformed process, perform the analysis and return the result to the executed process.

In order to implement the monitoring functionality, the authors present two complementary approaches. The first one relies on the C# object-oriented language for specifying and implementing monitor specifications. In this approach, the assertions are expressed completely in this language, thus providing very expressive means for the specification. The corresponding monitor is then automatically generated in this language and deployed as a Web service. The approach is very flexible and expressive; however, it requires the process designer to work at a very low programming level.

The other approach relies on the use of CLIX, a rich and expressive XML-based assertion language, based on first-order logic, which supports, for instance, quantifiers. The implementation of the monitor in this case relies on wrapping the corresponding assertion processor as a Web service. The approach is less flexible, but benefits from high-level and declarative assertion specification language.

Dynamo [9, 10, 11]

In these works the authors extend and elaborate the ideas presented in previous approach ([8]). The main challenges for the new approach are:

- to enable “separation of concerns” in the design of monitoring specification so that new monitoring rules are defined separately from the process itself;
- to come up with a well-defined and expressive language for the monitoring specifications;
- to provide means for monitoring non-functional (QoS) properties of the services;
- to support the collection of the information from external sources;
- to provide means for managing monitoring process;
- to devise and implement the corresponding architecture.

The authors propose an extended notation for specifying monitoring rules. Monitoring rules are defined explicitly and externally in a well-defined structured file. This separation allows different sets of rules to be associated with same process. The rules define the location, where the evaluation should be performed, and its type. For precondition and postcondition, location indicates the BPEL invoke activity, to which the rule is applied; for an invariant it indicates the BPEL scope; and in case of assertion it indicates any point of WS-BPEL process before the assertion should hold.

The monitoring expression is defined in WSCoL [9], the notation inspired by Java Modeling Language, which enables specifying expressions over the process variables and supports set of built-in functions, logical and mathematical operators, and quantification. Furthermore, the language is extended with the facilities to obtain data from external information sources, if they are defined as Web services. It provides the necessary constructs to define both functional and non-functional constraints and properties.

Besides, the monitoring rules provide parameters to govern the degree of monitoring at run-time: such as validity time frame, priority, set of certified providers for which monitoring may be omitted.

The monitoring rules are deployed together with the process through a weaving procedure, i.e., parsing monitoring rules and adding specific WS-BPEL activities to process in order to achieve dynamic monitoring. At run-time the modified process interacts with the proxy service, namely rule manager, which is responsible for processing the monitoring management instructions, processing monitor configuration, obtaining information from external data sources, evaluating monitoring expressions, and interacting with the actual services (instead of the original process). If some constraints are not met, monitoring manager is responsible to inform BPEL process.

In [11] the authors extend this work for what concerns the kind of properties the approach can monitor. The extended specification language, namely Timed WSCoL now allows for specifying temporal properties over the events that occur during the process execution. In these regards, the approach is close to the one proposed in [5, 6]. In particular, the authors present classical linear temporal operators (always, sometime, until) and specific operators to express a property over restricted time window (within, between, count). Differently to the previous approach, the monitoring of these temporal properties is asynchronous, and performed by a new dedicated component called WSCoL Analyzer. This analyzer is executed in parallel with the execution of a process, and receives the relevant event through a dedicated publish/subscribe mechanism. Similarly to [5, 6], the automata-based algorithms are used to devise the monitor for a dedicated temporal property.

In [10] the authors show how this approach may be integrated with the WS-Policy framework. The work concentrates on the effective policies that represent the properties of a Web service deployed on a particular server and invoked by a specific user. Once effective policies are derived, services should be monitored at run-time to guarantee that they offer the service levels stated by their associated policies.

Monitoring for diagnosis [12]

In [12] the authors present a monitoring approach for the purpose of diagnosis of failures in service-based systems. While in most existing approaches fault recovery actions are associated to the observable effects of the occurred problems, it is necessary for a system to adequately react to their causes.

In the context of composite service-based systems, the goal of the diagnosis is “to identify the service responsible for the problem, the faulty activities, and the other services involved in failure”.

The presented approach deals with a particular case of the system diagnosis, namely consistency-based diagnosis, where the goal is to assign certain behavioral model to the components and observe its consistency with respect to the real execution. Furthermore, the authors present an approach to the diagnosis-aware fault handling, where the global hypothesis on the occurred problem drives the recovery actions within the application components.

The proposed solution relies on a distributed architecture, where each component service is associated with the corresponding local diagnoser, which interacts with the service through a special diagnosis interface. Such diagnoser collects the local information regarding the actions executed by its service and the messages it interacts with the other services, and derives local hypothesis about the problem and the other involved services. This hypothesis is then transferred to the global diagnoser component that is in charge of requesting other local diagnosers in order to collect the most complete information regarding the occurred problem. The global hypothesis, identified in this way, represents the cause of the problem and may be used to drive the fault handling. The latter is achieved by making the local fault handling diagnosis-aware, i.e., taking into account the hypothesis in the definition of the recovery actions.

The diagnosis information allows not only identifying the cause of the occurred problem, but also enables more accurate and coordinated recovery management. An intelligent local fault handling scheme proposed in the approach, as demonstrated and validated through an example of a composed business service, permits the resolution of a problem, not within the service, where it was observed, but within a component that caused the problem.

Monitoring privacy-agreement compliance [13]

In [13] Benbernou et al. address the problem of run-time monitoring of compliance of the privacy agreement defining the users privacy rights and their possible handling by the service provider. This problem goes beyond the traditional access control management and defines the necessity to face the usage control management of the private user information.

The proposed solution presents the privacy agreement model, where the requirements on the management and handling of the privacy data are specified, together with the approach for run-time compliance monitoring. The privacy properties are given in the form of data-rights (authorized operations) and data-obligations (required actions) together with their validity frames, and specified in the extended WS-Agreement specification. The set of privacy requirements, privacy units, and typical misuse scenarios are defined based on these properties. The formalism adopted for the representation of the privacy units and misuse relies on linear temporal logic. For the monitoring purpose, privacy units are transformed into state machine representation that correspond to the evolution of the privacy data management and define both correct and incorrect usage of this data.

The monitoring framework incorporates three main ingredients, namely requirements specification, privacy unit observer and monitor. The requirements specification is transformed into the corresponding privacy unit state machines, which at run-time evolve in parallel with the service execution. The monitor collects the information about the privacy data use from the service logs, and updates the status of the privacy unit observer accordingly. The latter reports the violations of the privacy requirements when a specific failure state of the corresponding state machine is reached.

The proposed framework relies on clear and simple model of privacy agreements, while the underlying requirements model relies on a comprehensive formalism for the representation of correct usage of private information. The run-time monitoring approach exploits automated techniques for the extraction and execution of monitor programs.

Requirements monitoring based on event calculus [14, 15, 16]

In [14, 15, 16] the authors approach the problem of monitoring service-based systems for conformance to a set of behavioral requirements. In particular, the works address the ability to represent and monitor complex and expressive properties that deal with events, states, timing constraints and relations, etc. Furthermore, there is a need to refer not only to functional but also to non-functional characteristic of the system, as dictated by the necessity of the run-time compliance checking between the actual behavior and the service-level agreement specifications [16].

In their approach the authors address the above problems by providing a formal specification framework for expressing complex requirements and SLAs, and a monitoring framework for run-time checking of these properties. The formal specification relies on the adapted version of the event calculus, a first-order logic-based formalism that allows expressing complex properties over events, fluents, and timestamps. The events and fluents in their notation refer to simple operations invocations, valuations of internal and external variables, built-in functions and operators, etc. In [16] this notation is adapted in order to reflect the SLA properties in the WS-Agreement notation. The monitoring process relies on automated extraction of the special templates that represent the formulas in event calculus, and updating them with the recorder and derived events. The reasoning schema, the derivation of events and run-time analysis are based on the inference rules in the corresponding first-order logic formalization of the calculus.

The monitoring framework was implemented as a toolkit for monitoring service compositions specified in BPEL. The logs generated by the process engine were used to identify the events and update the corresponding formula templates in the monitors. In order to evaluate and validate the presented approach, the authors set up a comprehensive benchmark with many test and generated events based on a simple case study parametrized by the frequency of events and the scale of the involved components.

Monitoring security patterns [17, 18]

In [17, 18] the authors address the problem of monitoring important security properties of service-based systems. While the static analysis techniques are widely used to check the security properties at design-time, the run-time verification of these properties and the assumptions, under which these properties were shown to hold, is still required.

In order to tackle with this problem the authors propose to use the techniques described by the previous framework in order to define and monitor basic security properties, namely confidentiality - the absence of unauthorized disclosure of information; integrity - the absence of unauthorized transformations of the state of a system; and availability - the readiness of a system to provide a correct service. These properties are defined using the special patterns modeled as event calculus properties, which allows monitoring security properties even for non-expert users.

Performance monitoring for utility computing [19]

In [19] Farrel et al. deals with monitoring of service-level agreements defining formal contract on the service provision between the service customer and provider. The presented solution addresses the domain of utility computing, where providing resources with certain quality characteristics is the cornerstone aspect.

Similarly to [14, 16], the presented approach exploits the event calculus as an underlying formalism. However, the way the contracts are designed and managed is different. The contracts are defined on the basis of a contract pattern (which obligations hold initially, which are initiated and terminated during the contract, which events and when occur), which is then axiomatized using the event calculus. Then, the effects of critical events on the contract state/evolution are defined. The authors present a monitoring architecture and an analyzer to manage contract life-cycle, their analysis and reporting, as well as a visualizer to represent the results of the SLA monitoring.

Planning and monitoring execution with business assertions [20]

In [20] Lazovik et al. apply monitoring to a framework, where the user requests are used to dynamically customize and execute standard business processes. Such customization aims at satisfying the user constraints and requirements to the execution of a standard business process, parametric to the set of available concrete services participating in the process. These services, however, as well as their composition, should satisfy certain domain business rules, referred to in the framework as assertions.

In this approach the authors propose an architecture, where the planning-based adaptation of the business process is interleaved with the execution and monitoring of the process and the corresponding assertions. The adaptation requests are specified in a XSRL query language that defined functional constraints and preferences of the user. The assertions are specified in the assertion language XSAL, which allows for defining the behavioral policies on the execution of process activities in the domain. Using these specifications and the process model, the framework tries to exploit various providers, in order to better satisfy the query, taking into account the domain assertions. While executing the adapted process, the framework monitors the relevant events and if the violation of the plan or assertion is detected, tries to dynamically modify the plan taking into account new situation and assertions.

Automated SLA monitoring [21]

In this work the authors address monitoring Service-Level Agreements (SLA), which represent certain form of a contract between service provider and consumer regarding such issues as performance, reliability, security, cost. This problem is made difficult by several factors. First, there is a need to present SLA in precise and unambiguous way. Second, due to distributed and multi-party nature of service-based systems, the relevant measurements should be done in a distributed way.

The proposed solution addresses these difficulties as follows. First, the authors present a formalized notation for specifying generic SLA, applicable also to the service-based systems. The main ingredients of the SLA specification refer to clauses, which define the measured parameter, on which side it is measured, upon which event it should take place, and the instructions for evaluation of the values. Second, the authors propose instrumentation facilities in order to collect the necessary data. At the service level this is done using special proxies that intercept messages and perform their evaluation. At the business process level the process logs (or database) are used to collect information about internal process activities. Third, a comprehensive service-level monitoring engine is proposed to carry out the management of the monitoring tasks. The engine performs the coordination of monitoring actions, raises and terminates new monitor instances according to the duration periods as specified in SLA, associates the monitored information with the monitored objects, and so on. The engine also supports distributed monitoring, if the measurements should be performed both at the consumer and provider side. In this case the measured information is being exchanged via specific measurement exchange protocol, supported by the engine. The implementation of the engine comes also with the management and audition facilities.

WSLA [22]

In [22] a framework for the specification and monitoring of service-level agreement is presented. The goal of the framework is to provide an expressive yet flexible model of QoS contracts and a run-time framework for their measurements and violation detection.

The WSLA framework defines a language for the specification of contract information that allows for describing the parties involved in the agreement (including third parties supporting the monitoring/audition process), the relevant quality of service characteristics, as well as the ways to observe and measure them, and the obligations and constraints imposed on these characteristics. The characteristics may refer both to simple and to complex metrics, which define composite properties of simple ones (e.g., aggregation, percentage, etc.). The measurement directives facilitate the definition of the means used to observe and analyze the relevant information, making the architecture and the approach flexible and easily extendable.

The WSLA architecture includes several components to provide the facilities for the deployment, measurement, and management of contract information. The actual monitoring is performed by the SLA compliance monitors that collects the measurements and probes from the provider and supporting parties, performs the evaluation of obligations and detects their violations.

Cremona [23]

In [23] Ludwig et al. propose an architecture and implementation for creation, management and monitoring service-level agreements represented as WS-Agreement documents. WS-Agreement specification provides a standardized way of defining contractual information between service provider and customer, and originates in the field of grid computing. It allows the clients to specify the required levels of service quality properties, the providers to derive the corresponding resource requirements and to drive the decisions on the requests. On the client side it is necessary also to provide facilities to monitor and control the actual correspondence of the resource quality and the negotiated contract specification.

The Cremona architecture and a toolkit propose a framework for all the aspects in the WS-Agreement management. Cremona provides a layered model for definition, creation, negotiation, binding, and monitoring of contracts. The monitoring module is not only used to observe and detect contract violations, but also to predict future violations and to engage corrective actions in advance. The monitoring facilities come also with the management interfaces that allow the customers to track the current situation and to take agreement-level actions (e.g., change provider).

Colombo [24]

In [24] the authors propose a platform for developing, deploying, and executing service-oriented applications and system that incorporates the tools and facilities for checking, monitoring, and enforcing service requirements expressed in WS-Policy notations. WS-Policy notations define the quality-of-service assertions that can be attached to a particular service, operation, or a message type. The concrete assertions are defined in a certain domain-specific language, e.g., WS-Transactions or WS-Security that define the properties of the transaction protocols and security characteristics respectively. Apart from checking the compliance of policies at deployment-time, it is necessary to verify them at run-time, when, e.g., service invocations calls/bindings take place or messages are sent/received.

The Colombo platform comes with the module that manages the policy assertions. Apart from evaluating the assertions attached to particular service-related entity, the framework provides means for policy enforcement, e.g., it may approve the delivery of a message, reject the delivery, or defer further processing.

2.3.2 Business Activity Monitoring

Business Activity Monitoring (BAM) provides near real-time monitoring of business activities, measurement of key performance indicators (KPIs), their presentation in dashboards, and automatic and proactive notification in case of deviations. A “business activity” thereby can be implemented as a service orchestration in a BPMS, or, more general, as part of a business process consisting of a series of activities implemented across workflow systems, ERP systems and legacy applications, possibly across organizational boundaries. BAM is based on event processing. Events have to be gathered from different kinds of applications which are not necessarily implemented as services: process engines, ERP systems, databases, legacy applications. Not all of these applications support publishing of events natively, so often adapters have to be implemented which extract events from these applications. BAM tools subscribe to these events and typically use Complex Event Processing (CEP) technology, which enables them to process high volumes of underlying technical events to derive higher level business events. A rule engine calculates KPI values based on business events and sends notifications. Events and KPI values are typically stored in a data warehouse of the BAM tool, which is queried by the dashboard component for visualizing the KPI values.

Query-based business process monitoring [25]

In [25] Beeri et al. propose an approach to the monitoring of business processes specified in BPEL. While the goal of the monitoring approach is similar to many other proposed approaches, the specific focus of the approach is different. In particular, the authors try to address the following monitoring design and implementation issues: the monitor specification should target the same level of abstraction as the original process; the monitoring activity should take into account the specific features of the underlying process models; the monitors should be deployed and executed in the same environment as the original process, without putting additional requirements on that environment.

For these purposes the Business Process Monitoring (BP-Mon) system is presented. In order to specify the monitoring properties, a high level intuitive graphical query language is proposed that allows users to visually define monitoring tasks. BP-Mon queries consist of two main ingredients: execution patterns that should be matched against the actual execution traces, and report specification generated from these matches. The patterns represent the composite events as partial control flow specifications, where the elements specify the activities that should appear in the critical execution. Additionally, the query contains the definition of the time window (period and interval), in which the query should be evaluated, and the condition to restrict set of matched executions. When the query is matched, the report is populated and emitted. The report represents a parametric XML pattern, which is instantiated when the pattern is matched. Two reporting modes are provided: a local one, where an individual report is issued for each process instance and a global one that considers all the instances.

To perform the run-time analysis, a specific pattern matching algorithm is proposed. The algorithm tries to greedily simulate the pattern to match events as early as possibly and backtracking on failure. A report is issued as soon as a match for the pattern is identified.

The system is implemented as follows. A BP-Mon query is compiled into a BPEL process specification, whose instances perform the monitoring task, which is translated into an executable code to be run on same BPEL engine as the monitored business process. An additional component, dispatcher, is used to listen to the events on the process activities, and forward them to the query process instance. An important feature of the approach is that it does not target a particular monitoring goal. Indeed, the reports provide just the required values, and therefore may be used for various purposes regarding BPEL processes.

Model-driven development of monitored process [26]

In [26] Momm et al. deal with the problem on how to develop automated SOA-based business processes with integrated monitoring information for process controlling. Automated BPEL-based business processes are often developed in a top-down manner, starting with a visual notation of the process (e.g. in BPMN) and then translating the visual model into an executable BPEL process model. If the BPEL process is to be monitored, then also process metrics have to be specified during process development.

The presented solution utilizes a model-driven approach to developing monitored business processes. The authors have created a metamodel which allows modelling of process performance metrics (PPIs) based on BPMN process elements. The BPMN process model with the corresponding PPI model is transformed to a BPEL process model which contains additional activities for publishing events needed for the calculation of the PPIs. These events are sent to an external monitoring tool by invoking its Web service interface. For measuring the duration of the activity, for example, two additional BPEL invoke activities would be inserted, before and after the activity, respectively. These activities would invoke corresponding operations on the monitoring tool.

The benefit of the approach is that events needed for monitoring are automatically determined and corresponding activities for event publishing are automatically generated. The authors demonstrate the approach based on a case study in the context of the management of examinations.

Model-driven development for BPM [27]

In [27] Chowdhary et al. deal with the problem on how to develop and deploy monitoring solutions in an efficient manner. Monitoring solution consists of different artifacts which have to be created: a monitor model which evaluates KPIs based on incoming events and sends notifications; a database schema which stores KPI values and allows efficient querying; dashboard views.

The presented solution introduces a model-driven approach to development of business process monitoring solutions. The authors have created a business performance management metamodel for modelling of monitoring tasks in a platform-independent way. The Business Performance Management (BPM) metamodel contains concepts such as “Metric”, “Business Event”, “Situation”, “Business Action”, etc. The user can create such models using UML. The BPM Model is transformed to two intermediate models: an observation model and a data warehouse model. The observation model captures information for the monitoring tool on how metrics are to be computed and which actions to take in certain situations. The data warehouse model deals with storage of metrics, and their visualization in dashboards. Both intermediate models can be adapted if needed. Finally, they are transformed to deployable code.

The benefit of the model-driven approach to creation of monitoring solutions is that much of the code can be fully generated. The approach has been implemented as part of the corresponding IBM Toolset (Rational Tooling for UML modelling, WebSphere Application Server, DB2).

Probing and monitoring WS-BPEL processes [28]

In [28] Roth et al. deal with the problem on how to extract events from a BPEL process in order to enable auditing in an interoperable way. The BPEL specification does not specify how the execution of the BPEL process should be logged in an audit trail. Each BPEL engine vendor implements a different event model and auditing mechanism. Thus, a monitoring tool would have to provide adapters for each BPEL engine it wants to support.

The presented solution extends a BPEL process model definition with special auditing activities which log state changes to an external monitoring web service. The extended BPEL process does not use any proprietary elements and is BPEL standard compliant. Therefore, the extended BPEL process can run on any process engine and send events to the monitoring tool. First, the authors introduce five different strategies for auditing BPEL processes: (i) instrumentation of web service requests of the BPEL process on protocol level and (ii) on application server level, (iii) utilizing the auditing service of a process engine used for enacting the BPEL process, (iv) using probes in the operational systems that track state changes of the business process, or (v) including the auditing mechanism as a partner within the BPEL process. They employ the fifth strategy and show how to transform a BPEL model into an auditable model which can be used for process monitoring purposes. For every audited activity, a new scope is created which hosts and executes all the necessary steps for pre- and post-auditing. For the monitoring service which is invoked by the extended BPEL process, an interface is presented. Finally, the authors propose some extensions to the BPEL specification for supporting their approach.

The benefit of the approach is that the extended BPEL process does not use any proprietary elements and is BPEL standard compliant. Therefore, the extended BPEL process can run on any process engine and send events to the monitoring tool. The authors have implemented a prototype consisting of a tool which extends a BPEL process adding auditing activities and a monitoring service which is invoked by the BPEL process at process execution time.

iBOM [29]

In [29] Castellanos et al. deal with the problem on how to create a monitoring solution, which not only enables to measure KPIs, but also to understand the causes of undesired KPI values, and prediction of future values. Conventional business activity monitoring solutions let the users define business metrics and then monitor and report them at runtime. They however do not support explanation of the causes of certain metric values to the user.

