Grant Agreement N° 215483

Title:	*Baseline of Adaptation and Monitoring Principles, Techniques, and Methodologies across Functional SBA Layers*
Authors:	*CITY, FBK, INRIA, Polimi, SZTAKI, USTUTT*
Editors:	*Raman Kazhamiakin (FBK)*
Reviewers:	*Andreas Metzger (UniDue)*
	Qing Gu, Patricia Lago (VUA)
Identifier:	*Deliverable # PO-JRA-1.2.3*
Type:	*Deliverable*
Version:	*1.0*
Date:	*15 June 2009*
Status:	*Final*
Class:	*External*

Management Summary

The goal of this document is twofold. First, the document aims to define the baseline for the adaptation and monitoring principles, techniques, and methodologies across functional SBA layers. For this purpose, the document describes the functional structure of SBA and shows the existing and potential adaptation and monitoring approaches and mechanisms within those layers. These mechanisms will form the basic building blocks on top of which the future cross-layer adaptation and monitoring frameworks and approaches will be founded. Using a set of scenarios, the document shows and classifies a set of problems and requirements that the cross-layer adaptation and monitoring framework should address. It is then shown how these requirements are mapped into the required principles and mechanisms and how the initial research results of the partners contribute to the framework.
Second, the document provides a detailed survey and a classification of the monitoring approaches that focus on context and HCI aspects. In this way, the survey provides a starting point for defining and elaborating novel integrated monitoring approaches that consider context, user and user interaction aspects, which are subject of research in the work-package task T-JRA-1.2.3.

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

Members of the S-Cube consortium:

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

Published S-Cube documents

All public S-Cube deliverables are available from the S-Cube Web Portal at the following URL:

http://www.s-cube-network.eu/results/deliverables/

The S-Cube Deliverable Series

Vision and Objectives of S-Cube

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

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

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

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

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

Foreword

This deliverable, PO-JRA-1.2.3 "Baseline of Adaptation and Monitoring Principles, Techniques, and Methodologies across Functional SBA Layers" aims to address the following goals:

- to define a baseline for the integrated cross-layer PTMs for SBA adaptation and monitoring in order to deal with the problem of fragmentation and isolation of the approaches existing in isolated functional SBA layers. For this purpose, the deliverable defines the functional structure of SBA and shows the existing and novel adaptation and monitoring approaches and mechanisms within those layers. On top of these mechanisms future cross-layer adaptation and monitoring frameworks and approaches will be founded. The document shows and classifies a set of problems and requirements that the cross-layer adaptation and monitoring framework should address with the help of illustrative scenarios. Furthermore, the document shows how these requirements are mapped into the required principles and mechanisms and how the initial research results of the partners contribute to the framework. This document provides the basis for the future research directions of the WP participants to deal with one of the challenges of the WP, namely provisioning of an integrated and comprehensive monitoring and adaptation framework.
- to provide a detailed survey and a classification of the monitoring approaches that focus on context and HCI aspects. This activity is a necessary prerequisite to address another key challenge of the WP, namely to enable context- and HCI-aware SBA adaptation and monitoring, where the wide range of the novel application areas (e.g., telecommunication domain and mobile applications) are dealt with. On top of this survey novel integrated monitoring approaches that consider context, user and user interaction aspects will be defined and elaborated in the following phases of the S-Cube project.

Acknowledgements: the editor would like to thank all the S-Cube partners who have contributed to the presented document and the reviewers of this deliverable, Andreas Metzger, Patricia Lago, and Qing Gu for their valuable and very detailed comments and suggestions regarding initial versions of this deliverable.

Contents

Chapter 1

Introduction

1.1 Motivation

Service-Based Applications (SBA) run in dynamic business environments and address constantly evolving requirements. These applications should hence become drastically more flexible; they should be able to adequately identify and react to various changes in the business requirements and application context. These challenges make monitoring and adaptation the key elements of modern SBA functionality.

A fundamental problem with the state-of-the-art results on monitoring and adaptation is their fragmentation and isolation. Indeed, literally all the existing solutions and frameworks address only a particular aspect of the SBA functionality, without considering the problem of designing, providing, and executing the application as a whole. Usually, the existing adaptation and monitoring solutions target a particular functional layer, i.e., business process layer, composition, or infrastructure layer. While these solutions are quite effective when considered in isolation, they may be incompatible or even harmful when the whole SBA is considered. In these settings it is crucial to provide novel solutions that should be able to consider the problem of SBA adaptation and monitoring as a whole, across different elements of the architecture. In this way the solutions will not only be able to find and understand the implications among changes in different layers and components, but also to provide new means to identify, integrate, and coordinate activities at different layers in a consistent and correct way.