The paper sketches a platform which combines business activity monitoring with data mining approaches to enable more intelligent analysis of business metrics. The platform should support following functionalities: (i) providing visibility into processes which are not executed by a process engine, but run implicitly across diverse systems; (ii) enabling the business user to define KPIs in an intuitive way; (iii) enabling explanation and prediction of KPI values. In order to provide visibility into processes, the authors take the approach to model a so called abstract process which models the steps of the process in terms of events which are to be extracted from existing systems. Events are extracted by using adapters. The abstract process model is not an executable process model, but serves as an input to the monitoring tool in order to display the status of the process as events are received. In order to enable the business user to specify business metrics, a template based approach is employed. Thereby, IT engineers specify business metric templates using SQL queries over the underlying event data store. Business users instantiate the templates with concrete values, thus specifying the monitoring tasks without further support by IT engineers. Finally, for explanation and prediction of metric values, data mining techniques based on decision trees are used.

An Agent-based Architecture for BAM [30]

Jeng et al. (2003) have developed an agent-based architecture for providing continuous, real-time analytics for business processes. The architecture includes an integration layer, an event processing container (EPC), a BI agent layer, and a dashboard layer.

The integration layer extracts events from business applications and workflow systems. These events are provided to the EPC which is a robust, scalable and high-performance event processing environment. It is able to handle a large number of workflow events in near real-time. The incoming process events are transformed on-the-fly into metrics that are stored in the process data store. Furthermore, the EPC also publishes information to the BI Agent Layer for analytical processing. The BI Agent Layer is based on the decision cycle involving the five sub processes: sense, detect, analyze, decide, and effect. The sensing agents retrieve the events and metrics from the EPC or process data store, and provide them to reactive, deliberate, and proactive agent layers, which analyze and respond to situations and exceptions in a business environment based on business policies. Response agents generate action outputs unto the business environment by following the directives delivered from the agent layers. The authors demonstrate the approach and the implementation based on a use case on supply chain management for microelectronics manufacturing.

2.3.3 Process Mining

Process mining is an approach of observing and extracting certain knowledge about business processes execution(s) from available event logs [31]. In the context of service-based applications these logs may refer to registration of SOAP messages between services, event logs registered by the business process engines, etc. The classification of the processing mining approaches [31] defines the following usages of the process mining approaches to monitoring:

- *Discovery*, i.e., to reconstruct the actual model of the underlying business process as it is used in production environment. This information may be used for evolution purposes, that is, to see how the actual business process model evolves, or to identify certain trends or future problems.

- *Conformance*, i.e., the correspondence between the actual model and the expected one, (i.e., SLA or behavioral process model). The analysis may be both timely and post-mortem, when it is used for statistical or auditing purposes.
- *Extension*, i.e., enriching the original model with the additional, a priori unknown, information, such as performance characteristics, information influencing the decisions performed, etc. Again, the enriched model may be exploited for evolution purposes.

The development of a process mining and analysis solutions, often referred to Business Intelligence, presents several challenges such as:

- Defining the appropriate user interface to facilitate the use of the tools by non-expert users;
- Defining the metrics that can be used to design mining algorithms and evaluate results of the formers.
- Defining the underlying machinery enabling the above functionalities to be delivered to many levels of an organization.

Fuzzy mining [32]

In [32], the problem of simplifying processes on the basis of actual execution traces is proposed. This problem amounts to reconstruction of the process model with the objective of avoiding the generation of Spaghetti-like models. A Spaghetti-like model is defined as a model very complex and (usually) useless for humans that have to understand and handle it. These models are neither easily readable (too many elements) nor human-understandable. Therefore, some sort of simplified visualization scheme should be found to present resulting models to users.

The approach proposed in the paper tries to simplify the process schema by preserving (or, at least, trying to degrade as less as possible) two general metrics. Significance, defined for both activities and their relations, tries to keep as high as possible the fidelity of the behavior of the model with respect to the behavior of the actual model. Correlation is a metric that tries to match the amount of correlation between two activities. Generally, this is measured as the amount of data they share in the recorded events in the log. More precisely, the approach can be sketched as follows:

- Highly significant behavior is preserved, i.e., contained in the simplified model.
- Less significant but highly correlated behavior is aggregated, i.e., clustered together within the simplified model.
- Less significant and lowly correlated behavior is removed from the simplified model.

To estimate significance and correlation one has to rely on models built upon users preference. Log-based versions of the metrics are defined to take into account user activities. Derivative metrics, instead, are derived as combination of the previous ones.

Conformance checking with ProM [33, 34]

In [33] the authors address the problem of business process conformance checking. Conformance checking deals with the problem of analyzing whether the model (i.e., the workflow on which the BP is modeled) and actual execution traces (registered in the log) conform each other. Conformance analysis aims at the detection of inconsistencies between a process model and an event (i.e., trace) log through appropriate metric values.

The authors propose a post-mortem monitoring technique based on two different metrics:

- Fitness: does the observed process comply with the control flow specified by the process model?

- Appropriateness: does the model describe the observed process in a suitable way? Appropriateness is formulated according to an "Occams razor" principle: "One should not provide a model with a number of entities too much high than those required to explain anything".

In this work the model taken into account for processes is Petri Net and the two metrics above are defined in its terms. Fitness (that is the most dominant requirement for conformance) is defined as the ability of the Petri Net to "generate" all the traces observed in the log. To be appropriate, a "good" process model should be minimal in structure and minimal in behavior. The paper, itself, presents a lot of interesting metrics upon which a conformance checking algorithm is built. The conformance analysis in terms of these two metrics may be performed using ProM toolkit [34].

A noticeable point of the paper is that the algorithm is very nicely tested on a real-life log of events, generated by the use of a town-hall task management application. This is very interesting, actually, since many of the approaches present in literature either give theoretical proof for the validity of their model, or test them on synthetically generated workloads.

Process mining for security [35]

The work presented in [35] deals with the analysis of the security properties of the business processes. The analysis of security may amount to the presence or absence of certain action patterns during the execution, i.e., based on the analysis of audit trails. An audit trail is a record of all events that take place in a system and across a network, i.e., it provides a trace of user/system actions so that security events can be related to the actions of a specific individual or system component.

The work presents the approach to the analysis of audit trails using the process mining techniques. Using this techniques and set of "correct" audit trails, the algorithm extract the workflow that corresponds to correct handling of security data. Using the extracted model, it is possible to monitor the security properties at run-time by simulating the ongoing execution on that model. The authors also show how this method may be used in combination with the conformance analysis based on ordering relations.

Deriving protocol models [36]

In [36], the authors investigate the problem of discovering protocol models by analyzing real-world service conversation post mortem monitoring. They developed an approach for discovering protocol definitions from real-world service interaction logs. The protocol definition may not be available. This can happen for various reasons: for example, the service has been developed using a bottom-up approach, by simply SOAP-ifying a legacy application for which the protocol specification was not available; even when the protocol information is documented, like in Google Checkout service, the protocol specification need to be extracted from the textual description, which is not as precise as a formal model. So, a novel discovery algorithm is proposed, which is widely applicable, robust to different kinds of imperfections often present in real world service logs, and able to derive protocols of small sizes also thanks to heuristics. For verification, conformance and checking and evolution, protocol discovery is useful to verify if the designed protocol is faithfully followed in the service interactions. As the implementation of service evolves, the protocol definitions become increasingly incorrect. Automated protocol discovery helps in maintaining a correct and up-to-date protocol definition.

Furthermore, the most precise and the smallest model is algorithmically not feasible from imperfect service logs. They also propose an approach to refine the discovered protocol via user interaction, to compensate for possible imprecision introduced in the discovered model. The approach has been implemented and experimental results show its viability on both synthetic and real-world datasets.

Timed transition discovery [37]

A business protocol is not always published together with the service interface, and this hinders automatic management. When conversation logs are available, a solution is to discover the business protocol from past service executions, but there many difficulties can arise. One of them is the extraction of some temporal constraints called timed transitions, which are not explicitly recorded in the logs.

In [37] an approach for discovering such transitions is discussed. A formal class of patterns called proper timeouts has been defined. It was shown that the patterns reveal the presence of timed transitions in the business protocol. A polynomial algorithm for extracting such patterns was presented, as well as some preliminary experiments.

2.3.4 Grid Monitoring

According to the Anatomy of the Grid [38], the Grid is about "coordinated resource sharing and problem solving in dynamic, multi-institutional virtual organizations". Here is a practical example: with the help of virtual organizations a researcher can access restricted resources of another institutes, meanwhile these resources are still not exposed to the public. Grid technologies emerged in the mid 90s and from its very beginning its has converged towards standards based environments. As a result first basic internet based protocols were adapted like LDAP, HTTP, FTP, later web services technologies were also applied. Grid users usually employ resources for longer terms, therefore they require services that remember them. Grid uses a SOA with the extension of unified state management capabilities like those specified in the Web Services Resource Framework standards [39]. Modern service oriented Grid systems offer their data management, execution management, security management and information management interfaces as web services.

Grid monitoring concepts were laid in the late 90s when the first monitoring systems were built like the NetLogger [40] and the Network Weather Service [41]. Then in 2002 the Global Grid Forum tried to clarify the main grid monitoring concepts by defining the Grid Monitoring Architecture discussed later on. Grid monitoring systems developed afterwards were all using this work's concepts. In 2004 service orientation appeared in Grid systems. Thus new monitoring tools (e.g. MDS-4, SCALEA-G, MonALISA) were also necessary to support the two levels of SBSs, since monitoring should not just diagnose the service frontend, but it also has to try to analyze the service backend. E.g. in order to make sure a reliable file transfer service is available all the related file servers should be available also.

In grid the monitoring task can be defined on two levels.

Infrastructure monitoring tackles middleware level services. Provides the overall health of the Grid system. Enables higher level services to determine what services are available for their users. E.g. a Grid broker selects sites which have the necessary software infrastructure on them, and they also have enough computing power/storage capacity needed by the user's jobs. Information about middleware level services are usually published in a well established format like the GLUE schema [42].

Application monitoring handles the different programs the members of the virtual organizations use. Thus the main tasks of the monitoring system here is to deal with the application specific monitoring data, and the delivery of this data to the user. If the user applies special monitoring solutions, then larger grids might not be supported since the monitoring infrastructure should be constructed on a site the user does not even know before the execution. In large grids like EGEE (where tens of thousands of compute nodes are available), a grid broker selects the site where the application is executed, thus a unified, cross-site monitoring solution is required.

The aim of the application monitoring solutions can be performance analysis, debugging, migration, etc. Performance analysis is applied to determine the scalability of an application, or it can also be used to observe the effects of higher communication latencies between the different processes of an application. Distributed debugging systems can also use the monitoring system as a way to push debugging information towards the user. Finally the monitoring system can be the place where the current application state is reported. This state can be later used by an application migration utility to transfer the process states of the application to the newly selected site(s).

Grid Monitoring Architecture (GMA)

The Grid Monitoring Architecture (GMA) [43] is an abstraction of the essential characteristics needed for scalable high performance monitoring on a large distributed computational Grid. The GMA proposal [43], published by the GGF [44] Performance Working Group, defines the components *producer*, *consumer* and *directory service* (also called *registry*). Figure 2.1 shows the basic GMA components and their interactions. Besides component definition, proposal for interaction between components and required operations for each components are also defined. Basic requirements on monitoring architecture are defined as low latency, being capable of high data rate, minimal overhead, security and scalability. Most of theses requirements however are left to the implementation to fulfil and are not addressed at the architecture level.

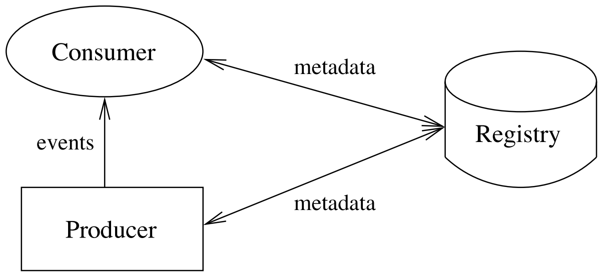

Figure 2.1: GMA components

In order to separate data discovery and data transfer, a service called ”directory service” is defined. Metadata describing producers of information and consumers waiting for some information are published in this service. Producers and consumers can then use published information to locate corresponding parties.

Several communication models between producers and consumers are identified: publish / subscribe, query/response, and notification. Operations supporting these communication models are defined for producers and consumers. Basic operations for directory service are also identified. In all communication models, monitoring data and control messages are sent directly between a particular consumer/producer pair and directory service is involved only in the location of the corresponding party.

In GMA, monitoring data are sent in the form of (time-stamped) events. Events are sent only from producer to consumer, but communication can be initiated by both parties.

The GMA document specifically mentions compound producer/consumer services. Such services may act as a gateway between different monitoring systems, or perform data archival or aggregation tasks. The possibility to use compound entities for load reduction is also mentioned.

Interoperability of different monitoring systems was one of motivations for the GMA. Definition of the GMA inspired several implementations of monitoring systems. While the GMA definition proved to be general enough to cover all these implementations, it proved to be to much general to guarantee real interoperability between proposed architectures. The GMA definition does not constrain communication protocols between components nor data model for description of producers and consumers.

The GMA mentions the importance of security to ensure the integrity and privacy of both the monitoring system and the event data itself, and mentions the X.509 PKI and SSL as possible technologies to use. The GMA however does not discuss authentication and authorization models and issues.

Strengths of the GMA proposal, that it describes many important aspects of grid monitoring and also provides definitions for common components. However it is too general, different GMA implementations usually can not interoperate.

SCALEA-G

SCALEA-G [45] is a unified monitoring and performance analysis tool for the grid. It is based on the Open Grid Services Architecture (OGSA) [46].

SCALEA-G consists of a set of web services. These include:

- The *Directory Service* serves as a registry for both producers and consumers. It also contains information about the data provided by every producer.
- The *Registry Service* is used to register and locate other SCALEA-G services.
- The *Archival Service* is a data repository holding past measurements.
- The *Sensor Manager Service* is used to manage sensors, store the monitored data and serve the data to consumers. Every Sensor Manager maintains a sensor repository that holds information about available system sensors, XML schemas, and sensors that must be activated at startup time.
- The *Mutator Service* conducts dynamic application instrumentation.
- The *Instrumentation Forwarding Service* receives instrumentation requests from clients and forwards them to the Mutator Service.
- The *Client Service* provides interfaces for administering other SCALEA-G services and accessing data in these services. The Client Service also supports features for analyzing performance data.
- A *GUI* enables the user to visualize the measured performance data.

Communication between various SCALEA-G components uses two methods: grid service operations are used for management tasks and interaction between high-level services, and TCP streams are used to transfer monitored data. The grid service operations utilize GSI [47] to provide authentication, privacy and integrity protection. Data channels are protected by SSL. Monitored data is encoded using XML, and data queries are expressed using XPath [48]. SCALEA-G supports both push-mode and pull-mode data delivery.

SCALEA-G distinguishes between system sensors and application sensors. System sensors are used to monitor the grid infrastructure (host, networks etc) while application sensors are used for monitoring application behaviour. The two sensor types differ in their control and security model. Every sensor (both system and application) has an identifier, an associated data schema and a set of parameters.

SCALEA-G supports application monitoring using either manual or dynamic instrumentation. Dynamic instrumentation is provided by Dyninst [49]. Security in SCALEA-G is based on Globus GSI. Access Control Lists (ACLs) are used for deciding which client can access what service/sensor. Strengths of SCALEA-G, that it has a flexible architecture, which also supports application monitoring, and provides a GUI for performance monitoring and visualization. Shortcomings of SCALEA-G include: pull-mode data delivery is implemented by the Sensor Manager always buffering sensor output, even if nobody is interested in it. And also the SCALEA-G assumes that only one user can monitor a given application at a given time.

Globus MDS-2

The Metacomputing Directory Service (MDS) [50] is part of the Globus Toolkit [51]. The first version was not very successful, but the second version (MDS-2) was widely deployed as part of various grid infrastructures despite its still numerous problems. MDS-4 is the fourth reincarnation.

There are two fundamental entities in the MDS-2 architecture: distributed *information providers* and specialized *aggregate directory* services. The information provider framework provided by MDS-2 is called Grid Resource Information Service (GRIS). It is based on an OpenLDAP [52] server back-end and allows plugging in various information sources. The data collected by various information sources (sensors) is cached inside GRIS to avoid overloading the machine by querying the information source too often. The framework for aggregate directory services is called Grid Index Information Service (GIIS). GIIS collects data from lower-level GRIS and GIIS services and mergest the collected data into an unified information space.

MDS-2 uses the LDAP [52] data model and representation. The LDAP is used as transport protocol and query language for GRIP, and as transport protocol for GRRP. The choice of LDAP however has several drawbacks:

- LDAP queries are efficient only if the structure of the information tree matches the query. If the query has to cross multiple branches of the information tree or has to involve more than one servers then performance suffers greatly.
- While LDAP is suitable for representing status information, it is not capable for delivering event notifications.
- Updating the information is quite slow with OpenLDAP.
- The schema is fixed and must be known in advance.

The caching nature of MDS-2 also presents problems. [53] and [54] show that MDS-2 performance is greatly reduced when the required data is not cached. However, most monitoring information has only very limited time period when it is valid so extensive caching is not possible.

MDS-2 uses the Globus GSI [47] infrastructure for authentication. Authorization is provided by OpenLDAP's access control facilities which allow a rich set of rules to be specified to control access to different parts of the information tree.

To overcome concerns about security due to the caching nature of GIIS, the GIIS only caches data that it acquired anonymously. The GIIS is also able to access data in lower level GRISes using a real GSI identity, but in this case it only returns a referral to the original GRIS provider to users, so every user is required to go through the authorization process of that particular GRIS in order to access the data.

Strengths of MDS-2 is its distributed, hierarchical design; and the standard protocols; finally the good authorization support.

Shortcomings of MDS-2 starts with the serious performance problems, continues with its only support for pull mod data delivery, then the monitoring information schema is fixed and must be globally known, and finally it has no support for actuators.

MDS-4

Globus Alliance has re-designed and re-implemented the MDS concepts using the Open Grid Services Architecture (OGSA) [46] and targeted all the previously mentioned shortcomings on some level.

As part of this effort the GRIS and GIIS was encapsulated within a single web service called the Index service. This service follows the Web Services Resource Framework [39] standards, and can monitor 3 types of producers. First WS-ResourceProperty [39] (WSRP) compatible services can be polled for their registered resource properties. Secondly it can subscribe for WSRPs of those services which support WS-Notification [55] by advertising their Resource Properties as topics. Finally for non WSRF enabled services or systems, the Index service periodically executes custom monitoring scripts.

With the new Index service they abandoned the old LDAP protocol and data model. They replaced the LDAP query protocol with the WS Resource Property query operation (supporting XPath language). As for the data model they are schema-agnostic, therefore the new MDS4 can store any kind of information users prefer as long as it is valid XML. The service supports pull data delivery with the help of WSRP, and push delivery on monitoring data changes through WS-Notification.

In the fourth release of the MDS a new service appeared called the Trigger. This service accepts conditional query definitions as input, then executes them regularly. If a condition becomes true it can initiate an actuator script.

R-GMA

The Relational Grid Monitoring Architecture (R-GMA) [56, 57] developed in the European DataGrid project is a relational database implementation of the GMA [43]. R-GMA is planned to be used not only as monitoring infrastructure, but also as generic information service. The R-GMA system uses web servlets implemented in Java, with SQL-like API. Client-side APIs are available in Java, C, C++, Perl and Python.

The R-GMA architecture identifies several types of producers:

DataBaseProducer - when the source of events is a real RDBMS,

StreamProducer - produces a continuous stream of events,

ResilientProducer - similar to StreamProducer, but the information is archived therefore it will not be lost even if the system crashes or is restarted,

LatestProducer - always provides the most recent event,

CanonicalProducer - generic producer class that does not have a predefined behaviour.

All the producer types except CanonicalProducer provide a simple interface for publishing events using SQL `INSERT` statements.

Producers register both their location and the description of the provided data in the registry. The registry includes a separate Schema component responsible for storing the data descriptions. Therefore R-GMA is capable of handling dynamic schema changes but only on the level of producers (i.e. if the schema provided by a producer changes, the complete producer registration process must be performed again). R-GMA does not support direct schema negotiation between the producer and the consumer therefore the registry may become a performance bottleneck when there are frequent schema changes.

There are three query types defined. History queries return events for a specified time period. Latest queries return the last published event. Continuous queries return events as soon as they are published by a producer. Therefore History and Latest queries implement pull-mode data delivery, while Continuous queries implement push-mode data delivery.

Consumers can express queries as SQL `SELECT` statements. Besides the original GMA components R-GMA also defines the *Mediator* although it is embedded in the consumer interface. The mediator is responsible for interpreting the `SELECT` statements issued by the consumer, redirecting the query to the appropriate producers by asking the registry, and combine the results from different producers.

Both producers and consumers are represented by a web service. While this provides maximum flexibility for the implementation and provides a standard interface to ensure maximum interoperability, it also turned out to be a major performance problem.

The original R-GMA design did not take security into account. Later R-GMA versions use the security framework of the European DataGrid Project [58]. Authentication is performed using GSI [47], on-wire privacy is provided by using SSL. The producer and consumer web services provide a simple access control mechanism to allow or deny consumers and producers based on their identity.

According GMA implementation comparison ([59]), R-GMA has several drawbacks: large number of external software dependencies, non-portable build system, and high run-time overhead due to its architecture. Also, the R-GMA API is quite large and complex.

Strengths of R-GMA are that it is a full implementation of the GMA proposal, and supports both push and pull mode data delivery, and it has a flexible design.

Shortcomings of R-GMA are that it has no support for actuators, it has a complicated API with a large number of external software dependencies and complicated administration, it also has performance problems partly because the architecture is too generic and partly due to the implementation, finally security was not included in the original design and was only added later.

MonALISA

The Monitoring Agents in a Large Integrated Services Architecture (MonALISA) [60] is based on Jini [61] technology. MonALISA is an ensemble of autonomous multi-threaded, self-describing agent-based subsystems. It is capable of interfacing with existing monitoring applications and tools.

The data collection is performed by the Data Collection Engine. The engine loads various data collecting modules dynamically and executes them in parallel. The collected data is stored in a local relational database.

MonALISA services register themselves with a set of Jini Lookup Discovery Services (registry). The Lookup Discovery Services provide automatic replication of common data. Communication between clients and monitoring services happens directly without the participation of the registry.

Archived data can be automatically compacted by averaging past values after a certain time. As a downside the only data type MonALISA framework can really handle well is a vector of floating point numbers.

Clients download proxy objects for the discovered services and use these proxy objects to perform both real-time and history queries or to subscribe to specific events. There is also a SOAP/WSDL binding available for non-Java clients but it does not have all the features of the native Java communication.

MonALISA provides a global GUI providing many visualization options. Every MonALISA service may provide the code for its own visualization using proxy objects. There is also a dedicated GUI for system configuration and management tasks. This administration GUI uses SSL certificates for authentication and authorization.

The dynamic loading of data collector modules and the flexible GUI for visualization are the strengths of MonALISA.

Meanwhile there are some shortcomings of MonALISA like: security is only available for the administrative GUI; java based, non-Java clients can not access all features; jini requires multicast which is not available everywhere; weak support for monitorable data types.

GridICE

GridICE [62] is a monitoring tool developed in the DataTag Project. The GridICE architecture consists of five layers: measurement, publisher, data collector, detection/notification and data analyzer, and finally presentation layer.

The measurement layer collects simple and composite metrics of resources. The measurements are stored locally. GridICE uses an extension of the GLUE Schema [42]. GridICE can use various third-party tools to implement measurements for the metrics defined in the GLUE Schema.

The collected data is made available for consumers by the Publisher Service. It acts as a gateway in case the monitored resource is behind a firewall. The Publisher Service is implemented on top of Globus MDS-2. This implies that only pull-mode data delivery is supported.

The Data Collector Service is responsible for detecting new available resources by periodically (typically daily) scanning the GIS. Once a resource is discovered, the Data Collector periodically fires observations to discover what monitored data is available and stores the result in a local PostgreSQL database. The Data Collector Service is implemented using Nagios.

The Detection/Notification Service can be used to automatically send out notifications when some monitored data matches certain conditions. The Data Analyzer Service provides general statistics and periodic reports of grid activity. The Presentation Service offers different views of the monitored data based on the type of the consumer. The Presentation Service is implemented in PHP.

Due to the implementation choices, the security characteristics of GridICE at the data collector level are equal to MDS-2, while at the presentation level only simple web-based access control is possible.

Shortcomings of GridICE are the centralized architecture, and that it does not support for event-like metrics, uniform security and it has no public producer/consumer API.

2.4 Comparing Monitoring Approaches

In this section we will summarize information on different contributions and compare the different approaches according to the above classification framework. The corresponding tabular forms will be provided separately for the above three aspects, namely monitoring task, specification, and implementation. We will also discuss the related works on monitoring approaches in other areas of information systems.

2.4.1 Monitoring Task

The results of the comparison of the presented approaches with respect to the monitoring task are represented in Table 2.1. The table represents the goal of the monitoring process, the role in the process of the stakeholder performing the activity, the main aspect addressed by the process, the kind and the scope of the monitored information, and finally the placement of this information within the overall architecture of the monitored application. We remark that many of the presented approaches were used not only for the purposes of detecting deviations from the expected execution (analysis), but for the purpose of correction (e.g., recovery from faults), dynamic adaptation (e.g., replacement of component services), and optimization. The use of these monitoring techniques in these is discussed in “Adaptation” part of the presented Deliverable.

The presented classification demonstrates that the research activities on monitoring has provided very rich and comprehensive set of approaches that cover wide range of goals, problem aspects, and information types, as well as the different components of the SBA architecture. The monitoring problem is considered from various stakeholder perspectives; it addresses both functional and non-functional aspects of the application execution, targeting not only the momentary information about the execution of a particular application instance, but trying to consider the system execution in “perspective”, when the relevant data is collected across the execution of all the instances of the application.

Another important trend that can be seen from the obtained results is that the research groups try to come out with the approaches and systems that integrated more and more various aspects of the monitoring task, such as different types of information, sources, types of event, their distribution, etc. The trend, however, in its infancy; only few approaches go beyond a particular monitoring problem or provide a wider perspective on the application execution. Furthermore, there are no existing approaches that cross all the functional layers of the SBA architecture, considers evolutionary aspect of executions with respect to a variety of information, etc. This, in turn, leads to another very important issue. Addressing only a particular aspect problem, or considering only a particular functional layer in isolation, does not allow to understand the relations and dependencies between

Table 2.1: Classification of approaches according the monitoring task

Approach	**Goal**	**Role**	**Aspect**	**Information**		**Location**		
				Kind	**Scope**	**Ext./Int.**	**Layer**	**Distr.**
Assumption-based monitoring	Analysis	consumer	behavior	func	instance and class	internal	SC	centr
Monitoring conversational Web services	Analysis	consumer (of involved services)	behavior	func	instance	internal	SC	centr
Smart monitors for Composed Services	Analysis (pre-post-conditions on services)	consumer (of involved services)	behavior	func	instance	internal	SC	centr
Dynamo	Analysis (assertions, temporal properties), Policy compliance	consumer (of involved services)	behavior, contracts, security	func and non-func	instance	internal and external	SC, SI	centr
Monitoring for Diagnosis	Diagnosis	consumer, provider	behavior	func	instance	internal	SC	distr
Monitoring privacy agreement	Analysis (privacy properties)	consumer	privacy	func	instance	internal	SI	centr
Requirements monitoring using event calculus	Analysis (requirements, contracts)	consumer	behavior, SLA	func and non-func	instance	internal	SI, SC	centr
Monitoring security patterns	Analysis (security patterns)	consumer	security	non-func	instance	internal	SI, SC	centr
Monitoring for utility computing	Analysis (contracts)	consumer	SLA	non-func	instance	internal	SI	centr
Planning and monitoring	Planning and adaptation	consumer (of involved services)	behavior	func	instance	internal	SC	centr
Automated SLA monitoring	Analysis (SLA compliance)	Third party	SLA	non-func	instance	internal	SI	distr
WSLA	Analysis (SLA compliance)	third party	SLA	non-func	instance and class	internal	SI	distr
Cremona	Analysis (SLA compliance)	provider, consumer	SLA	func and non-func	instance and class	internal	SI	centr
Colombo	Analysis (policies)	provider	service policies	non-func	instance	internal	SI	centr
Query-based monitoring of BPEL processes	General, in particular, BAM	consumer (of BPEL process)	various	func and non-func	instance	internal	SC	centr
MDD of monitored processes	BAM	consumer (of BPEL process)	KPIs	non-func	instance and class	internal	BP, SC	centr
MDD for BPM	BAM	consumer (of involved services)	KPIs	non-func	instance and class	internal	BP, SC	centr
Probing and monitoring BPEL	BAM	consumer (of BPEL process)	various	non-func	instance and class	internal	SC	centr
iBOM	BAM	consumer (of involved services)	KPIs	non-func	instance and class	internal	BP, SC	centr
An Agent-based Architectue for BAM	BAM	consumer of arbitrary business applications	business metrics, business policies	non-func	instance and class	internal	BP, SC	centr
Fuzzy mining	Analysis, evolution	provider	behavior	func	class	internal	BP	centr
Confromance checking in ProM	Analysis (conformance)	provider	behavior	func	instance and class	internal	BP	centr
Process mining for security	Analysis (security properties)	provider	behavior	func	instance and class	internal	BP	centr
Deriving protocol models	Analysis, evolution	consumer	behavior	func	class	internal	BP, SC	centr
Timed transition discovery	Analysis, evolution	consumer	timed behavior	func	class	internal	BP, SC	centr
Grid Monitoring Architecture	Grid	provider, consumer	behavior	func	instance and class	internal and external	SI	N/A
SCALEA-G	Grid	provider, consumer	behavior	func	instance	internal and external	SI	dist
Globus MDS-2	Grid	consumer	behavior	func	instance	internal and external	SI	dist
Globus MDS-4	Grid	provider, consumer	behavior	func	instance	internal and external	SI	dist
R-GMA	Grid	provider, consumer	behavior	func	instance	internal and external	SI	dist
MonALISA	Grid	consumer	behavior	func	instance	internal and external	SI	dist
GridICE	Grid	consumer	behavior	func	instance	internal and external	SI	cent

Table 2.2: Classification of approaches according the specification

Approach	Monitoring primitives		Abstraction	Notation	Formalism	Interleaving
	Type	Structure				
Assumption-based monitoring	message events, process start/end, modification of variable	simple	related to the composition specification	behavioral assumptions and goals in the form of logical sentences	linear temporal logic with time/count constructs	Defined separately
Monitoring conversational Web services	Service operations and messages	simple	service operation signatures	Algebraic specification of conversational services	Typed algebraic system over operation semantics	Defined separately
Smart monitors for Composed Services	Process variables	simple	process specification or programming level	Programming language or assertion language	First-order logic	process annotations
Dynamo	Process variables, external variables	simple, structured	process specification	Special assertion-based notation, WSCoL	First-order logic	Defined separately
Monitoring for Diagnosis	Service faults	composite (distributed faults)	composition specification	diagnosis description of expected behavior	N/A	Defined separately
Monitoring privacy agreement	privacy-critical service operations	structured	service operation	privacy extensions	Linear temporal logic	Defined separately
Requirements monitoring using event calculus	Internal/external variables, service operations	structured	composition and service specification	EC-Assertion language	Event calculus	Defined separately
Monitoring security patterns	Security parameters	structured	service specification	Security patterns	Event calculus	Defined separately
Monitoring for utility computing	Service operations	simple	service specification	ecXML	Event calculus	Defined separately
Planning and monitoring	Service-operations, process variables	simple	process specification	WSAL	Specific planning domain formalism	Defined separately
Automated SLA monitoring	Metrics over exchanged message	simple	Low-level information	SLA specifications	Various math languages	Defined separately
WSLA	Domain-specific metrics and events	simple, composite	Low-level information	Specific WSLA notation	N/A	Defined separately
Cremona	Specific metrics	simple	Low-level information	Specific contract model	N/A	Defined separately
Colombo	operations, messages	simple	Low-level information	WS-Policy	N/A	Defined separately
Query-based monitoring of BPEL processes	BPEL process events	composite (execution patterns)	process specification	control-flow pattern query language	Specific trace-based formalism	Defined separately
MDD of monitored processes	Process variables	simple	process specification	UML	N/A	Defined separately
MDD for BPM	Arbitrary Events	simple	process specification	UML	N/A	Defined separately
Probing and monitoring BPEL	Process variables	simple	process specification	N/A	N/A	Defined separately
iBOM	Arbitrary Events	simple	process specification	KPI Templates, SQL Queries	N/A	Defined separately
An Agent-based Architectue for BAM	Arbitrary Events	simple	N/A	N/A	N/A	Defined separately
Fuzzy mining	Process activities	simple	business process	Workflow/ Petri nets	Petri nets	implicit
Confromance checking in ProM	Process activities	simple	business process	Workflow/ Petri nets	Petri nets	Process model itself
Process mining for security	Security-related actions	simple	business process	Workflow/ Petri nets	Petri nets	Defined separately
Deriving protocol models	messages	simple	conversation	business protocol	finite state machines	implicit
Timed transition discovery	messages and timestapms	simple	conversation	timed business protocol	timed transition system	implicit
Grid Monitoring Architecture	Domain-specific metrics	N/A	N/A	UML	N/A	Defined separately
SCALEA-G	Domain-specific metrics	composite	Process and infrastructure	XML schema	N/A	Defined separately
Globus MDS-2	Domain-specific metrics	simple	Infrastructure	LDAP schema	N/A	Defined separately
Globus MDS-4	Domain-specific metrics	composite	Infrastructure	UML and XML schema	N/A	Defined separately
R-GMA	Domain-specific metrics	composite	Process and infrastructure	UML	N/A	Defined separately
MonALISA	Domain-specific metrics	simple	Process and infrastructure	N/A	N/A	Defined separately
GridICE	Domain-specific metrics	composite	Infrastructure	LDAP schema	N/A	Defined separately

various modules of the system execution, and to propagate the consequences across the application model in order to properly understand the real problem, and therefore to correctly react to it.

Certain critical aspects of the monitoring problem are still underestimated. This is the case, for instance, for the monitoring of contextual information. The monitoring approaches mainly focus on the events that are internal to its architecture and functionality, while with the increasing importance of open, mobile and pervasive service-based systems and application the role of the application context gains dramatically greater importance. Another important aspect that is not correspondingly addressed by the state of the art in monitoring is diagnosis. The ability to predict and explain the problems is a crucial requirement for the evolution of service-based applications, while only few approaches try to target this problem.

2.4.2 Specification

The results of the comparison of the presented approaches with respect to the specification of the monitoring information are represented in Table 2.2. The table represents the monitoring primitives used in the specification, level of abstraction of the specification, notation and formalism used (if any), and the degree to which the specification is interleaved with the application definition.

As in case of the monitoring task, the coverage of different aspects in the monitoring specification is rather broad. It covers variety of events, faults, service quality aspects and metrics, specification notations used, and so on. Another important observation is a strong correlation between abstraction level and the functional layer addressed by the task, as well as between the monitored events and information and the goal and the aspect of the monitoring task. This is also the case for the complexity of the underlying formalism and the addressed aspect: the more complex properties are addressed (e.g., structured properties over the application behavior) the more complex formalism is used for the representation of the task (e.g., extended temporal logics, event calculus, etc).

Literally all recent works follow the approach, where the monitoring specification is independent from the application specification. Such an approach enables clear separation of concerns and simplifies the modularization of the application design and development.

There are several problems, however, in the way the problem of specifying the monitoring information is addressed. First, due to the fact that the monitoring task is often rather narrow, the proposed specification notations address only the corresponding specific aspects. Correspondingly, the monitoring specification languages are often ad-hoc and very specific; each new extension proposes a special way to represent the same elements. Only few approaches use standards for this activity, often mainly adopting them in order to represent required information. As a result, there are many approaches that use different languages to represent the same information.

2.4.3 Implementation

The comparison of implementation techniques for monitoring approaches is presented in Table 2.3. The approaches are classified according to the method of capturing the relevant information, the way the monitoring is integrated into the application architecture and the execution process, the timing of collecting information with respect to its “actualization”, the sources of information, and the specific techniques used for that purposes.

A wide set of sources used for collecting the monitored information, ranging from very specific and internal events and data storages, to service message exchanges, to application logs. The set of methods to collect data, as well as the techniques to implement such a procedure, are also very broad. For simple events and properties, the event-based approach is typically in use, while for more sophisticated and structured properties the simulation methods are exploited in combination with the corresponding reasoning techniques. Obviously, this correlates with the complexity of monitoring tasks and the formalisms and notations used for their representation.

As for the degree of invasiveness of the monitored activity into the execution, the asynchronous techniques are more used. This is explained by the performance issues on the one side, and by the necessity not to affect the application logic on the other side. The synchronous methods are applied, when, for instance, certain correction actions should be applied in order to drive the application execution from some identified abnormal execution to the "normal" state. In a similar way, the post-mortem techniques are used, when the collected information does not require immediate reaction, but is needed for more evolutionary analysis.

The problem with many of the proposed implementation is the necessity to rely on a particular, very specific, and not always modularized architecture. Only few approaches propose solutions that are compatible with the standards systems, are easily adaptable and customizable. Many of the proposed approaches define very complex theoretical solutions that may fail to scale in industrial settings.

2.4.4 Related Works on Monitoring Information Systems

The necessity to carry out monitoring activities in order to support the management of a system in hand has been identified and widely studied in various IT disciplines. The monitoring goals and objectives in these areas - fault detection and recovery, system optimization and reconfiguration, diagnosis, management of business events - are similar to those identified in the monitoring of service-based systems, and have to tackle similar problems.

In [1] the problem of software fault monitoring is discussed. The survey targets the approaches towards run-time verification of behavioral properties of the analyzed system, such as execution assertion, safety properties or complex temporal requirements or models. The proposed classification focuses on the properties of the classification aspect, monitor implementation aspect, event handling mechanisms, and some specific operational issues (e.g., maturity, dependency, range of monitorable systems). Over thirty monitoring system were surveyed ad classified. One of the most significant conclusions made by the authors is the lack of approaches, where one can specify and monitor domain-level and design-level properties, not only the implementation-level ones. This is a critical issue for the complex software systems, where the ability to capture properties prior to implementation and then to check that they are met by the implementation is very important.

In [63] the problem of monitoring and management of events in active databases is discussed. The main purpose of event processing in active database systems is to be able to maintain data when relevant changes occur, to react to external events that have certain impact on the data, and to perform management tasks in conventional databases. While the approaches commonly follow the Event-Condition-Action (ECA) paradigm in the event processing, critical issues in the area deal with the proper architectural solutions for monitoring and processing events, timing and concurrency issues, cascading of processing, structure and distributed events, etc. In particular, analyzing the efficiency issues of different proposed approaches and implementation, the authors conclude that the proper architectural solution and intelligent support to the management and structuring of the monitored system is one of the most critical aspects in these regards.

In [64] a survey of the monitoring of distributed systems is presented. Apart from the specification and processing aspects of the monitoring process, the authors discuss the importance of validation, presentation and dissemination of the monitored information, as well as the problems peculiar to the distributed systems. In particular, this relates to the difficulties to obtain a global, consistent view of all components in a distributed system, synchronization, coordination of the monitoring activities, handling of events, etc. In many cases service-based systems are the instances of distributed systems, and these issues and problems are therefore equally applicable to their monitoring.