Another important problem that a wide range of the SBAs should face is the need to consider and exploit the knowledge about the application context in the process of monitoring and adaptation. The context, e.g., stakeholders, other IT systems, rules and regulations as well as business objects, end-user settings and even physical environment, plays an important role for developing and maintaining software systems. Changes in the context – which may for instance be due to changing stakeholders and their requirements – must be reflected in the software system and managed in appropriate ways, otherwise the system falls out of use. In particular, service based applications should be equipped with the required mechanisms to adapt quickly to changes in the system's context, particularly at run-time.

This document, deliverable PO-JRA-1.2.3 consists of two parts. In the first part we aim to define a baseline for the SBA monitoring and adaptation principles, techniques, and mechanisms across functional SBA layers. This task is achieved in the presented document (chapter 2) as follows.

- We describe the functional structure of SBA as it is defined and described in the work-packages of the joint research activity JRA-2 and show the existing and potential adaptation and monitoring approaches and mechanisms within those layers. This mechanisms will form the basic building blocks on top of which the future cross-layer adaptation and monitoring frameworks and approaches will be founded.

- We identify the set of problems that the SBA monitoring and adaptation may face when the existing solutions are applied independently and in isolation from each other. This analysis is based on a set of scenarios from the automotive case study presented in Deliverable CD-IA-2.2.2 [37]. With

the help of scenarios we illustrate the potential problems and identify the required functionalities and features that the cross-layer approaches and methodologies should provide.

- Using the conceptual adaptation and monitoring framework presented in Deliverable CD-JRA-1.2.2 [26] we map the identified set of requirements onto the conceptual monitoring and adaptation architecture, showing the key mechanisms and principles to be provided in the scope of the cross-layer monitoring and adaptation.

- Finally, we show the initial proposals and ideas of the work-package members to instantiate and realize those mechanisms and principles.

We remark that the results presented in this chapter are based on the materials presented in a set of papers. In particular, the analysis of the requirements for the cross-layer adaptation and monitoring is described in [41], while the approaches and solutions for are presented in [35, 7, 22].

The second part of the deliverable (chapter 3) presents a survey on the monitoring approaches that particularly focus on the context and human-computer interaction (HCI) aspects. This survey first makes a review of the existing works in those areas and then provides a classification of key concepts in these domains. This classification refines and focuses the general taxonomy of principles, techniques and mechanisms presented in the previous deliverable of this WP [26]. This survey provides a starting point for defining and elaborating novel integrated monitoring approaches that consider context, user and user interaction aspects, which are subject of research in the work-package task T-JRA-1.2.3.

1.2 Relations to the WP Research Objectives

Below we recap the key challenges of the WP, for which this deliverable is relevant. We also show how the presented documents contributes to those challenges. The challenges are described in [26]. We remark that, as the WP-JRA-1.2 addresses the problem of SBA monitoring and adaptation that is cross-cutting to the technology layers of the SBA, the presented challenges are also cross-cutting with respect to the research areas addressed by the S-Cube consortium.

- **Comprehensive adaptation and monitoring framework**. The work-package will concentrate on providing a holistic framework for adaptation and monitoring principles, techniques, and methods, which will enable the integration of different, isolated, and fragmented solutions. In particular, this concerns the *need to integrate the existing approaches for SBA monitoring and adaptation across functional SBA layers*: cross-layer monitoring provides a way to properly locate and evaluate the source of the problem and its impact, while cross-layer adaptation will allow us to properly identify and propagate the necessary adaptation activities in different elements of the SBA architecture. As well as in case of monitoring, new solutions will integrate isolated adaptation mechanisms available at different functional layers (and investigated in the corresponding workpackages of JRA-2) into the holistic cross layer approaches.

- **Exploiting contextual information and user aspects for SBA monitoring and adaptation**. The information about different types of the SBA context, as well as about the user and its settings is crucial for the application logic. Novel approaches are necessary for being able to specify and observe this information and for driving the selection, realization, and enactment of the corresponding adaptation actions. The research here should answer the following questions:

  - What are the relevant elements in the context (context factors), which are crucial for the application functionality?
  - Which techniques can be used to recognise changes of these context factors (generation of monitoring events)?

– How could these monitoring events be translated into an adaptation strategy?