[65] presents a taxonomy for workflow analysis distinguishing between two dimensions: (i) technical analysis vs. business-oriented analysis and (ii) whether the analysis involves live data or history data. One can then further distinguish whether the monitoring task focuses on single instances or multiple instances of the workflow. Technical analysis involves, for example, system monitoring (live data) and workflow recovery in case of system failure (history data), whereas business oriented analysis copes with tracking of the individual process state (workflow monitoring) and workflow controlling which deals with the business-oriented ex-post analysis of potentially finished workflow

Table 2.3: Classification of approaches according the implementation

Approach	Method	Invasiveness		Timing	Source	Techniques
		Arch	Exec			
Assumption-based monitoring	simulation	integrated into BPEL engine	async	imm	message queues, process manager	automata- theoretic techniques for monitor extraction
Monitoring conversational Web services	Simulation	Integrated into BPEL engine	async	imm	Modified process executor to signal interactions	AOP for attaching monitors, symbolic execution of monitoring specification
Smart monitors for Composed Services	Event-based	original code is extended to interact with monitors	sync	imm	Internal data of the modified process	Process weaving to introduce monitoring operations
Dynamo	Event-based, simulation	The original process code is extended to interact with monitors	sync, async	imm	Internal process data, external services	AOP to attach monitors; automata-theoretic approaches
Monitoring for Diagnosis	Event-based	diagnosis is tightly integrated with service	sync	imm	diagnosis interactions and events	Specific protocol for hypothesis extraction
Monitoring privacy agreement	Simulation	Integrated with the service execution platform	async	imm	Service logs	extraction of monitors as state-machines
Requirements monitoring using event calculus	Simulation, inference	Extension to the process engine	async	imm	Process engine logs	FOL reasoning for identifying violations
Monitoring security patterns	Simulation, inference	Extension to the process engine	async	imm	Process engine logs	FOL reasoning for identifying violations
Monitoring for utility computing	Simulation	Separate monitoring platform	async	imm	Other monitors and observers	N/A
Planning and monitoring	Event-based	Monitoring is interleaved with execution	sync	imm	Invocation infrastructure	Specific planning/monitoring algorithms
Automated SLA monitoring	Event-based	monitoring engine is independent from platform	async	imm, deffered	intercepted messages, logs (DB)	N/A
WSLA	Situation-based	Measurement services integrated into the platform	async	imm	measurement services, messages, resource plug-ins	N/A
Cremona	Situation-based	Measurement and monitoring facilities are integrated	async	imm	Domain-specific plug-ins and tools	N/A
Colombo	Event-based	Policy management integrated into the platform	sync	imm	Invocation infrastructure	N/A
Query-based monitoring of BPEL processes	Simulation	Event dispatcher is embedded into the process engine	async	imm	Events corresponding to BPEL activity instantiation	Specific trace matching algorithm for simulation
MDD of monitored processes	Event-based	process code is extended to interact with monitors	sync	imm	Internal data of the modified process	Process weaving to introduce monitoring operations
MDD for BPM	Event-based	N/A	async	imm	any kind of application events	Querying a data warehouse / event store
Probing and monitoring BPEL	Event-based	The original process code is extended to interact with monitors	sync	imm	Internal data of the modified process,	Process weaving to introduce monitoring operations
iBOM	Event-based	N/A	async	imm	any kind of application events	Querying a data warehouse / event store
An Agent-based Architectue for BAM	Event-based	Extracting events from arbitrary systems using adaptors	async	imm	any kind of application events	Querying a data warehouse / event store
Fuzzy mining	Situation-based	Separated	async	post	Process logs	Data mining
Confromance checking in ProM	Simulation- and situation-based	Separated	async	post, imm	Process logs	Petri net-based analysis, data mining
Conformance for Security	Simulation-based	Separated	async	post, imm	Process logs	Petri net-based analysis, simulation
Deriving protocol models	situation-based	Separated	async	post	message logs	special data mining and noise estimation
Timed transition discovery	situation-based	Separated	async	post	message logs	detect possible proper timeout patterns
Grid Monitoring Architecture	Event-based	Separated	async	imm	N/A	N/A
SCALEA-G	Event-based	Partially integrated	async	imm	Internal process data, external services	N/A
Globus MDS-2	Event-based	Separated	async	imm	service, message and event logs	N/A
Globus MDS-4	Event-based	Separated	async	imm	service notifications/logs	N/A
R-GMA	Event-based	Partially integrated	async	imm	Internal process data, external services	N/A
MonALISA	Event-based	Partially integrated	async	imm	Internal process data, external services	N/A
GridICE	Event-based	Separated	async	imm	message logs	N/A

instances. The same classification is applicable to service-based workflows, e.g., implemented as BPEL processes.

BAM solutions are similar to reporting tools and business intelligence tools when it comes to display KPIs in dashboards. The distinction is that BAM tools process events in near real time, calculate KPIs and push the results to the dashboard. BI dashboards on the other hand query a data warehouse post-mortem when requested by the user or refresh the views periodically. Depending on the refresh interval selected, BAM and BI dashboards can be similar. Often, BAM tools also use a data warehouse schema for efficient querying of metric values. Another difference is that BI tools, in addition to reporting the values of KPIs, use data mining techniques to analyze the causes KPIs not reaching target values. In this context also, there are approaches which attempt to combine BAM and data mining for explanation of KPI values (see related work [29]).

The presented monitoring approaches in other areas of information systems give rise to questions and problems, which are very relevant to the service science as well. Among the most important one can name the problem of generality of a particular approach or technique. In software engineering the monitoring approaches are typically tightly coupled with the underlying application, implementation, or execution platform it is executed on. As a consequence, the solutions are difficult to migrate or customize in order to address new problems or systems. Related to this is the fact that many approaches are focused on a particular aspect of the system execution, which often leads to incomplete or insufficient reports that allow for detecting the result of the problem, but do not reveal the problem itself.

The study in databases demonstrate the necessity to not only come up with an expressive approach, but to provide an intelligent and efficient architecture and implementation for it.

In a similar way, the problem of distribution of the application architecture, and therefore, of the relevant information, makes the problem of monitoring coordination, management and synchronization very crucial in distributed systems. This is equally applicable to the service-based applications and service compositions.

2.5 Observations

Below we discuss the research challenges identified in the result of the presented survey work and make an attempt to give a vision on the research perspectives in the area of monitoring of service-based applications.

2.5.1 Research Challenges

In domain of service-based applications, where the implementation and management of the underlying services is not under the control of the system integrator, important application characteristics, such as correctness, often can be evaluated only at the production settings, and require dynamic means of the analysis. This leads to an increasing importance and spread of monitoring techniques in the development and procurement of the service-oriented architectures. A wide range of research and industrial approaches towards run-time monitoring of service-based systems have been developed recently. These approaches target the problem from different perspectives, and provide a broad spectrum of means for observing various application aspects at different functional SBA layers.

In spite of successful development in this area, the research community still has to face a lot of very important challenges in order to be able to deliver complete and mature monitoring solutions. One of the main problems in these regards relates to high fragmentation and specificity of the proposed approaches. A typical situation is that the approach addresses a very specific aspect of the system; it targets only a particular functional layer of the application in isolation of other aspects and components. As a consequence, the approaches often come up with a very specific, ad-hoc specification languages and methodologies, as well as specific and hardly adoptable implementation solutions. This not only makes the application of many monitoring approaches difficult in practice, but gives rise to another critical problem, the one of application diagnosability. Indeed, in order to be able to identify the real source of the detected problem, it is necessary to observe the execution of the application from all its aspects and to understand and infer dependencies between these aspects.

The new approaches to the problem of monitoring service-based applications should come up with holistic and comprehensive methodologies that:

- integrate various monitoring techniques and methods at all the functional SBA layers;
- provide a way to target all the relevant application aspects and information;
- define rich and well-structured modelling and specification languages capable of representing these aspects;
- allow for modelling, identifying, and propagating dependencies and effects of monitored events and information across various functional layers and aspects in order to enable the diagnosability of applications.

Another important and challenging problem for the research on SBA monitoring principles, techniques, and methodologies is the necessity to cross the application boundaries and be able to observe the relevant aspects of its environment. In other words, the problem of monitoring contextual information becomes more and more critical. This is particularly true for the hybrid service-based applications, systems in the domain of ubiquitous and pervasive computing, user-centric applications, etc. For those kinds of system the role of the environment plays crucial role; the ability to detect and react to the changes in the application context becomes a must in the provisioning of such systems.

Last, but not least, in order to achieve a universal adoption, the monitoring approaches should come up with smart architectures and efficient implementations. As it is the case for monitoring in active databases, the necessity to operate in highly loaded and dynamic environments puts additional requirements on the scalability of monitoring techniques and their implementations. The research should be able to address this aspects, potentially giving up with very expressive, but to complex and resource-consuming techniques and methods. Furthermore, the monitoring approaches should be implemented in such a way that they may be easily integrated with the standard service middleware platforms and infrastructures. This, in turn, requires well-defined interfaces between the monitoring management functionality and the underlying platform, thus bringing the problem of monitoring management and implementation to the level of SOA governance.

2.5.2 Research Perspectives

In the light of the identified research problems and challenges, we foresee several directions that will constitute the roadmap for the research on monitoring of service-based applications and systems. We believe, that these directions will provide a basis for the guidance and coordination of the research agendas of various institutions involved in this network of excellence, which focus on the monitoring problem.

First, in order to achieve considerably more comprehensive and holistic monitoring approaches that are crucial for delivering robust, dependable, and highly adaptable applications the integration of the efforts of researchers from various disciplines is very important. This include the "vertical" integration, where the business process management and service infrastructure should be brought together, as well as the "horizontal", integration, where different competences and development phases should be considered, such as requirements engineering, design methodologies, quality assurance, realization mechanisms, etc. These aspects are also critical for the new emerging forms of the service-based applications, where dynamicity and change correspond to nominal mode of executions. In these settings, the monitoring research will concentrate on proactive approaches, which deal with the ways to predict and/or prevent changes, and on the contextual approaches, where the changes are adequately modelled and considered. This trend already becomes a reality as the works and projects devoted to diagnosis and contextual aspects appear.

Second, in order to evaluate the real-life adoption of the novel holistic monitoring methodologies and platforms, to properly identify the boundaries, technological barriers, and scalability limits, there is a need to come up with the serious and comprehensive case studies and scenarios dictated by the industrial production settings of the service-based applications. In this way it will be possible not only to understand conceptual applicability and necessity of the monitoring approaches, techniques and methodologies proposed by the industry, but also to estimate its feasibility and to devise efficient and robust software architectures and implementations easily adoptable by the industry.

Chapter 3

Adaptation

3.1 Introduction

The dynamic nature of the business world highlights the continuous pressure to reduce expenses, to increase revenues, to generate profits, and to remain competitive. This requires Web services to be highly reactive and adaptive. It should be equipped with mechanisms to ensure that their constituent component Web services are able to adapt to meet changing requirements. In fact, services are subject to constant changes and variations. Services can evolve due to changes in structures (attributes and operations), in behavior (when services are interacting) and policies. Changes in policy assertions and constraint on the service prescribe, limit, or specify any aspect of a business agreement that is possible agreed to among interacting parties). Such changes can be identified, detected, and foreseen in the SBA during the monitoring of the application execution and its environment.

We need to know the reason why the adaptation is done, which impacts how the adaptation is done. Here are some purposes of the service adaptation:

Ensuring the interoperability: While standardization in Web services makes interoperability easier, adaptation still remains necessary. Adaptation is an important functionality that should be offered to enable integration inside and across enterprise boundaries. We need to generate a service that mediates the interactions among two services with different interfaces and protocols so that interoperability can be made effective. The need for adapters in Web services comes from two sources: (1) the heterogeneity at the higher levels of the interoperability stack (e.g., at business-level interfaces and protocols), and (2) the high number and diversity of clients, each of which can support different interfaces and protocols.

Optimization: The demand for quickly delivering new applications and adaptively managing them is a business imperative today. The QoS offerings in an application may change, new service providers and business relationships may emerge and existing ones may be modified or terminated. Furthermore, the general performance of the service is related to service velocity the time related aspect of business operations, such as service cycle time, round trip delays, wait time time between events. The challenge, therefore, is to design robust and responsive systems that address these changes effectively while continually trying to optimize the operations of a service provider.

Recovery: Various faults can occur relatively often and unexpectedly in distributed systems. It is necessary to handle faults reported during execution of the composition instance when monitoring business process. In fact, Web services compositions often implement business critical processes whose correct and uninterrupted operation is paramount. Therefore, to achieve dependable business processes, Web services compositions have to be made reliable. Reliability can be defined as the continuity of correct service delivery. This implies zero or, at worst, relatively few failures and rapid recovery time.

Context change: Services are constituted by reusable software components. The adaptations goal is to optimize the service function of their execution context. The same adapter searches and applies the possible adaptation solutions: component customization, insertion, extraction or replacement.

The issue of service evolution and change management is a complicate one, [66] highlights the challenges of service evolution. Papazoglou introduces some key approaches and helpful practices that can be used as a springboard for any further research in service evolution.

This survey deals with the characterization of the Web services adaptation problem by identifying and classifying different kinds of adaptation needs, requirements and approaches. The architectural aspect is not discussed in this report.

3.2 Classification of Adaptation Concepts

The papers which are surveyed in this deliverable are classified according to the following dimensions of the provided classification framework. The classification of the adaptation is based on the taxonomy found in the literature. We propose to discuss the adaptation in three dimensions:

- Adaptation requirements
- Specification of adaptation strategies
- Techniques for the implementation of adaptation approaches

3.2.1 Adaptation Requirements

This dimension includes the characteristics of the relations established between the monitoring artifacts and the changes of SBA addressed by the approaches; e.g., models, types, granularity, etc.

Adaptation models

Nowadays, the services used in the Web, are similar in terms of functionalities are offered through different interfaces (operation definitions) and protocol level (ordering the involved operations in the message. The default approach for a company, using one of such services, in switching to another similar service is to develop new clients for the new service. This approach is often time consuming and error prone. It requires redesigning, rewriting, redeploying client's code. In the service side, it is necessary to keep several versions of a service for different clients. To tackle with these problems, an approach was proposed developing adapters that mediate the interactions among two services with different interfaces and protocols so that interoperability can occur. To address the problem, in the literature, many approaches identified possible classes of mismatches (1) between service interfaces and (2) between protocols (behaviors).

1. Interface adaptation: The problem of adapting interfaces models in software have been studied in different contexts, and more particularly in the field of software components (e.g.,[67]) and also recently in Web services (e.g.,[68, 69]). They focused on highlighting signature mismatches between components/services.

2. Behavioral adaptation: Solving signature problems does not guarantee that the services / components will suitably interoperate. Indeed mismatches may also occur at the protocol level due to the ordering of the message exchanged and also blocking conditions, that is, because of mismatches of the involved service / components. The problem of adapting behavioral models in software have been studied in different contexts, and more particularly in the field of software components (e.g.,[70, 67, 71, 72, 73, 74, 75]) and also recently in Web services (e.g.,[76, 77, 78]).

Adaptation types

Two general approaches have identified to realize adaptation, namely *parameter adaptation* and *compositional adaptation*:

1. *Parameter adaptation* involves the modifications of variables that determine program behavior. It has been used in many systems in software component [79], in Web services[78]. However, a weakness of adaptation is that it can not adopt algorithms or components/services left unimplemented during design and construction of an applications. That is parameters can be tuned or an application can be directed to use a different existing strategy, but strategies implemented after the construction of the application cannot be adopted.

2. *Compositional adaptation* results in the exchange of algorithmic or structured parts of system with ones that improve a program's fit to its current behavior. In comparison to parameter adaptation, compositional adaptation enables an application to adopt new algorithms for addressing concerns unforeseen during the original design and construction (e.g., [80]).

The kinds of information

Both functionality/non-functionality have been studied in SBA. In fact, the core functionality of a service has been addressed for adaptation to meet the changing requirements. The mechanism of adaptation in web services has been studying into the process of web services choreography in terms of web services selection/invocation with pre-specified SLA constraints. Web service selection consists in identifying which is the best Web service with respect to the user needs. Service selection can be performed by two kinds of users: applications and human beings. Even if both aim at selecting the best Web service with respect to their needs, the basis on which application and human users perform the selection are slightly different. Such a difference derives from the model that they adopt to describe the quality of a Web service. Nowadays, specifications available for expressing the quality of a Web service are mainly addressed to the applications and do not deal with the situation in which the service users are human beings. Very little thought has been put forward in the direction of managing the non-functional requirements adaptively.In fact, the notion of adaptability in context of Adaptive Web services and/or processes till date have concentrated on capturing functional requirements of the adaptive behavior, catering to the variations in service consumer's preferences (e.g.,[81, 82]). Furthermore, the adaptation of the functional/non-functional properties follows a granularity of the techniques used in the approach. It can be local instance based (i.e.,, only current composition instance is needed), or global instance based (all instances of the SBA are required).

The research discipline an occurrence of the information

These characteristics show (1) where the adaptation is involved, Internally (current SBA) or externally (other SBA/components are involved) (2) which layer of SBA is concerned by. These layers are Business Process Management, Service Composition and Coordination, Service Infrastructure.

Formalization of the requirements to adapt

This dimension describes the level of formalization (formal, semi-formal, non-formal) of requirement specifications as well the notation or language which is used for the formalization which can be (1) goal based notation, where a *goal* describes why a system is being developed, or has been developed, from the point of view of the business, organization or the system itself. In order to identify it, both functional goals (expected services of the system) and non functional goals (quality of service, constraints on the design, etc.) should be determined (2) event-based requirements specifications (specifications are expressed in terms of required reactions to events (inputs, changes in environmental conditions,etc.). Furthermore, such specifications can be implicitly expressed through application invariants that were violated or explicit by using rules like “if events happen do that”.

3.2.2 Adaptation Strategies

The identified adaptation requirements from the results of the monitoring should follows procedure. In this section, we discuss the possible ways in which the requirements are achieved.

The dynamics of adaptation

Static adaptability involves a change in implementation due to the adaptation need, while *dynamic* adaptability involves a change in run time behavior without any changes in the implementation of the software/service system.

Automation

This characteristic allows classifying the degree of automation of the adaptation. The degree can range from total automated (no user intervention is needed), over semi-automated (some aspects use tools, and the other need user intervention), to manually.

The methodology

This characteristic shows the direction of the adaptation. It may be backward/top-down (the whole application adapts its structure, configuration, and/or composition) or bottom up /afterward (an application/system accepts input intended for later versions of itself).

The specification language

A notation is needed to specify the strategies to follow for the adaptation. It can range from procedural approach (concrete actions to be performed), over declarative (the description of the goals to be achieve), to hybrid.

3.2.3 Implementation techniques for adaptation

Here, we discuss some dimensions dealing with the implementation techniques for adaptation.

Timing of the adaptation

The adaptation can be mapped to the application lifetime. The services/software enables adaptation during the application design time, run time, or post-mortem i.e., when the execution is achieved.

The topology

The adaptation can be hold in a centralized manner, where the adaptation steps are defined and executed in a controlled way on all the affected components in the integrated way, or distributed manner, where adaptation is performed locally and then propagated among components. Furthermore, the adaptation can range from tightly integrated into platforms over injected into the platform at design/deployment time, to self adaptation or self-healing (it depends which taxonomy it is used). In the literature, self-healing refers to the automatic detection of failures and anomalies and their subsequent correction in a temporary or a permanent manner. Self-healing systems are of particular interest as they directly impact improvements in dependability. Self-adaptive systems are ones that monitor their execution environment and react to changes by modifying their behavior, in order to maintain an appropriate quality of service. Obviously, there is a substantial intersection between self-healing and self-adaptiveness: self-healing systems may be viewed as a special kind of self-adaptive systems.

The techniques

Four key software technologies are proposed to support adaptation. Computational reflection enables service/system to inspect, reason about, and adapt itself at runtime. Component-based design enables decomposition of SBA functionality, which makes it easier to manage and modify the structure of the applications both statically and dynamically. Aspect oriented programming enables separation of applications cross-cutting concerns (such as quality of service, energy consumption, security, etc.) at development time;later at deployment and run time, these concerns can be selectively woven into the application code.

3.3 Related approaches to adapt in service-based application

Researchers and developers have proposed a variety of methods for supporting adaptation in SBA. In this section we list a number of projects and commercial software packages that supports some form of adaptation. The list is by no means exhaustive, many other groups both in academia and industry, are working on various aspects of this problem. We grouped the approaches into four groups: semantic and correctness for adaptation, QoS in the adaptation, self adaptation systems, and adaptation in service interaction/composition

3.3.1 Semantic and Correctness for Adaptation

Ensuring syntactical correctness, is often not sufficient. In particular, for processes undergoing frequent changes performed by various staff members, mechanisms to ensure the semantic correctness of the processes become necessary. For this purpose, mechanisms to integrate semantic domain knowledge into adaptive service based applications are required. In this group, we discuss frameworks for supporting semantic knowledge integration and semantic process verification in the context of changes, and how the changes can be realized in a correct and consistent manner.

Integration and verification of semantic constraints in adaptive process management IVS [84]

In practice, approaches to ensure system correctness after process changes at the syntactical level have been developed. Those approaches tackle only a subset of possible changes and not consider changes related to the semantics (business rules, policies,etc). Recently, Ly et al. [84] investigated the design of a framework allowing the specification of semantic constraints over processes in such a way that maintenance and semantic process verification can be supported. The authors introduce techniques to ensure semantic correctness for both single and concurrent changes.

Protocol Mediation for Adaptation in Semantic Web Services PMAS [86]

[86] focus on web services protocol mediation by providing a framework that allows interaction between two services despite the difference of the protocols they rely on. Mediation allows automatic adaptation of a service requester behavior while meeting the interface constraints of the service provider. This is done by abstracting from the existing interface descriptions before services interact and instantiating the right actions when contracting a service. To be possible, a framework managing these levels of abstraction was provided. It consists of defining abstract primitives used by services during their interaction that will be mapped to concrete primitives that represent the real actions in terms of messages exchange between the two communicating parties. Monitoring is however necessary for ensuring the correct use of the abstract primitives that must follow the constraints defined for the service in a low level description language. It could be stated in a state chart expression. An ontology of shared concepts (of the business domain for example) where each concept used in the protocols is defined is also necessary for understanding the semantics of the domain actions which in addition have to exploitable by machine.

On Dealing With Semantically Conflicting Business Process Changes SCBP [87]

In [87], the authors discussed how to correctly propagate changes occurring in a business process to the process instances. They identified and classified semantic conflicts between the process type and instance caused by the changes in the process schema. Their paper describes formal dynamic methods for conflicts detection arising at change propagation time. More precisely, in order to enable the process management system to semantically detect conflicting changes the authors provided two methods: the first is based on execution equivalence of the respective process schemes, and the second, is based on a direct comparison of the applied changes.

The METEOR-S Approach for Configuring and Executing Dynamic Web Processes [88]

In [88] a framework for service compositions reconfiguration, called Meteor-S, is presented. This framework supports the execution and the reconfiguration of a service composition giving more flexibility to the service binding operations. Binding can be performed at design time, deployment time, and just before the execution, but it does not support service run time rebinding (i.e. in case of service invocation error it is not possible to find another candidate for binding without stopping the composition run). The composition is divided into scopes; semantic web languages are used to describe both domain constraints and services; matchmaking algorithms are used to associate each scope with the concrete set of services to invoke.

A Constraint Satisfaction Approach to Non-functional Requirements in Adaptive Web Services CSANF [81]

In [81] is presented a constraint satisfaction based approach to model non-functional requirements in the context of Adaptive Web services. Policy reconciliation between service providers and service requesters in adaptive SOA systems is often a difficult requirement to satisfy owing to the generality of the policies not only those associated with a single service provider but across multiple services. Addressing this dynamic adaptation requirement for policy based non-functional requirements, the authors discuss a constraint satisfaction based framework to represent, model, and work with policy based non-functional requirements in Adaptive Web services. They defined two level of constraints (1) Global level constraints, These constraints can be binary (only two services are involved) and tertiary for instance , the temporal constraints related to the service execution order (2) Service level constraints, where constraints are specific to each service. The partner service has to be consistent with the service level constraints defined by a service.

Evaluation of Correctness Criteria for Dynamic Workflow Changes ECD [89]

In [89], the authors introduced criteria that should be considered to perform correct adaptation of running workflows. Those criteria can be used as a benchmark for evaluating the strengths of the proposed approaches. In [89],the authors identified the rules and information that are needed for satisfying the criteria.

Adaptive Service Flows with BPEL ASFB [90]

In [90], the authors investigated the problem of consequences of enriching BPEL engines with dynamic change capabilities. They focused on the question of how changes can be realized in a correct and consistent manner and which prerequisites must be met in such a context.

3.3.2 QoS in the Adaptation

Business environments in which Web services must function are often volatile. The quality-of-service (QoS) parameters as well as the execution data may change during the life time of a business process. Both data and component volatility may adversely affect the performance of a Web service composition over time, so the adequate adaptation is needed. This group highlights the QoS in the adaptation.

Parameter Adaptation PAR [78]

Web processes must often operate in volatile environments where the quality of service parameters changes during the life time of the process. In order to remain optimal, the Web process must adapt to these changes. [78] addresses such problem . The authors provide a mechanism called the value of changed information which measures the impact of expected changes in the service parameters on the Web process. They use service expiration times obtained from pre-defined service level agreements to reduce the computational overhead of adaptation. They empirically demonstrated the speedups obtained in executing and adapting a Web process to changes in service parameters when using a method that is cognizant of expiration times in comparison to an adaptation strategy that ignores them.

Negotiation of SLAs: An Architecture and a Search-Based Approach NSLA [91]

In [91] an approach for negotiation of Service Level Agreements is presented. This approach defines a framework that can use different negotiation algorithms and protocols, defined at design time. Negotiation can either be performed directly by the interested stakeholders or it can be automatic. In this second case, human beings are replaced by automated negotiators that try to achieve the objective that has been suggested to them. Automated negotiation is particularly important when the consumer of a service is a software systems that has to negotiate on the fly (part) of the SLA with the service. The paper presents a multi agent system-like framework that is used to perform SLA negotiation. The architecture of this framework, is composed of a Marketplace and various Negotiation Multi Agent Systems. Each negotiation participant is associated to a multi agent system and can represent a human being or an automatic system. In the second case the system encapsulates a decision model that features various search-based algorithms. The Marketplace is composed of two main parts, one taking care of the execution of the negotiation workflow and the second one controlling the correctness of the negotiation protocol. In particular, the Marketplace acts as an intermediary in all interactions among the participants, providing validity checks for the offers exchanged (through the Protocol Engine), based on their structure and the current state of the negotiation workflow.

A user driven policy selection model UDP [92]

In [92] a model is presented expressing the quality of service according to both applications and human users perspectives. Such a model not only mediates between the application and human user perspectives, but considers the different relevance assigned by the user to a given quality dimension. Secondly, the paper introduces a policy selection model based on the adopted quality model. According to this selection model, a human user can express its requirements according to a high level language and its requirements are matched against the Web service quality specification expressed through a technical language.

Model identification for power management of web services LPV [93]

In [93] the identification of LPV models for the performance control of Web services is addressed. In the context of Web services, hosting centers need to comply to the Service Level Agreements (SLAs) stipulated with their customers while minimizing the operating costs, mainly related with the energy expenses due to servers operations. Recently, this problem has begun to be formalized in terms of a constrained control problem, where the SLAs are translated into set-points for the response time of the servers, possibly different according to the customer class, and tracking performance must be traded-off with energy saving objectives. As the behavior of the server response time is highly time varying and the workload conditions substantially change within the same business day, the LPV framework seems very promising for modeling such systems, since it better fits situations in which variability is high, with respect to the traditional queue nets model. Specifically, the suitability of subspace LPV identification methods is investigated and their performance assessed on experimental data measured on a custom implementation of a workload generator and a micro-benchmarking Web service application.

3.3.3 Self-Adaptation and Self-healing Systems

Here are grouped the service based applications which support the self adaptation mechanism. They can automatically detect failures and anomalies and their correction.

WS-DIAMOND: DIAgnosability, MONitoring and Diagnosis of Web Services [94]

In [94] the basic project assumptions are presented, in terms of a self-healing platform supporting the self-healing execution of complex Web Services. The focus is on the problems that occur at run time. A second general consideration is the focus on diagnosing problems that occur at run time and we are not considering the issue of debugging a service. Web Service execution environments are extended to include features which are useful to support the diagnostic/fault recovery process. Then, an architecture supporting self-healing service execution is illustrated. The WS-Diamond Project aims at the development of a framework for self-healing Web Services, that is, services able to self-monitor, to self-diagnose the causes of a failure, and to self-recover from functional failures (e.g., the inability to provide a given service) and from non-functional failures (e.g., loss of Quality of Service: QoS). The focus of WS-Diamond is on composite and conversationally complex Web Services. The second goal of WS-Diamond is to devise guidelines and tools for designing services in such a way that they can be easily diagnosed and recovered at execution time, and tools to support the design of complex self-healing adaptable processes.

Towards Self-Adaptive Service-Oriented Architectures TASOA [95]

In [95] the problem of discovering and adapting mismatches at run time is studied. This paper proposes to discover mismatches at run time associating to a candidate service a test suite. At run time the conformance of a request to the service interface and protocol will be verified by a proxy running the test suite associated with the service. In case the proxy finds a difference between supported requests and present requests it will try to transform the request according to the information it can extract from the run tests. The main limitations of this approach are that the test suite has to be manually provided by a developer that analyzes the candidate service and that usually many tests runs are required for a proxy to provide a suitable transformation. Moreover the approach is as effective in discovering and solving mismatches as the provided test cases are exhaustive, which is an undecidable problem.

Self-Healing BPEL Processes with Dynamo and the JBoss Rule Engine SELF [96]

In [96], present a solution for self-healing of BPEL processes is presented. The approach is based on Dynamo, a monitoring framework, together with an AOP extension to ActiveBPEL, and a monitoring and recovery subsystem that uses Drools ECA rules. A composition designer provides assertions for invoke, receive or pick activities in the business process. These assertions can be specified using two domain specific languages: WSCoL, the Web Service Constraint Language and WSReL, the Web Service Recovery Language, which respectively define Web service monitoring and recovery rules. When an assertion is not verified it is possible to execute recovery actions associated to that assertion, in order to bring the business process back in a consistent state.

Automatic Learning of Repair Strategies for Web Services ALR [97]

In [97] a methodology and a tool for learning the repair strategies of Web Services to automatically select repair actions are proposed. The methodology is able to incrementally learn its knowledge of repairs, as faults are repaired. Thus, it is at runtime possible to achieve adaptability according to the current fault features and to the history of the previously performed repair actions. This learning technique and the strategy selection are based on a Bayesian classification of faults in permanent, intermittent and transient, followed by a comparative analysis between current fault features and previously classified faults features which suggests which repair strategy has to be applied. Therefore, this methodology includes the ability to learn autonomously both model parameters, which are useful to determine the fault type, and repair strategies which are successful and proper for a particular fault.

Automated Generation of BPEL Adapters AGA [98]

In [77] a methodology for the automated generation of adaptors capable of solving behavioural mismatches between BPEL processes is proposed. In this methodology BPEL processes are translated in an intermediate representation called YAWL. This representation is used as input for a match making heuristics that tries to build an adapter for received requests. Once the adapter is generated, the interaction between the invoker and the service is analyzed, using lock analysis techniques, to discover if some interaction scenarios cannot be resolved.

3.3.4 Adaptation in Service Interactions and Compositions

While standardization in web services makes interoperablity easier, adaptation still remains necessary. The service adaptation discussed in this group refers to the mechanism of generating a service, that mediates the interactions among two services with different interfaces and protocols so that interoperability can be made effective.

Aspect-Oriented Framework for Service Adaptation AOPA [99]

In [99] mediators are considered as cross-cutting concerns and are implement as aspects. For this purpose join points have been defined for BPEL and solution patterns for some possible mismatches have been defined. Even if this approach provides separation of concerns for a developer it anyway requires BPEL process modification when aspects are woven into the base system code. This implies that services have to be statically bound to invokers.

Semi-automated adaptation of service interactions SAAI [69]

In [69] a tool to assist a system integrator in developing adapters for web service integration is presented. This tool works under the assumption that the requestor is also a service. The tool takes as input an interface and behavioral description of the requestor and of the service and provides hints to the human system integrator about possible mismatches it finds matching the input representations. When a mismatch is found the developer has to specify a function to solve it. The main limitation of this tool is that it supposes the developer has a pervasive knowledge of the system and can use low level hints to produce functions to solve a given mismatch. This assumption appears to be a strong one, considering that in a service oriented system invoked services can be third party components, with an unknown implementation and partial documentation.

Developing Adapters for Web Services Integration DAI [101]

In [101] a classification of mismatches featuring solution patterns is proposed. A mismatch is defined as the set of evidences that shows when there is a difference between the invoked service and the service the requester is expecting to invoke. The paper also proposes a solution pattern for each classified mismatches, as a BPEL mediator between the requester and the available service. These mediators should act as adapters, offering the expected interface and protocol to the requester and interacting with the available service with the interface and protocol the latter offers. The solution patterns proposed in this paper require that the service workflow is modified in order to integrate adapters.

Run-Time Adaptation of Non-functional Properties of Composite Web Services Using Aspect-Oriented Programming RAAOP [82]

In [82], the authors are focusing on run-time adaptation of non-functional features of a composite Web service by modifying the non-functional features of its component Web services. The aspect-oriented programming (AOP) technology is used for specifying and relating non-functional properties of the Web services as aspects at both levels of component and composite. This is done via a specification language for representing non-functional properties. From the end users' viewpoint, the aspect-oriented modeling of non-functional properties enables on-demand composite Web service adaptation with minimal disruption in quality of service.

Adaptation in Web Service Composition and Execution ASCE [102]

In [102] the authors developed a staged approach for adaptive WSCE (A-WSCE) that cleanly separates the functional and non-functional requirements of a new service, and enables different environmental changes to be absorbed at different stages of composition and execution. The main contributions are: (a) A characterization of the problem of Adaptive Web Service Composition and Execution ($A - WSCE$) (b) A staged solution that allows adaptation by generating multiple workflows at different stages, and selectively deploying them based on feedback mechanisms and suitable ranking functions. The incorporation of the proposed system within an $end-to-end$ service creation environment that demonstrate the effectiveness of the solution.

SCENE: A Service Composition Execution Environment Supporting Dynamic Changes Disciplined Through Rules SCENE [103]

In [103] the SCENE platform is presented. This platform provides the runtime execution environment for service compositions, specified using BPEL and an ECA rule language. SCENE incorporates a BPEL engine, responsible for the execution of the process part of the composition, an open source rule engine, Drools, responsible for running the rules associated to the composition, and WS-Binder [104]. This is responsible for executing binding actions at runtime based on directives defined in the rule language. This component is able to execute various policies for selecting the candidate services. When a SCENE composition is deployed, a deployer component analyzes it, identifies the cases in which dynamic binding and rebinding have been foreseen by the designer, generates some proxies that will be devoted to manage these cases and modifies the BPEL code to include calls to these proxies where they are needed. The role of the proxies is to mask to the BPEL engine the presence of the rule engine and the Binder in charge of managing the actual binding to services. Each proxy is assigned a WSDL to describe its interface and is exposed as an abstract service. At run time when an abstract service is called inside a process the call will be forwarded to the associated proxy which will be in charge of forwarding the call to the selected concrete service. The selection process takes place on the base of the rules specified by a composition designer.

An approach to adapt service requests to actual service interfaces ASRAI [105]

In [105] the SCENE framework is extended to support run time resolution of differences between interfaces and protocols of invoked services. This paper defines a classification of possible mismatches, a set of predefined adaptation functions to solve each mismatch, a language to combine these functions in adaptation scripts and a component, called adapter, integrated in SCENE proxies that can execute the scripts specified by a system integrator at design time. When a designer defines a service composition he/she individuates for each invocation a service, called primary service, whose interface and protocol are used to build the invocation in the composition. When a proxy for this invocation will be created it will be assigned the chosen interface and protocol. At this point the designer can select other services, having possibly different interfaces and protocols from the chosen one, that can be used for possible rebindings at run time. He/She has to specify an adaptation script for the remaining concrete services in which he/she specifies the differences between the primary service and the other selected for rebinding. For each difference the script contains a rule that specifies which function, from the set of predefined ones, has to be used to solve the difference. At run time, when one service different from the primary is invoked, the proxy instead of forwarding the request directly will pass it to the adapter. The latter interprets the script eventually associated to the invoked service and adapts the request. Finally the adapted response is returned to the SCENE proxy that provides to forward it to the client.

Adaptive Service Composition in Flexible Processes ASF [106]

In [106], a new modeling approach to the Web service selection problem is introduced. This approach is particularly effective for large processes and when QoS constraints are severe. In the model, the Web service selection problem is formalized as a mixed integer linear programming problem, loops peeling is adopted in the optimization, and constraints posed by stateful Web services are considered. Moreover, negotiation techniques are exploited to identify a feasible solution of the problem, if one does not exist. Experimental results compare our method with other solutions proposed in the literature and demonstrate the effectiveness of our approach toward the identification of an optimal solution to the QoS constrained Web service selection problem.

PAWS: A Framework for Executing Adaptive Web-Service Processes [107]

[107] introduces PAWS (Processes with Adaptive Web Services), a framework for flexible and adaptive execution of managed Web service-based business processes. In the framework several modules for service adaptation are integrated in a coherent way. An original characteristic of this framework is to couple design-time and run-time mechanisms for process specification and execution in a global framework. At design-time, flexibility is achieved through a number of mechanisms, i.e., identifying a set candidate services for each process task, negotiating quality of service, specifying quality constraints, and identifying mapping rules for invoking services with different interfaces. In turn, the run-time environment exploits the design-time mechanisms to support adaptation during process execution, in terms of selecting the best set of services to execute the process, reacting to a service failure, or preserving the execution when a context change occurs. The application of PAWS in several case studies is finally discussed.

Business process design: towards service-based green information systems BPD [108]

[108] discusses the impact of energy consumption on information systems and business processes design. The goal is the development of context-aware and sustainable information systems where energy consumption reduction is considered at the technological level, on the basis of adaptable technology, on the governance level, with the design of context-aware processes and data, and at the strategic level. The adoption of a service-oriented approach and the interconnection among the different levels are discussed.

Context Management for Adaptive Information Systems CMA [109]

In [109], an approach for context-aware composition of E-services based on an abstract description of both E-services and context is proposed. E-services are described in terms of functionality and quality of service. The context describes the channels that can be used to access E-services. The paper proposes adaptation rules as the means to allow the composition and dynamically select E-service channels according to the constraints posed by available architectures and application-level requirements. Composition and adaptation rules are exemplified on a simple case for emergency management.

3.4 Comparing adaptation approaches

To better assess the current state of the art, in this section the three dimensional taxonomy, introduced in section 3.2.1 is mapped into three tables that provide a higher-level view of the detailed discussions provided in Section 3.3 on the basis of the requirements we listed earlier. All the approaches have their own drawbacks and advantages. None of the proposals are complete, even though current approaches address significant subsets of relevant requirements.

3.4.1 Adaptation requirements

Figure 3.1 describes the comparison of the presented approaches with respect to the adaptation requirements. The figure depicts the objective of the adaptation process, the kind and the scope of adapted information, which model is based on to perform the adaptation activity, the placement of this information within the overall architecture of the adapted application and finally the specification language to express the adaptation process.

As this table shows, adaptation has gained a significant attention in complementary way of the monitoring process to overcome the mismatches between services for service integration, repairing the failures, reconfiguring the systems by leveraging existing approaches in software engineering. We also notice that, the notion of adaptability in SBA till date have concentrated on capturing functional requirements, few thought and projects have been done on the non functional properties. Furthermore, there are no existing approaches addressing the adaptation cross all the functional layers of the service based systems with respect to a variety of information discussed earlier. The proposed approaches address either only particular functional layer, or a particular problem e.g. identification mismatch.

The table also shows that in the adaptation research, the trend is toward all-in-one solutions by integrating a maximum of requirements in order to have a complete framework.

Moreover, some other important aspects such as contextual information is thought-worthy. In fact, insignificant investigations have tackled the context when adapting the system, while its application gained more and more the service based systems.

3.4.2 Strategies

Figure 3.2 categories the presented approaches with respect to the adaptation strategies. The table represents the dynamicity of the strategies, the degree of the automation, the methodology used for the adaptation and finally the language and notation used when expressing the strategy.

This table shows that the research on the adaptation strategy has exploited both static and dynamic type. However, the trend is toward the dynamic and automatic adaptation when a change is tackled at run time behavior without any changes in the implementation part.

Furthermore, there are several problems in the way the methodology of the adaptation is handled.In fact, in some projects the direction of the adaptation is not discussed or unclear, due to the fact that the adaptation can be automatic and handled at runtime behavior or in the implementation time.

Moreover, the notations and the languages needed to specify the strategies are wide-ranging. Only few approaches use standards to express the way the adaptation is performed.

3.4.3 Implementation

Figure 3.3 describes the comparison of the presented approaches with respect to the implementation views. The figure shows the timing of performing the adaptation, where it is performed, the way

Approach	**Objective**	**Information**		**Type/Model**		**Location**		**Specification/Association**	
		F/NF	**Scope**	**Type**	**Model**	**Int/Ext**	**Layer**	**Goal/Event**	**Impl/Expl**
IVS	semantics	F	global	comp	behav	ext	BP	goal	expl
PMAS	semantics	F	global	comp	both	-	SC	event	impl
SCBP	semantics	F	global	comp	interf	ext	BP	event	expl
METEOR	configuration	F	local	comp	behav	int	SC/SI	-	impl
CSANF	HCI	NF	-	param	behav	ext	BP/SC	event	-
ECD	performance	F	global	param	interf	int	BP	event	impl
ASFB	performance	F	local	comp	interf	int	BP	goal	expl
PAR	QoS	F	local	param	interf	int	BP	goal	expl
NSLA	SLA	NF	local	param	interf	ext	SC	goal	expl
UDP	user needs	NF	local	comp	interf	-	SI	-	expl
LPV	performance	F	global	param	interf	int	SC	event	expl
DIAM	diagnosis	F	global	param	interf	int	SC/SI	goal	impl
TASOA	mismatch	F	global	param	interf	int	SC/SI	goal	impl
SELF	recovery	F	-	comp	behav	int	BP/SI	event	impl
ALR	learning	F	global	param	interf	int	SC/SI	goal	impl
AGA	mismatch	F	global	comp	behav	int	BP	event	expl
AOPA	mismatch	F	local	param	interf	int	BP/SI	event	expl
SAAI	integration	F	global	param	both	-	BP	event	expl
DAI	integration	F	-	param	interf	-	BP	event	-
RAAOP	SLA	NF	local	comp	behav	int	SC/SI	event	-
ASCE	QoS	F/NF	-	comp	behav	-	SC	goal	-
SCENE	-	F	local	comp	behav	int	SC	event	impl
ASRAI	mismatch	F	local	comp	both	ext	SC	event	impl
ASF	QoS	NF	-	param	-	ext	BP	-	-
PAWS	configuration	F	global	param	interf	ext	BP	goal	impl
BPD	performance	F	-	param	-	-	-	-	-
CMA	QoS/context	F	global	param	interf	int	BP	event	expl

Figure 3.1: Comparing requirements

it is integrated into the application architecture and execution process, and finally the supporting paradigms employed by each approach.