With respect to these challenges, the role of the presented deliverable is as follows. The first part of the deliverable aims to define the baseline for cross-layer SBA adaptation and monitoring, and, therefore, to provide a unified vision on the cross-layer integration of the principles and mechanisms. Having provided the set of requirements, set of necessary mechanisms and principles, the deliverable defines a roadmap for the future joint activities of the WP members that should achieve a first round of such integration (to be presented in Deliverable CD-JRA-1.2.4).

The second part of the document aims to achieve a shared view and a classification of the monitoring principles and mechanisms that focus on the SBA context and user-related information. Building on top of these results, another round of integration of approaches will focus on considering contextual and HCI aspects in SBA monitoring (to be presented in Deliverable CD-JRA-1.2.5).

Chapter 2

Baseline for Cross-layer Adaptation and Monitoring

The problem of monitoring and adaptation of various types of software systems has gained a lot of interest both in the research community and in industry. In the recent years, these aspects have attracted more and more interest in the area of SBA and in Service-Oriented Computing (SOC). However, the results and directions are still insufficient w.r.t. the challenges. One of the key issues here is that the proposed approaches are very fragmented: they address only specific problems peculiar to a particular aspect of the SBA functioning and a particular functional SBA layer, i.e., business process management layer, service composition and coordination layer, and service infrastructure layer. In complex real-scale applications, however, the realization of different SBA layers may be highly interleaved: different artifacts at one layer may refer to the same artifacts in another layer, while such relations are ignored by the isolated monitoring and adaptation solutions. As a consequence, wrong problems are detected, incorrect decisions are made, and the modifications at one level may damage the functionality of another layer.

In this section we present the baseline for the problem of adaptation and monitoring principles, techniques, and mechanisms across functional SBA layers. First, we provide a classification of the monitoring and adaptation approaches according to the functional SBA layers. We then define the requirements for the cross-layer monitoring and adaptation, that should be satisfied by the novel integrated approaches providing coherent and holistic solutions for monitoring and adapting the whole application. We illustrate the problem using a series of case studies and provide a set of requirements that the novel cross-layer approaches should address. Based on the taxonomy of those requirements, we also define the necessary mechanisms and principles that are necessary for addressing the requirements and that constitute an integrated cross-layer framework. Finally, we describe initial approaches, principles and methods aiming to enable cross-layer monitoring and adaptation.

The presented results are based on the works of the WP members. In particular, the analysis of the requirements for cross-layer adaptation and monitoring summarizes the materials presented in [41]. The initial approaches towards cross-layer adaptation and monitoring principles techniques and mechanisms are based on the works described in [35, 7, 22].

2.1 Adaptable and Monitorable Service-Based Applications

An SBA can be represented by its three functional layers, namely business process management (BPM), service composition and coordination (SCC) and service infrastructure (SI). BPM is the highest level functional layer where the application activities, constraints and requirements are described without going into design details. Here we consider the entire business process as a workflow and the business activities as the constituents of this workflow. SC is the layer between BPM and SI layers where the

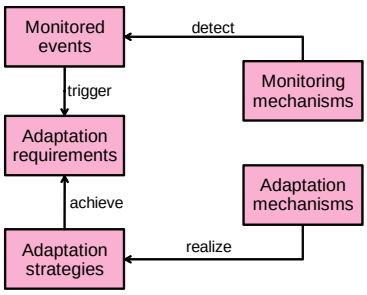

Figure 2.1: Conceptual A&M framework

basic workflow constructed at BPM is refined by the composition of suitable services that are capable of realizing the corresponding business activities and altogether accomplishing the construction of the whole SBA. SC is the responsible layer to organize and manage the control and data flows among services in the composition. Finally, at the lowest level we have SI layer which provides the underlying run-time environment for the composed SBA. As well as the service realization, the collection of all available services is kept at this layer through a service registry. Consequently, the identified services for the composition are discovered and realized at SI level [36].

2.2 Monitoring and Adaptation in Different SBA Layers

A general framework for SBA monitoring and adaptation defined in [26] is represented in Figure 2.1. A wide range of the adaptation and monitoring problems as well as the existing approaches identified and realized at different functional SBA layers can be mapped onto this framework. Such adaptation and monitoring mechanisms should then used by the holistic integrated cross-layer adaptation and monitoring framework as a building blocks, from which the cross-layer mechanisms will be built. Note, however, that currently not for all of the presented elements appropriate mechanisms and techniques has already been defined. New approaches still have to be provided by the research in order to enable cross-layer monitoring and adaptation in a holistic, integrated manner.