The implementation of the adaptation can hold along the lifetime of the application, ranging from design over deployment, startup time to runtime. But, the adaptation at runtime is more used. We can also combine those results with those from the comparison of the strategies and notice that the applications enabling adaptation during the runtime are classified as dynamic application and those enabling the adaptation at design/compile/starting time are classified as static applications.

The distributed way is more adapted than the centralized manner. In fact, the existing systems performed the adaptation internally and then propagate the changes among the components. Besides, the degree of the integration of the adaptation is sparse, ranging from tightly integrated into the platform over injected to self-adaption. We can also noticed that the self adaptation/healing systems are recently considered as the trend of the new systems for the adaptation.

Furthermore, the table depicts that the computational and component paradigms have been relatively more studied than aspect-oriented programming. Nevertheless, the very recent trend is to separate concerns for both functional and non functional aspects.

Moreover, none of the existing approaches have studied the complexity in the implementation. How much the adaptation is complex at any time of the lifetime of the application according must be addressed. Furthermore, none of existing approaches is related to industry rather to academia.

3.5 Related Work on Adaptation in Information Systems

Due to the constant changes in information systems, the adaptation has been considered crucial. It has been identified and studied in various information system disciplines. Here, we discuss and show how to handle the adaptation in component based software Engineering, in software product line and

Approach	**Dynamicity**	**Degree of automation**	**Methodology**	**Language/Notation**
IVS	dynamic	auto	-	constraint language
PMAS	dynamic	semi-auto	top-down	constraint language
SCBP	dynamic	semi-auto	top-down	DSLs/declarative
METEOR	dynamic	-	-	DSLs/declarative
CSANF	dynamic	semi-auto	backtracking	Constraint language
ECD	static	manually	top-down	BPEL
ASFB	dynamic	auto	backwards	BPEL
PAR	dynamic	auto	top-down	BPEL
NSLA	dynamic	semi-auto	top-down	DSLs/declarative
UDP	dynamic	auto	top-down	DSLs/declarative
LPV	dynamice	manually	-	DSLs
DIAM	dynamic	auto	-	BPEL, DSLs
TASOA	dynamic	auto	-	BPEL, DSLs
SELF	dynamic	auto	-	BPEL,WSCOL,WSRel
ALR	dynamic	auto	top-down	-
AGA	dynamic	auto	top-down	BPEl,Yawl
AOPA	static	semi-auto	top-down	BPEL/aspects
SAAI	static	semi-auto	-	BPEL/procedural
DAI	static	manually	top-down	BPEL
RAAOP	dynamic	on-demand	-	specific language
ASCE	dynamic	manually	backtracking	-
SCENE	dynamic	auto	bottom-up	ECA/declarative
ASRAI	dynamic	semi-auto	top-down	DSLs/declarative
ASF	dynamic	manually	-	DSLs/declarative
PAWS	dynamic	auto	-	OWL,DSLs/declarative
BPD	static	manually	context	DSLs
CMA	dynamic	manually	context	OWL, BPEL

Figure 3.2: Comparing the strategies

parallel programming. We will see, they share the same concerns presented for the SBA.

3.5.1 Adaptation in Component-Based Software Engineering

The problem of building adaptive computing systems has gained dramatically more interest over recent years. The emergence of ubiquitous computing and the growing demand for autonomic computing are the main factors entailing this interest [110]. Ubiquitous computing aims to remove traditional boundaries for how, when, and where humans and computers interact. To do this, computer systems must adapt to its environment of computing platforms and communication networks. Autonomic computing refers to the ability of a system to manage and protect their own resources. Such systems require run-time adaptation in order to survive failures, network outages, and security attacks.

In [111] the authors identify the following groups of reasons for software system adaptation: corrective (remove faulty behavior), adaptive (response to changes affecting the context), extending (extend the system with new functionalities), and perfective (improve characteristics of an application). The authors also classify the adaptation into the following classes: architectural adaptation (affect the structure of the system), implementation adaptation (affect implementation of the components without changing the interface), interface adaptation (affect the interfaces of the components), geography adaptation (affect distribution of the components over the network). Orthogonally to this, adaptation approaches in [110] are classified into parameter adaptation, where the variables that determine the system behavior are affected, and the composition adaptation, where the structural parts of the system are changed.

The rapid growth in the area of adaptation in software engineering is explained by a set of technological reasons [110]. Separation of concerns, computational reflection, and component-based design provided programmers with the tools to construct adaptable system in a systematic way, while widespread use of middleware provided a place to locate and enable adaptive behavior. These tech-

Approach	**Timing**	**Location**		**Techniques**
		Topology	**Degree/Integration**	
IVS	design/runtime	central	injected	computational
PMAS	design	central	tightly-integ	computational
SCBP	design/runtime	distributed	injected	computational
METEOR	design/deploy	distributed	tight-integrated	component
CSANF	runtime	distributed	injected	computational
ECD	design	distributed	tightly-integ	computational
ASFB	runtime	central	tightly-integ	computational
PAR	design	distributed	tightly-integ	computational
NSLA	runtime	distributed	injected	component
UDP	design	central	-	computational
LPV	design/runtime	distributed	tightly	computational
DIAM	runtim	distributed	self-adaptation	computational
TASOA	runtime	distributed	self-adaptation	computational
SELF	design	-	self-adaptation	AOP
ALR	runtime	-	self-adaptation	computational
AGA	runtime	distributed	self-adaptation	computational
AOPA	design	central	tightly-integrated	AOP
SAAI	runtime	central	injected	component
DAI	runtime	distributed	injected	computational
RAAOP	runtime	distributed	self-adapta	AOP
ASCE	Deploy/runtime	distributed	injected	computational
SCENE	design/deployment/runtime	distributed	injected	component
ASRAI	runtime	distributed	injected	component
ASF	design	distributed	-	-
PAWS	design/runtime	distributed	self-adaptation	computational
BPD	design/runtime	-	-	component
CMA	design/runtime	distributed	injected	component

Figure 3.3: Comparing the implementation

nologies, combined in different ways, lead to the development of a wide range of application adaptation approaches and principles [110, 112].

Separation of concerns provides a way to separate development of the functionality and the crosscutting concerns (e.g., quality of service, security). This principle has become one of the cornerstone principle in software engineering, and has lead to a wide spread od aspect-oriented programming (AOP) approach [113]. AOP supports adaptation in several ways. First, many adaptations are relative to some crosscutting concern (e.g., quality-of-service) and therefore AOP may be used to define and implement this concern. Second, it permits delaying the modification of the system to run-time, making adaptation more flexible and dynamic.

Computational reflection refers to the ability of a program to reason about, and possibly alter, its own behavior. Reflection enables a system to reveal (selected) details of its implementation without compromising portability. It comprises two activities: introspection (enables an application to observe its own behavior) and intercession (enables a system or application to act on the observations and modify its own behavior). Together with AOP, it allows for observing and reasoning on the system behavior, enabling its run-time modification.

Component-based design comes with well-defined interfaces, providing a way to develop separately providers and consumers independently, and, therefore, promoting component re-use. We remark that this technology was further advanced by the service-oriented architecture providing even better de-coupling, interoperability and re-use of the underlying services (components).

Middleware is a set of services that separate applications from operating systems and protocols. These services include high-level programming abstractions, different aspects (QoS, security, fault tolerance, persistence, transactionality), and specific functionalities. Since middleware provides an abstraction of many adaptation-related concerns, it serves as good place for implementing adaptation mechanisms.

The above technologies and principles has many similarities with the service-oriented architectures and technologies for the SBA development. Not surprisingly, a wide range of adaptation approaches for SBAs adopt similar concepts as the ones for component-based software systems, such as aspect-oriented approaches, use of middleware for realizing adaptation, etc.

3.5.2 Adaptation in Software Product Line Engineering

Software product line engineering (SPLE [114, 115]) has proven to be the paradigmfor developing a diversity of similar software applications and software-intensivesystems at low costs, in short time, and with high quality. Numerous reports document thesignificant achievements of introducing software product lines in industry [115].Key to SPLE is to define and realize the commonality and the variability of the productline and its applications. The commonalities comprise the artifacts and the propertiesthat are shared by all product line applications. The variability defines how the variousapplications derived from the product line can vary. A prerequisite for managing softwareproduct line variability is the explicit documentation of the variability.

A Framework for Software Product Line Engineering

The SPLE framework depicted in Figure3.4 illustrates the two product line engineering: processes domain engineering and application engineering. The framework has been developed in the context of European SPLE research projects ESAPS, CAFE, and FAMILIES [116]. The domain engineering process is responsible for defining the commonality and the variability of the applications of the product line [117]. Furthermore, the do-main artifacts are realized which implement the commonalities and provide the variability required to derive the set of intended applications. The domain artifacts constitute the product line platform and include, among others, requirements models (e.g., use case diagrams), architectural models (e.g., component or class diagrams) and test models. The application engineering process is responsible for deriving applications from the domain artifacts. Application engineering exploits the variability of the domain artifacts by binding (resolving) variability according to the requirements defined for the particular application.

By splitting the overall development process into do-main engineering and application engineering a separation of the two concerns building a robust product line platform and creating individual, customer or market specific applications is established.

Two Approaches for Modelling Variability

To model the variability of a product line, two principle types of approaches are proposed in the literature. One type of approaches proposes to integrate variability information into existing models. For example, extensions for UML models by defining stereotypes for product line variability are proposed (e.g., see [118]), or feature models are extended to facilitate the documentation of variability information (e.g., FORM [119], CBFM [120]). The other type of approaches proposes employing a dedicated variability model, i.e. those approaches argue that variability should not be integrated into existing models, but defined separately. Among, others, the Orthogonal Variability Model (OVM, [115]) has been proposed for documenting software product line variability in a dedicated model. In a dedicated variability model only the variability of the product line is documented (independent of its realization in the various product line artifacts). The variability elements (Figure3.5)in a dedicated variability model are, in addition, related to the elements in the traditional conceptual models which "realize" the variability defined by the variability model.

In a dedicated variability model, at least the following information is documented:

- Variation Point ("what does vary?"): This documents a variable item or a variable property of an item.
- Variant ("how does it vary?"): This documents the possible instances of a variation point.

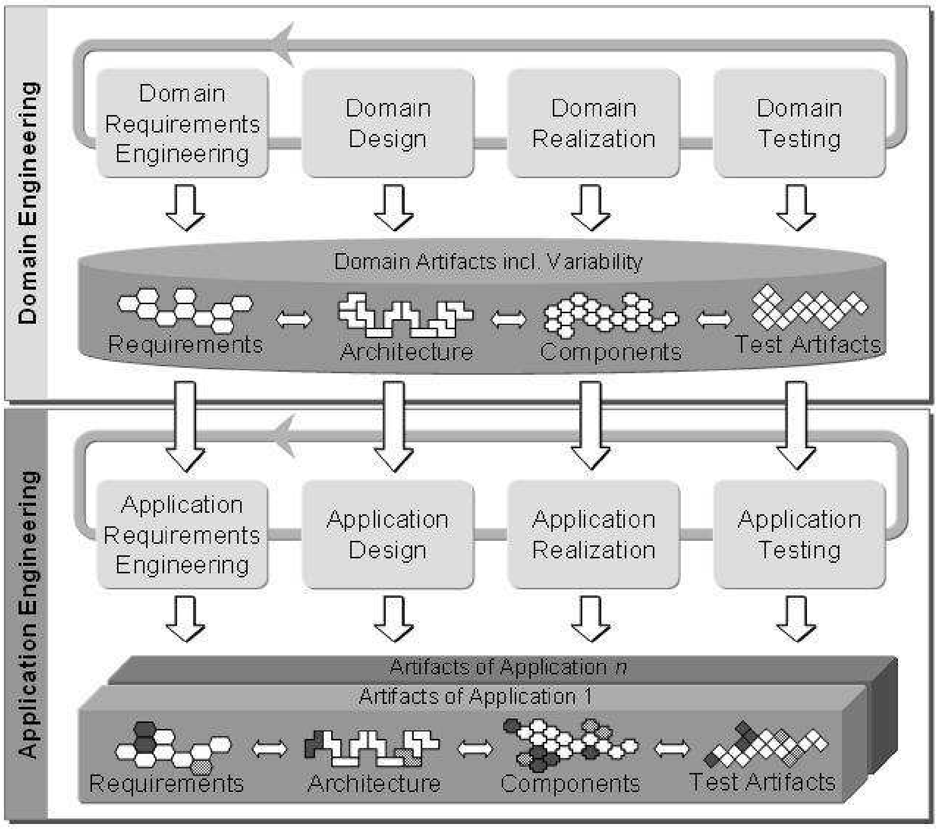

Figure 3.4: SPLE Framework (simplified version of the one in [115])

- Variability Constraints: There can be constraints on variability, because product management decided, e.g., not to offer certain combinations of variants in an application or because the realization of one variant requires another variant to be present.

A Variability Modeling Method for Adaptable Services in Service-Oriented Computing [121]

The authors observe that techniques for modeling service-based systems mainly focus on the business process and the service composition layer. However, those techniques do not satisfactorily address how to model the flexibility which is required by business processes and services. In particular the authors state that the variability of business processes and services is not explicitly modeled.

According to the authors, implementing adaptive service-based systems requires understanding the variability of the business processes and of the environment in which the service-based system will execute. This means that the range of variants which the future service-based system will cover should already be defined during development time.

The contribution proposed to model the variability of a service-based system using principles and methods from the field of software product line engineering. These include the explicit modeling of commonalities and the variability of a software product line. The variability of a software product line is modeled explicitly by variation points and variants. The authors observe that an adaptive service-based system can have different types of variability, which can be found at the following layers of a service-based system:

- Business Process Layer: This layer includes the definition of business processes from the customer's perspective. The authors use BPMN (the Business Process Modeling Notation) to model the business processes. The required flexibility in the sequence of activities of a business process is called Workflow Variability by the authors.

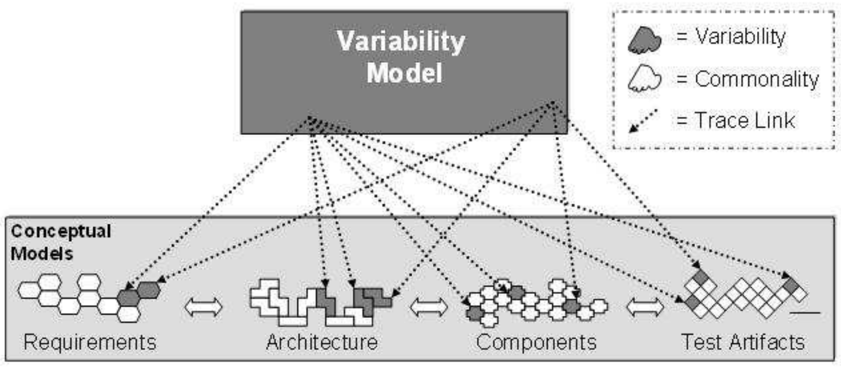

Figure 3.5: Variability Model and relationship to existing conceptual models

- Unit Service Layer: This layer includes the definition of service units from an architect's perspective, which are necessary for handling a business process. The dynamic composition of services is called Composition Variability and it is modeled using BPEL (The Business Process Execution Language).
- Service Interface Layer: This layer includes the definition of service interfaces which reflect the requirements of the service units. The interfaces are described by WSDL (The Web Service Description Language). If alternative interfaces are defined this is called Interface Variability.
- Service Component Layer: This layer includes the implementations of services (according to their interfaces). Variations in the implementation of an interface are called Logic Variability.

For a description of web services and their composition, the authors use WSDL and BPEL. Both concepts employ XML as description language. The respective XML schemas thus are expanded to describe additional properties and composition possibilities of services. The already presented variability at Unit Service Level and Service Interface Level are inserted into the relevant XML schema with the identifier VType. In this way in BPEL variability is introduced in the XML schema via the BPEL "extension" tag.

By determining the variability types and simple XML-based description of variability, the variability of an adaptive service-based system can be explicitly documented.

The proposed approach is demonstrated by an abstract of an XML-descriptions which shows how variability can be described in XML. The example shows the application of the approach for the Unit Service Layer and Service Interface Layer.

Using Product Line Techniques to Build Adaptive Systems [122]

According to the authors, increasingly distributed systems have to be adapted to changing user needs and context changes during run-time, while ensuring that there are no substantial disruptions of the current operation of the service-based application. As the authors claim, this requires that potential variations in the configuration of the system (in terms of components) must already be identified during development of the application, such that the adaptation of the system will be successful during run-time.

For designing highly flexible distributed adaptive systems, the authors propose to employ techniques from SPLE. Similar to SPLE, where two distinct development processes (domain engineering and application engineering) exist, the authors separate between common architectural elements and variable components. This differentiation is made at the conceptual level and the variation points and their variants are documented in so-called Blueprint Plans and Composition Plans respectively. A

Blueprint Plan - from the architect's perspective - documents an atomic component, which can consist of several implementation classes that realize this component. As a consequence, a blueprint plan documents the variation with respect to the implementation of a component. In a Composition Plan, a component is described that is composed of other components. The variability in such a (partial) component architecture is expressed by the concept of a "role". Roles - as proposed by the authors - specify the functionality of each single component and its possible interactions with other components. Unfortunately, the authors do not provide further details on the concept of roles. In addition to the plans, the following kinds of specifications are introduced: Connection Spec, Port Delegation Spec and Node Deployment Spec. These specifications describe compositions, invocation sequences, instantiation of components and their variants in more detail. However, the authors do not explain how to select a component variant. In order to determine at run-time whether an elected composition fulfils the requirements of the system environment, the following cascade of actions is executed (see figure3.6). First, a so-called Context Manager evaluates the information from the system environment and the plans (i.e., the blueprint and composition plans) of the active components and forwards the result to the so-called Adaptation Manager. The Adaptation Manager decides whether a new plan of component composition is necessary. If required, a so-called Configurator carries out the adjustment steps as instantiation, binding, separation or removal of components.

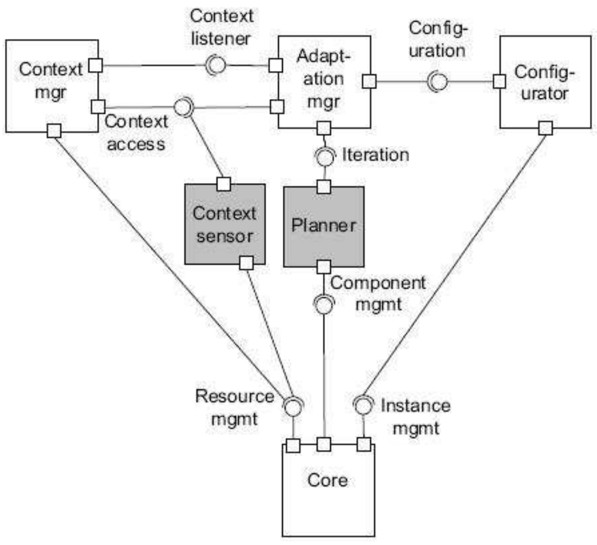

Figure 3.6: Platform architecture

The proposed solution is not applicable for situations that require dealing with unpredictable changes. This is due to the fact that in order to adequately respond to such kinds of changes, the developers have to explicitly describe these changes and develop associated component plans before the service-based application is deployed. After realization and registration of this variant, it can also be invoked or replaced by another variant at runtime. Currently, the proposed approach does not support handling dependencies between variation points and variants.

The proposed conceptual building blocks (blueprint plan, composition plan, etc.) support the developers in explicitly modelling variable characteristics of components and support the developers in managing variation during design and implementation of adaptive systems.

The approach has been applied to a simple example from the network area, where the focus was on how to deal with variable bandwidth and availability of mobile data services. The proposed solution, applied to this example, was able to provide the required adaptation capabilities during run-time. Further, the approach has been used in two industrial projects.

Supporting Runtime System Adaptation through Product Line Engineering and Plug-in Techniques [123])

According to the authors, existing adaptation approaches either focus on the context-, contract- or goal-driven adaptation of applications. For the end-users, there is hardly any support for user-specific adaptations of their software during run-time. The authors thus see a challenge in devising an intuitive and easy to handle approach for the user-driven adaptation of applications.

The contribution proposes to integrate of techniques for variability modeling from the area of software product line engineering (SPLE) and plug-in techniques for extension, adaptation and evolution of software systems. This integrated approach is designed to support users in adapting their systems to their individual work situations. This support is enabled through wizards. The wizards allow a user to select the desired working aids from a digital question catalog. The working aids which are selected are linked to an internal variability model. This variability model in turn is connected with a selection of the plug-ins that implement the variants. Thereby, an application can be (re-)configured based on a user's intentions. The variability model describes the scope of the adaptation in form of included and excluded components, which represent the technical implications of adaptation decisions. That means that only components can be uploaded, which are linked to the variability model and representing “known” variants. With the help of the plug-in concept, it is possible to dynamically upload components in the run-time environment, and thereby allow the integration of new components into the running system.

The benefits of the approach are on the one hand, that it enables end-users to (re-)configure their working area according to their current needs. On the other hand, the link between variant selection and system (re-)configuration is established by integrating two well-established concepts. The integration of those concepts allows user-driven adaptation of an application at run-time.

The approach has been implemented as follows: The variability model and decision model have been realized based on the DOPLER tool suite (Decision-Oriented Product Line Engineering for effective Reuse, see [124]. The decision for the plug-in platform was made for .NET technologies (assembly with explicitly specified extensions and slots, deployment). The approach has been developed in cooperation with a company and has been tailored according to the business cases of this company. The application scenarios are e.g. on-the-fly product adjustments for sales, targeted software upgrades, configuration analysis in productive systems, role-specific system configurations and tailored on-line training programs for learners.

Dealing with Changes in Service-Oriented Computing Through Integrated Goal and Variability Modelling [125]

According to the authors, dealing with changes in service-oriented systems requires the following information:

- knowledge about the rationales for decisions;
- understanding the alternatives;
- traceability between stakeholders' goals and technical realization elements.

In addition they observe that very flexible and adaptable systems cannot be fully specified in advance and thus postulate that design methods and modelling techniques are needed in order to describe flexible and evolvable systems as far as possible in a declarative way.

The approach proposes integrating goal modelling and variability modelling techniques. This is done by identifying variability aspects in i^* goal models and by proposing an approach for mapping i^* models to variability models. Those variability models in turn are then refined and employed to support monitoring and adaptation of service-oriented systems.

The proposed approach focuses on the following two “directions” of changes:

1. Top-down stakeholder-driven changes: A new requirement affects existing services or service architecture. Maybe negotiations with 3rd-party service providers are needed and new SLAs are "signed". The new decisions have to be considered within the service architecture through reconfiguration.

2. Bottom-up monitoring-driven changes: Monitoring results demand changes within the current service configuration to fulfil SLA directives.

In order to deal with both kinds of changes, the authors state that the following facts need to be considered:

- common and specific (individual) goals of different stakeholders;
- knowledge about the current service configuration;
- information about alternative service configurations;
- traceability information to handle dependencies between goals, service types, services and service instances (see definitions below);
- kind of representation of the system at different levels to understand the assignment of stakeholders' needs down to the current and alternative system configurations.

The proposed approach allows modelling these different facts together with traceability and variability information to support system adaptation. In order to map between these facts, the authors propose integrating the meta-models of i^* and the variability modelling language used.

To map concrete i^* models to variability models, the authors have identified six different types of variability (e.g. softgoal and instance variability) and show how these can be specified with the different i^* modelling constructs and map to variability models.

The mapping between i^* models and variability models is supported by the DOPLER tool suite. The Variability management engine (VME) of this tool is used to manage the variability model of available service instances, services, and goals together with the relevant traceability information. Additionally, the engine computes the impact of changes based on the monitoring results or based on user-triggered adaptations. The Monitoring component of the tool receives its monitoring parameters from the VME and offers additionally runtime parameters (e.g. response time, service-life-cycle). The Adaptor component of the tool ultimately performs the requested updates of the service-oriented system.

The approach presents how a variability model can be derived from an i^* goal model and how this variability - derived from the goal model - can be employed to support the monitoring as well as adaptation of service-based applications.

The approach is illustrated by two scenarios demonstrating a top-down-change and a bottom-up-change in a virtual travel booking system. It is illustrated, how a change has to be considered on the different layers (e.g. goal definition, variability modelling, service implementation, monitoring and adaptation mechanism).

3.5.3 Dynamic Adaptation for Parallel Programs

Parallel and distributed programming can be used as a basis to build single high performance services. In this case the design of a High Performance service is a "flavour" of parallel programming because the service is implemented as a parallel program. This seems to be the simplest case.

With this view in mind, the composition of High Performance services is not so easy because if the interface of a service is implemented within a single node of the system, all requests will be serialized between these services, thus the composition of parallel services have no chance to guarantee the same level of performance. In this case, one can reuse the "parallel component" based approaches for Grid computing such as GRID CCM [126] allowing efficient communication and described in more details in deliverable PO-JRA 2.3.1. Within such a component model, the internal logic of the service can be self-adaptable by using the infrastructure facilities such as load balancing and scheduling.

Another view of parallel and distributed Grid computing is to see a parallel (distributed) program as a special composition of services. In this case, some tools designed for parallel programming can be used or customized for orchestration and choreography of services. Things are not so simple because parallel programs communication model is message passing model and not based on client/server approaches,so quite far from service oriented compositions.

However, in GRID computing some attempts to design self-adaptable software have been done independently of the communication model, thus can be reused for services : ASSIST [127] and Dynaco [128]

3.6 Observations

3.6.1 Key challenges

If Web services technology shall be applied to a broad spectrum of applications, it will have to be significantly improved with respect to some capabilities. In particular, in dynamic environments it must be possible to quickly implement and deploy new services, to enable ad-hoc modifications of single service instances at runtime (e.g., to add, delete or shift service steps), and to support evolution that might occur in the environment. These requirements must be met without affecting service consistency and by preserving the robustness of the applications.

There are many research issues concerning adaptation that users will further benefit from the advances:

- The integration and verification of domain knowledge will help in the development of adaptive service management technology. It is necessary to design a framework for defining semantic constraints over services in such a way that they can express real-world domain knowledge on the one hand and are still manageable concerning the effort of adaptation.

- Providing correctness criteria to decide how to adapt web services.

- Identification of actual differences between the (interface and protocol) specifications of a given pair of Web services, and the composition of adapter template solutions for resolving mismatches between protocols to generate the adapter code.

- Another interesting issue is to investigate a framework for automatic adaptation that accommodates hybrid constraints. In this case, a possible solution is to study a way to combine solvers. This will provide more flexibility and let users concentrate on requirements rather than on implementation.

3.6.2 Conclusion and research perspectives

This survey is motivated by the problem of identifying and analyzing approaches for adaptation in the context of Web services. This problem has become extremely important in the age of the world-wide web, Internet connectivity, and globalisation, but also due to the numerous architectural decisions to be made and their impact on quality goals. The importance of the problem was, for example, recognized and emphasized by many researchers and organizations, which consider design methodologies and maintenance for advanced service-oriented applications as a main challenge problem. As a result, there have been numerous efforts by leading research groups to address this problem. All this is to say that the general motivation for the problem of adaptation is widely accepted as being very strong and of recent interest.

Current approaches for adaptability of service based applications mostly focus on the infrastructural layer or on specific aspects of the behaviour of service compositions, such as binding and enactment. Comprehensive approaches covering higher-level aspects of the service composition and coordination layer and the business process management layer are still to be defined. So far, no integrated cross-layer approach to adaptation has been devised. In service based applications, adaptation can invoke different services. Traditionally, these services are web-services that deliver computational and informational services to users. More recently, declarative user-interface definition languages and tools can enable automatic user interface development based on invoked services, thus enabling coherent adaptation of both the application behaviour and its user interface. Several emerging standards and technologies provide important input to the integration of research activities in user-level adaptability.

Bibliography

[1] N. Delgado, A. Q. Gates, and S. Roach, “A Taxonomy and Catalog of Runtime Software-Fault Monitoring Tools,” *IEEE Trans. Softw. Eng.*, vol. 30, no. 12, pp. 859–872, 2004.

[2] B. Plattner and J. Nievergelt, “Monitoring Program Execution: A Survey,” *IEEE Computer*, vol. 14, no. 11, pp. 76–93, 1981.

[3] C. Ghezzi and S. Guinea, “Run-Time Monitoring in Service-Oriented Architectures,” in *Test and Analysis of Web Services*, L. Baresi and E. D. Nitto, Eds. Springer, 2007, pp. 237–264.

[4] D. Bianculli, R. Jurca, W. Binder, C. Ghezzi, and B. Faltings, “Automated Dynamic Maintenance of Composite Services based on Service Reputation,” in *Service-Oriented Computing - ICSOC 2007, Fifth International Conference*, 2007.

[5] M. Pistore and P. Traverso, “Assumption-Based Composition and Monitoring of Web Services,” in *Test and Analysis of Web Services*, L. Baresi and E. D. Nitto, Eds. Springer, 2007, pp. 307–335.

[6] F. Barbon, P. Traverso, M. Pistore, and M. Trainotti, “Run-Time Monitoring of Instances and Classes of Web Service Compositions,” in *IEEE International Conference on Web Services (ICWS 2006)*, 2006, pp. 63–71.

[7] D. Bianculli and C. Ghezzi, “Monitoring Conversational Web Services,” in *IW-SOSWE’07*, 2007.

[8] L. Baresi, C. Ghezzi, and S. Guinea, “Smart Monitors for Composed Services,” in *Service-Oriented Computing - ICSOC 2004, Second International Conference*, 2004, pp. 193–202.

[9] L. Baresi and S. Guinea, “Towards Dynamic Monitoring of WS-BPEL Processes,” in *Service-Oriented Computing - ICSOC 2005, Third International Conference*, 2005, pp. 269–282.

[10] L. Baresi, S. Guinea, and P. Plebani, “WS-Policy for Service Monitoring,” in *Technologies for E-Services, 6th International Workshop, TES 2005*, 2005, pp. 72–83.

[11] L. Baresi, D. Bianculli, C. Ghezzi, S. Guinea, and P. Spoletini, “A Timed Extension of WS-CoL,” in *2007 IEEE International Conference on Web Services (ICWS 2007)*, 2007, pp. 663–670.

[12] L. Ardissono, R. Furnari, A. Goy, G. Petrone, and M. Segnan, “Fault Tolerant Web Service Orchestration by Means of Diagnosis,” in *Software Architecture, Third European Workshop, EWSA 2006*, 2006, pp. 2–16.

[13] S. Benbernou, H. Meziane, and M.-S. Hacid, “Run-Time Monitoring for Privacy-Agreement Compliance,” in *Service-Oriented Computing - ICSOC 2007, Fifth International Conference*, 2007, pp. 353–364.

[14] G. Spanoudakis and K. Mahbub, "Requirements Monitoring for Service-Based Systems: Towards a framework based on Event Calculus," in *19th IEEE International Conference on Automated Software Engineering (ASE 2004), 20-25 September 2004, Linz, Austria*, 2004, pp. 379–384.

[15] K. Mahbub and G. Spanoudakis, "Run-time Monitoring of Requirements for Systems Composed of Web Services: Initial Implementation and Evaluation Experience," in *2005 IEEE International Conference on Web Services (ICWS 2005)*, 2005, pp. 257–265.

[16] ——, "Monitoring WS-Agreements: An Event Calculus-Based Approach," in *Test and Analysis of Web Services*, L. Baresi and E. D. Nitto, Eds. Springer, 2007, pp. 265–306.

[17] G. Spanoudakis, C. Kloukinas, and K. Androutsopoulos, "Towards security monitoring patterns," in *Proceedings of the 2007 ACM Symposium on Applied Computing (SAC)*, 2007, pp. 1518–1525.

[18] C. Kloukinas and G. Spanoudakis, "A Pattern-Driven Framework for Monitoring Security and Dependability," in *Trust, Privacy and Security in Digital Business, 4th International Conference, TrustBus*, 2007, pp. 210–218.

[19] A. Farrell, M. Sergot, C. Bartolini, M. Salle, D. Trastour, and A. Christodoulou, "Using the Event Calculus for the Performance Monitoring of Service-Level Agreements for Utility Computing," in *Proceedings of First IEEE International Workshop on Electronic Contracting (WEC 2004)*, 2004.

[20] A. Lazovik, M. Aiello, and M. P. Papazoglou, "Associating Sssertions with Business Processes and Monitoring their Execution," in *Service-Oriented Computing - ICSOC 2004, Second International Conference*, 2004, pp. 94–104.

[21] A. Sahai, V. Machiraju, M. Sayal, A. P. A. van Moorsel, and F. Casati, "Automated SLA Monitoring for Web Services," in *13th IFIP/IEEE International Workshop on Distributed Systems: Operations and Management, DSOM 2002*, 2002, pp. 28–41.

[22] A. Keller and H. Ludwig, "The WSLA Framework: Specifying and Monitoring Service Level Agreements for Web Services," *J. Network Syst. Manage.*, vol. 11, no. 1, 2003.

[23] H. Ludwig, A. Dan, and R. Kearney, "Cremona: An Architecture and Library for Creation and Monitoring of WS-Agreements," in *Service-Oriented Computing - ICSOC 2004, Second International Conference*, 2004, pp. 65–74.

[24] F. Curbera, M. J. Duftler, R. Khalaf, W. Nagy, N. Mukhi, and S. Weerawarana, "Colombo: Lightweight Middleware for Service-Oriented Computing," *IBM Systems Journal*, vol. 44, no. 4, pp. 799–820, 2005.

[25] C. Beeri, A. Eyal, T. Milo, and A. Pilberg, "Monitoring Business Processes with Queries," in *Proceedings of the 33rd International Conference on Very Large Data Bases*, 2007, pp. 603–614.

[26] C. Momm, R. Malec, and S. Abeck, "Towards a Model-driven Development of Monitored Processes," *Wirtschaftsinformatik*, vol. 2, 2007.

[27] P. Chowdhary, K. Bhaskaran, N. S. Caswell, H. Chang, T. Chao, S.-K. Chen, M. Dikun, H. Lei, J.-J. Jeng, S. Kapoor, C. A. Lang, G. Mihaila, I. Stanoi, and L. Zeng, "Model Driven Development for Business Performance Management," *IBM Syst. J.*, vol. 45, no. 3, pp. 587–605, 2006.

[28] H. Roth, J. Schiefer, and A. Schatten, "Probing and Monitoring of WSBPEL Processes with Web Services," in *CEC-EEE '06: Proceedings of the The 8th IEEE International Conference on E-Commerce Technology and The 3rd IEEE International Conference on Enterprise Computing, E-Commerce, and E-Services*, 2006, p. 30.

[29] M. Castellanos, F. Casati, M.-C. Shan, and U. Dayal, "iBOM: A Platform for Intelligent Business Operation Management," in *ICDE '05: Proceedings of the 21st International Conference on Data Engineering*, 2005, pp. 1084–1095.

[30] J.-J. Jeng, J. Schiefer, and H. Chang, "An Agent-based Architecture for Analyzing Business Processes of Real-Time Enterprises," in *EDOC '03: Proceedings of the 7th International Conference on Enterprise Distributed Object Computing*, 2003, p. 86.

[31] W. M. P. van der Aalst and M. Pesic, "Specifying and Monitoring Service Flows: Making Web Services Process-Aware," in *Test and Analysis of Web Services*, L. Baresi and E. D. Nitto, Eds. Springer, 2007, pp. 11–55.

[32] C. W. Günther and W. M. P. van der Aalst, "Fuzzy Mining - Adaptive Process Simplification Based on Multi-perspective Metrics," in *Business Process Management, 5th International Conference, BPM*, 2007, pp. 328–343.

[33] A. Rozinat and W. M. P. van der Aalst, "Conformance Checking of Processes Based on Monitoring Real Behavior," *Inf. Syst.*, vol. 33, no. 1, pp. 64–95, 2008.

[34] B. F. van Dongen, A. K. A. de Medeiros, H. M. W. Verbeek, A. J. M. M. Weijters, and W. M. P. van der Aalst, "The ProM Framework: A New Era in Process Mining Tool Support," in *Applications and Theory of Petri Nets 2005, 26th International Conference, ICATPN*, 2005, pp. 444–454.

[35] W. M. P. van der Aalst and A. K. A. de Medeiros, "Process Mining and Security: Detecting Anomalous Process Executions and Checking Process Conformance," *Electr. Notes Theor. Comput. Sci.*, vol. 121, pp. 3–21, 2005.

[36] H. R. M. Nezhad, R. Saint-Paul, B. Benatallah, and F. Casati, "Deriving Protocol Models from Imperfect Service Conversation Logs," *IEEE Transactions on Knowledge and Data Engineering (TKDE)*, 2008, to appear.

[37] D. Devaurs, K. Musaraj, F. De Marchi, and M.-S. Hacid, "Timed Transition Discovery from Web Service conversation Logs," in *20th International Conference on Advanced Information Systems Engineering (CAISE'08)*, 2008.

[38] I. Foster, C. Kesselman, and S. Tuecke, "The Anatomy of the Grid: Enabling Scalable Virtual Organizations," *Internat. J. Supercomput. Appl.*, vol. 15, no. 3, pp. 200–222, 2001.

[39] "WS-Resource Framework. http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=wsrf."

[40] B. Tierney, W. Johnston, B. Crowley, G. Hoo, C. Brooks, and D. Gunter, "The NetLogger Methodology for High Performance Distributed Systems Performance Analysis," in *7th IEEE International Symposium on High Performance Distributed Computing (HPDC'98)*, 2998, pp. 260–267.

[41] R. Wolski, N. T. Spring, and J. Hayes, "The Network Weather Service: a Distributed Resource Performance Forecasting Service for Metacomputing," *Future Generation Computer Systems*, vol. 15, no. 5–6, pp. 757–768, 1999.

[42] S. Andreozzi, "GLUE Schema Implementation for the LDAP Model," INFN, Tech. Rep., May 2003. [Online]. Available: http://www.cnaf.infn.it/~sergio/publications/Glue4LDAP.pdf

[43] B. Tierney, R. A. Aydt, D. Gunter, W. Smith, V. Taylor, R. Wolski, and M. Swany, "A Grid Monitoring Architecture," Global Grid Forum, Informational Document GFD-I.7, January 2002.

[44] "Global Grid Forum. http://www.ggf.org."

[45] H.-L. Truong and T. Fahringer, "SCALEA-G: a Unified Monitoring and Performance Analysis System for the Grid," *Scientific Programming*, vol. 12, no. 4, pp. 225–237, November 2004, axGrids 2004 Special Issue.

[46] I. Foster, H. Kishimoto, A. Savva, D. Berry, A. Djaoui, A. Grimshaw, B. Horn, F. Maciel, F. Siebenlist, R. Subramaniam, J. Treadwell, and J. von Reich, "The Open Grid Services Architecture, Version 1.0," Global Grid Forum, Tech. Rep. GFD-I.030, January 2005. [Online]. Available: http://www.ggf.org/documents/GFD.30.pdf

[47] I. Foster, C. Kesselman, G. Tsudik, and S. Tuecke, "A Security Architecture for Computational Grids," in *Proceedings of the 5th ACM conference on Computer and communications security*, 1998, pp. 83–92.

[48] "XPath specification. http://www.w3.org/TR/xpath/."

[49] B. Buck and J. K. Hollingsworth, "An API for Runtime Code Patching," *The International Journal of High Performance Computing Applications*, vol. 14, no. 4, pp. 317–329, November 2000.

[50] K. Czajkowski, S. Fitzgerald, I. Foster, and C. Kesselman, "Grid Information Services for Distributed Resource Sharing," in *10th IEEE International Symposium on High Performance Distributed Computing (HPDC-10 2001)*, 2001, pp. 181–194.

[51] "The Globus Toolkit. http://www.globus.org."

[52] "The OpenLDAP project. http://www.openldap.org."

[53] X. Zhang and J. M. Schopf, "Performance Analysis of the Globus Toolkit Monitoring and Discovery Service, MDS2," in *Proceedings of the International Workshop on Middleware Performance (MP 2004)*, 2004.

[54] X. Zhang, J. L. Freschl, and J. M. Schopf, "A Performance Study of Monitoring and Information Services for Distributed Systems," in *12th International Symposium on High-Performance Distributed Computing (HPDC-12 2003)*, 2003, pp. 270–282.

[55] "WS-Notification. http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=wsn."

[56] S. Fisher, "Relational Model for Information and Monitoring," Global Grid Forum, Tech. Rep. GWD-GP-7-1, 2001.

[57] A. Cooke, W. Nutt, J. Magowan, P. Taylor, J. Leake, R. Byrom, L. Field, S. Hicks, M. Soni, A. Wilson, R. Cordenonsi, L. Cornwall, A. Djaoui, S. Fisher, N. Podhorszki, B. Coghlan, S. Kenny, D. O'Callaghan, and J. Ryan, "Relational Grid Monitoring Architecture (R-GMA)," in *Proceedings of UK e-Science All Hands Meeting*, 2003, pp. 744–750.

[58] "The European DataGrid project. http://www.eu-datagrid.org."

[59] M. A. Baker and M. Grove, “jGMA: A Lightweight Implementation of the Grid Monitoring Architecture,” Distributed Systems Group, University of Portsmouth,” Technical Report, April 2004.

[60] H. B. Newman, I. Legrand, P. Galvez, and R. Voicu, “MonALISA: A Distributed Monitoring Service Architecture,” in *International Conference on Computing in High Energy Physics (CHEP2003)*, 2003.

[61] J. Waldo, “The Jini Architecture for Network-Centric Computing,” *Communications of the ACM*, vol. 42, no. 7, pp. 76–82, July 1999.

[62] S. Andreozzi, N. D. Bortoli, S. Fantinel, A. Ghiselli, G. L. Rubini, G. Tortone, and M. C. Vistoli, “GridICE: a monitoring service for grid systems,” *Future Generation Computer Systems*, vol. 21, no. 4, pp. 559–571, April 2005.

[63] S. Chakravarthy, “Architectures and Monitoring Techniques for Active Databases: An Evaluation,” *Data Knowl. Eng.*, vol. 16, no. 1, pp. 1–26, 1995.

[64] M. Mansouri-Samani and M. Sloman, “Monitoring Distributed Systems (A Survey),” Imperial College of Science Technology and MedicineImperial College of Science Technology and Medicine, London, Tech. Rep., 1992, research Report No. DOC92/2.

[65] M. zur Muehlen and M. Rosemann, “Workflow-based Process Monitoring and Controlling - Technical and Organizational Issues,” in *HICSS*, 2000.

[66] M. P. Papazoglou, “The Challenges of Service Evolution,” in *Advanced Information Systems Engineering, 20th International Conference, CAiSE*, 2008, pp. 1–15.

[67] A. Bracciali, A. Brogi, and C. Canal, “A Formal Approach to Component Adaptation,” *Journal of Systems and Software*, vol. 74, no. 1, pp. 45–54, 2005.

[68] M. Dumas, M. Spork, and K. Wang, “Adapt or Perish: Algebra and Visual Notation for Service Interface Adaptation,” in *International Conference on Business Process Management*. Springer-Verlag, September 2006, p. 6580.

[69] H. R. M. Nezhad, B. Benatallah, A. Martens, F. Curbera, and F. Casati, “Semi-automated adaptation of service interactions,” in *WWW ’07: Proceedings of the 16th international conference on World Wide Web*. New York, NY, USA: ACM, 2007, p. 9931002.

[70] D. M. Yellin and R. E. Strom, “Protocol Specifications and Component Adaptors,” *ACM TOPLAS*, vol. 19, no. 2, p. 292333, 1997.

[71] X. Xiong and Z. Weishi, “The Current State of Software Component Adaptation,” in *First International Conference on Semantics, Knowledge and Grid*. IEEE Computer Society, 2005.

[72] P. Inverardi, L. Mostarda, M. Tivoli, and M. Autili, “Synthesis of Correct and Distributed Adaptors for Component-Based Systems: an Automatic Approach,” in *20th IEEE/ACM International Conference on Automated Software Engineering (ASE 2005)*. ACM, November 2005, pp. 405–409.

[73] D. Hemer, “A Formal Approach to Component Adaptation and Composition,” in *ACSC ’05: Proceedings of the Twenty-eighth Australasian conference on Computer Science*. Darlinghurst, Australia, Australia: Australian Computer Society, Inc., 2005, p. 259266.

[74] I. Mouakher, A. Lanoix, and J. Souquires, “Component Adaptation: Specification and Verification,” in *11th International Workshop on Component Oriented Programming (WCOP 2006), In Conjunction of ECOOP 2006*, 2006.

[75] W. Jiao and H. Mei, "Automating Integration of Heterogeneous COTS Components," in *9th International Conference on Software Reuse*, ser. Lecture Notes in Computer Science, vol. 4039. Springer, June 2006, pp. 29–42.

[76] W. Kongdenfha, R. Saint-Paul, B. Benatallah, and F. Casati, "An Aspect-Oriented Framework for Service Adaptation," in *International Conference on Service Oriented Computing*. Springer-Verlag, December 2006.

[77] A. Brogi and R. Popescu, "Automated Generation of BPEL Adapters," in *International Conference on Service Oriented Computing*, 2006.

[78] J. Harney and P. Doshi, "Speeding up Adaptation of Web Service Compositions Using Expiration Times," in *WWW '07: Proceedings of the 16th international conference on World Wide Web*. New York, NY, USA: ACM, 2007, p. 10231032.

[79] X. Xie and W. Zhang, "A Framework of Software Component Adaptation," in *Algorithms and Architectures for Parallel Processing, 7th International Conference, ICA3PP*, 2007, pp. 153–164.

[80] G. Chafle, K. Dasgupta, A. Kumar, S. Mittal, and B. Srivastava, "Adaptation in Web Service Composition and Execution," in *ICWS '06: Proceedings of the IEEE International Conference on Web Services*. Washington, DC, USA: IEEE Computer Society, 2006, p. 549557.

[81] S. Padmanabhuni, B. Majumdar, M. Chawla, and U. Mysore, "A Constraint Satisfaction Approach to Non-functional Requirements in Adaptive Web Services," in *NWESP '06: Proceedings of the International Conference on Next Generation Web Services Practices*. Washington, DC, USA: IEEE Computer Society, 2006, p. 109116.

[82] N. C. Narendra, K. Ponnalagu, J. Krishnamurthy, and R. Ramkumar, "Run-Time Adaptation of Non-functional Properties of Composite Web Services Using Aspect-Oriented Programming," in *International Conference on Service-Oriented Computing - ICSOC*, 2007, pp. 546–557.

[83] A. Hibner and K. Zielinski, "Semantic-based Dynamic Service Composition and Adaptation," in *IEEE SCW*, 2007, pp. 213–220.

[84] L. T. Ly, S. Rinderle, and P. Dadam, "Integration and Verification of Semantic Constraints in Adaptive Process Management Systems," *Data Knowl. Eng.*, vol. 64, no. 1, pp. 3–23, 2008.

[85] S. Rinderle and M. Reichert, "A formal framework for adaptive access control models." *J. Data Semantics*, vol. 9, pp. 82–112, 2007.

[86] S. K. Williams, S. A. Battle, and J. E. Cuadrado, "Protocol Mediation for Adaptation in Semantic Web Services," in *2nd European Semantic Web Conference*, 2006, pp. 635–649.

[87] S. Rinderle, M. Reichert, and P. Dadam, "On Dealing With Semantically Conflicting Business Process Changes," University of Ulm, Tech. Rep., 2005. [Online]. Available: http://vts.uni-ulm.de/docs/2005/5342/vts_5342.pdf

[88] K. Verma, K. Gomadam, A. P. Sheth, J. A. Miller, and Z. Wu, "The METEOR-S Approach for Configuring and Executing Dynamic Web Processes," in *Technical report*, 2005.

[89] S. Rinderle, M. Reichert, and P. Dadam, "Evaluation of Correctness Criteria for Dynamic Workflow Changes," in *Business Process Management, 1st International Conference*, 2003, pp. 41–57.

[90] M. U. Reichert and S. B. Rinderle, “On Design Principles for Realizing Adaptive Service Flows with BPEL.” in *Proceedings EMISA 2006 - Methoden, Konzepte und Technologien für die Entwicklung von dienstebasierten Informationssystemen, Hamburg, Germany*, 2006, pp. 133–146.

[91] E. D. Nitto, M. D. Penta, A. Gambi, G. Ripa, and M. L. Villani, “Negotiation of Service Level Agreements: An Architecture and a Search-Based Approach,” in *Service-Oriented Computing - ICSOC 2007, Fifth International Conference, Vienna, Austria, September 17-20, 2007*. Springer, 2007, pp. 295–306.

[92] M. G. Fugini, P. Plebani, and F. Ramoni, “A User Driven Policy Selection Model,” in *Service-Oriented Computing - ICSOC 2006, 4th International Conference, Chicago, IL, USA, December 4-7, 2006, Proceedings*. Springer, 2006, pp. 427–433.

[93] M. Tanelli, D. Ardagna, and M. Lovera, “LPV Model Identification for Power Management of Web Services,” in *In IEEE Multi-conference on Systems and Control, 2008*, 2008.

[94] L. Console and M. Fugini, “The WS-Diamond Team :WS-DIAMOND: an approach to Web Services - DIAgnosability, MONitoring and Diagnosis,” in *e-Challenges Conf. 2007, The Hague, Oct. 2007*, 2007.

[95] G. Denaro, M. Pezzé, D. Tosi, and D. Schilling, “Towards Self-Adaptive Service-Oriented Architectures,” in *TAV-WEB ’06: Proceedings of the 2006 workshop on Testing, analysis, and verification of web services and applications*. New York, NY, USA: ACM, 2006, pp. 10–16.

[96] L. Baresi, S. Guinea, and L. Pasquale, “Self-healing BPEL processes with Dynamo and the JBoss rule engine,” in *ESSPE ’07: International workshop on Engineering of software services for pervasive environments*, 2007, pp. 11–20.

[97] B. Pernici and A. M. Rosati, “Automatic Learning of Repair Strategies for Web Services,” in *In Proceedings of the Fifth European Conference on Web Services (ECOWS 2007) (November 26 - 28, 2007)*, 2007, pp. 119–128.

[98] A. Brogi and R. Popescu, “Automated Generation of BPEL Adapters,” in *Service-Oriented Computing - ICSOC 2006, 4th International Conference, Chicago, IL, USA, December 4-7, 2006*, 2006, pp. 27–39.

[99] W., R. Saint-Paul, B. Benatallah, and F. Casati, “An Aspect-Oriented Framework for Service Adaptation,” in *Service-Oriented Computing - ICSOC 2006, 4th International Conference, Chicago, IL, USA, December 4-7, 2006*. Springer, 2006, pp. 15–26.

[100] H. R. M. Nezhad, B. Benatallah, A. Martens, F. Curbera, and F. Casati, “Semi-automated Adaptation of Service Interactions,” in *Proceedings of the 16th International Conference on World Wide Web, WWW 2007, Banff, Alberta, Canada, May 8-12,*. ACM Press, 2007, pp. 993–1002.

[101] B. Benatallah, F. Casati, D. Grigori, H. R. M. Nezhad, and F. Toumani, “Developing Adapters for Web Services Integration,” in *Advanced Information Systems Engineering, 17th International Conference, CAiSE 2005, Porto, Portugal, June 13-17, 2005, Proceedings*, 2005, pp. 415–429.

[102] G. Chafle, K. Dasgupta, A. Kumar, S. Mittal, and B. Srivastava, “Adaptation inWeb Service Composition and Execution,” in *International Conference on Web Services - ICWS*, 2006, pp. 549–557.

[103] M. Colombo, E. D. Nitto, and M. Mauri, "SCENE: A Service Composition Execution Environment Supporting Dynamic Changes Disciplined Through Rules," in *Service-Oriented Computing - ICSOC 2006, 4th International Conference, Chicago, IL, USA, December 4-7, 2006*. Springer, 2006, pp. 191–202.

[104] M. D. Penta, R. Esposito, M. L. Villani, R. Codato, M. Colombo, and E. D. Nitto, "WS Binder: a Framework to Enable Dynamic Binding of Composite Web Services," in *SOSE '06: Proceedings of the 2006 international workshop on Service-oriented software engineering*. New York, NY, USA: ACM, 2006, pp. 74–80.

[105] L. Cavallaro and E. D. Nitto, "An Approach to Adapt Service Requests to Actual Service Interfaces," in *SEAMS '08: Proceedings of the 2008 international workshop on Software engineering for adaptive and self-managing systems*. New York, NY, USA: ACM, 2008, pp. 129–136.

[106] D. Ardagna and B. Pernici, "Adaptive Service Composition in Flexible Processes," *IEEE Trans. Software Eng.*, vol. 33, no. 6, pp. 369–384, 2007.

[107] D. Ardagna, M. Comuzzi, E. Mussi, B. Pernici, and P. Plebani, "PAWS: A Framework for Executing Adaptive Web-Service Processes," *IEEE Software*, vol. 24, no. 6, pp. 39–46, 2007.

[108] B. Pernici, D. Ardagna, and C. Cappiello, "Business Process Design: towards Service-based Green Information Systems," in *In SSME Cross Session in IFIP World Computer Congress, invited paper, Springer, Sept. 2008*, 2008.

[109] L. Baresi, D. Bianchini, V. D. Antonellis, M. G. Fugini, B. Pernici, and P.Plebani, "Context-Aware Composition of E-services," in *Technologies for E-Services, 4th International Workshop, TES 2003, Berlin, Germany, September 8*. Springer, 2003, pp. 28–41.

[110] P. K. McKinley, S. M. Sadjadi, E. P. Kasten, and B. H. C. Cheng, "A Taxonomy of Compositional Adaptation," Department of Computer Science and Engineering, Michigan State University, Tech. Rep., 2004.

[111] A. Ketfi, N. Belkhatir, and P.-Y. Cunin, "Dynamic Updating of Component-based Applications," in *SERP*, 2002.

[112] M. Aksit and Z. Choukair, "Dynamic, Adaptive and Reconfigurable Systems Overview and Prospective Vision," in *ICDCS Workshops*, 2003, pp. 84–.

[113] G. Kiczales, J. Lamping, A. Mendhekar, C. Maeda, C. V. Lopes, J.-M. Loingtier, and J. Irwin, "Aspect-oriented programming," in *ECOOP*, 1997, pp. 220–242.

[114] P. Clements and L. Northrop, *Software Product Lines: Practices and Patterns*. Addison-Wesley, Reading, Mass., 2001.

[115] G. B. Klaus Pohl and F. J. van der Linden, *Software Product Line Engineering: Foundations, Principles and Techniques*. Secaucus, NJ, USA: Springer-Verlag New York, Inc., 2005.

[116] F. van der Linden, "Software Product Families in Europe: The Esaps & Café Projects," *IEEE Software*, vol. 19, no. 4, pp. 41–49, 2002.

[117] J. Coplien, D. Hoffman, and D. Weiss, "Commonality and Variability in Software Engineering," *IEEE Software*, vol. 15, no. 6, pp. 37–45, 1998.

[118] H. Gomaa and M. Barber, *Designing Software Product Lines With UML: From Use Cases to Pattern-Based Software Architectures*. Addison-Wesley, Reading, Mass., 2004.

[119] K. Kang, S. Kim, J. Lee, and al., “FORM: A Feature-oriented Reuse Method with Domain-specific Reference Architectures,” *Annals of Software Engineering*, vol. 5, pp. 143–168, 1998.

[120] K. Czarnecki, S. Helsen, and U. Eisenecker, “Formalizing Cardinality-based Feature Models and Their Specialization,” *Software Process: Improvement and Practice*, vol. 10, no. 1, pp. 7–29, 2005.

[121] S. H. Chang and S. D. Kim, “A Variability Modeling Method for Adaptable Services in Service-Oriented Computing,” in *SPLC ’07: Proceedings of the 11th International Software Product Line Conference*. Washington, DC, USA: IEEE Computer Society, 2007, p. 261268.

[122] S. O. Hallsteinsen, E. Stav, A. Solberg, and J. Floch, “Using Product Line Techniques to Build Adaptive Systems,” in *SPLC*, 2006, pp. 141–150.

[123] R. Wolfinger, S. Reiter, D. Dhungana, P. Grunbacher, and H. Prahofer, “Supporting Runtime System Adaptation through Product Line Engineering and Plug-in Techniques,” in *ICCBSS ’08: Proceedings of the Seventh International Conference on Composition-Based Software Systems (ICCBSS 2008)*. Washington, DC, USA: IEEE Computer Society, 2008, p. 2130.

[124] *DOPLER: An Adaptable Tool Suite for Product Line Engineering*, vol. 2. IEEE Computer Society, 2007.

[125] R. Clotet, D. Dhungana, X. Franch, P. Grnbacher, L. Lpez, J. Marco, and N. Seyff, “Dealing with Changes in Service-Oriented Computing Through Integrated Goal and Variability Modelling,” in *Proceedings 2nd International Workshop on VaMoS, 2008*, 2008.

[126] C. Pérez, T. Priol, and A. Ribes, “A Parallel CORBA Component Model for Numerical Code Coupling,” *The International Journal of High Performance Computing Applications (IJHPCA)*, vol. 17, no. 4, pp. 417–429, 2003, special issue Best Applications Papers from the 3rd Intl. Workshop on Grid Computing. [Online]. Available: http://www.irisa.fr/paris/Biblio/Papers/Perez/PerPriRib03IJHPCA.pdf

[127] M. Aldinucci, M. Coppola, M. Danelutto, M. Vanneschi, and C. Zoccolo, “ASSIST as a Research Framework for High-performance Grid Programming Environments,” in *Grid Computing: Software environments and Tools*, J. C. Cunha and O. F. Rana, Eds. Springer, Jan. 2006.

[128] J. Buisson, F. André, and J.-L. Pazat, “Dynamic adaptation for Grid computing,” in *Advances in Grid Computing - EGC 2005 (European Grid Conference, Amsterdam, The Netherlands, February 14-16, 2005, Revised Selected Papers)*, ser. LNCS, P. M. A. Sloot, A. G. Hoekstra, T. Priol, A. Reinefeld, and M. Bubak, Eds., vol. 3470. Amsterdam: Springer-Verlag, June 2005, pp. 538–547. [Online]. Available: http://www.irisa.fr/paris/Biblio/Papers/Buisson/BuiAndPaz05EGC.pdf

Glossary

- **Adaptation** is a process of modifying Service-Based Application in order to satisfy new requirements and to fit new situations dictated by the environment on the basis of Adaptation Strategies designed by the system integrator.
- **Adaptation Requirements and Objectives** are the requirements and needs that the Service-Based Application should achieve in reaction of critical changes and events.
- **Adaptation Strategies** define the possible ways to achieve Adaptation Requirements and Objectives given the available Adaptation Mechanisms.
- **Adaptation Mechanisms** are the tools and mechanisms provided by the underlying platform in different Functional Layers of Service-Based Application that allow for implementation of various Adaptation Strategies.
- **Adapter** is a hardware device or software component, that converts transmitted data from one presentation form to another.
- **Aspect-Oriented Programming (AOP)** is a programming paradigm that attempts to aid programmers in the separation of concerns, specifically cross-cutting concerns, as an advance in modularization.
- **Business Activity** is apart of a business process consisting of a series of activities implemented across workflow systems, ERP systems and legacy applications, possibly across organizational boundaries.
- **Business Activity Monitoring** provides near real-time monitoring of Business Activities, measurement of Key Performance Indicators, their presentation in dashboards, and automatic and proactive notification in case of deviations.
- **Business Intelligence (BI)** refers to technologies, applications and practices for the collection, integration, analysis, and presentation of business information and also sometimes to the information itself.
- **BPEL (Web Services Business Process Execution Language)** is a de-facto standard language for specifying business process behavior based on Web Services.
- **Component-based Software Engineering (CBSE)** is a branch of the software engineering discipline, with emphasis on decomposition of the engineered systems into functional or logical components with well-defined interfaces used for communication across the components.
- **Computational reflection** refers to the ability of a program to reason about, and possibly alter, its own behavior.
- **Context** refers to the physical and social situation in which a Service-Based Application or a service is embedded. It is defined by any information that can be used to characterize the situation of an entity - be it a person, a place or a physical or computational object.

- **Diagnosis** is the process of finding the cause of a problem starting from an observable manifestation (symptom) of the problem itself.
- **Evolution** of Service-Based Application is a long-term history of continuous modification of SBA after its deployment in order to correct faults, to improve performance or other attributes, or to adapt the product to a modified environment.
- **Grid Computing** is on-line computation or storage offered as a service supported by a pool of distributed computing resources, also known as utility computing, on-demand computing, or cloud computing. Data grids provide controlled sharing and management of large amounts of distributed data, often used in combination with computational grids.
- **Key Performance Indicator (KPI)** are financial and non-financial metrics used to help an organization define and measure progress toward organizational goals. KPIs are used in Business Intelligence to assess the present state of the business and to prescribe a course of action.
- **Mediation** refers to an activity in which a neutral third party, the mediator, assists two or more parties in order to help them achieve an agreement on a matter of common interest.
- **Middleware** is layer of services separating applications from operating systems and network protocols.
- **Monitor** is a program or a set of programs that observes the execution of SBA.
- **Monitoring** is a process of collecting and reporting relevant information about the execution and evolution of the Service-Based Application.
- **Monitoring Events** are the events that deliver the relevant information about the application evolution and changes in the environment.
- **Monitoring Mechanisms** are the tools and facilities for continuous observing and detecting relevant Monitoring Events.
- **Optimization** of Service-Based Application is the process of modifying an application to make some aspect of it work more efficiently or use fewer resources.
- **Parallel programs** refer to a set of software technologies to express parallel algorithms and match applications with the underlying parallel systems.
- **Process Mining** is a set of techniques and methods that allow for the analysis of business processes based on event logs. They are often used when no formal description of the process can be obtained by other means, or when the quality of an existing documentation is questionable
- **Recovery (Repair)** is a process of restoring the application after failing to perform one or more of its functions to fully satisfactory execution by any means other than replacement of the entire application.
- **Service-Based Applications (SBA)** is an application composed by a number of possibly independent services, available in a network, which perform the desired functionalities of the architecture. Such services could be provided by third parties, not necessarily by the owner of the service-based application.
- **Service-Level Agreement (SLA)** is that part of a service contract where the level of service is formally defined. In practice, the term SLA is sometimes used to refer to the contracted delivery time (of the service) or performance.
- **Software Product Line Engineering (SPLE)** refers to engineering methods, tools and techniques for creating a collection of similar software systems from a shared set of software assets using a common means of production.