We will describe monitoring and adaptation approaches at different layers according to the key elements of the taxonomy presented in [26]: i.e., subject of monitoring, monitored events, and monitoring mechanisms, adaptation requirements, adaptation strategies/actions and adaptation mechanisms. It is based on the survey on adaptation and monitoring approaches presented in [8] and the taxonomy of adaptation and monitoring principles, techniques and mechanisms presented in [26]. The existing approaches that address the specific problems of adaptation and monitoring of different elements of the SBA has been related to the corresponding elements of the taxonomy. Tables 2.1 and 2.2 represent this classification; more details and references to the corresponding approaches may be found in [41].

We note that although the state-of-the-art approaches may cover a wide spectrum of the problem and application domains and may address a significant adaptation need for an SBA, none of them are complete. They usually considered in isolation from each other; exploit different information models of monitored information and events and different platforms; focus on a local solution for a specific adaptation requirement without taking into account its dependencies or effects on different SBA layers. As we will illustrate in the next section, such ad-hoc approaches are not promising in terms of addressing a proper solution to construct a monitorable and adaptable SBA in a real environment.

Table 2.1: Monitoring at different functional SBA layers

Layer		Subject	Events	Mechanisms
BPM	Business Process monitoring	business process model; transaction protocol; data and control flow	violation of correctness properties of instance executions; correspondence to the process model; violation of transactional properties	BAM/BI; process log analysis, process mining
	KPI monitoring	KPI	KPI violations	BAM/BI, specific monitoring approaches
SCC	Monitoring functional properties of SC	service composition; data and control flow	violation of functional properties of a composition; violation of functional properties of a constituent services	special purpose monitoring engines based on temporal logics; rule languages; calculi
	Monitoring non-functional properties of SC	composition PPMs, utility functions, QoS properties of constituent services	violations of expected values/thresholds, SLA violations	special purpose monitoring engines, service monitors
SI	Grid monitoring	grid infrastructure (site, virtual organization, whole grid); grid applications (application state, application progress)	wide range of infrastructural and application events	grid monitoring platforms and architectures
	Monitoring of component-based systems	components (state, bindings, messages, internal data), component platform (performance, dependability, state/use of resources)	component- and middleware-related events	middleware monitoring mechanisms, internal component monitoring mechanisms

Table 2.2: Adaptation at different functional SBA layers

Layer		Requirements	Strategies/actions	Mechanisms
BPM	Business process adaptation	optimize process (to meet violated KPI); recover from unforeseen execution; process customization	modify business process control flow (add, delete, replace process tasks and process fragments), modify business process data flow (change data dependencies, values); process model re-design	ad-hoc modifications (performed by business analysts); evolution
	KPI adaptation	adjust to changing business goals, business context, ASN elements	add or remove KPI, change KPI values	ad-hoc modification; negotiation; evolution
	ASN adaptation	optimize costs; transactionality; accommodate to ASN changes	change transaction protocol; change service; re-negotiate for an offering	ad-hoc modifications, negotiation mechanisms, evolution
SCC	Composition-related adaptation	adjust to changed process model or KPI, process optimization, recovery	re-composition; control/data flow changes; PPM changes	automated composition; model-driven transformations; fragmentation
	Service-related adaptation	changes or failures of constituent services; optimization; SLA violation	replace a service; re-execute a service; re-negotiate QoS	dynamic discovery and binding; negotiation
SI	Service discovery-related adaptation	optimization, business requirements	change registry; update registry (new services, new descriptions); change discovery mechanism; change selection mechanism	platform-specific; reputation management
	Service realization-related adaptation	optimization, adjust to infrastructural failures	modify/re-configure service platform (software updates, OS changes, virtual machines, physical platforms); modify/re-configure service resources (allocation/release of resources, load balancing); adapt resource management (change resource brokering mechanisms, re-configure grid application, re-execute application)	ad-hoc changes; self-* techniques

2.3 Requirements for Cross-layer SBA Adaptation and Monitoring

Lack of cross-layer integrated monitoring and adaptation principles and mechanisms may lead to a variety of problems and hazards in SBAs. We distinguish the following classes of problems: *lack of alignment of monitored events*, *lack of adaptation effectiveness*, *lack of compatibility*, and *lack of integrity*. In

[41] these problems are illustrated and explained with the help of the automotive case study [37]. Here we just recap those classes of problems.

2.3.1 Lack of Alignment of Monitored Events

In many cases, if the monitoring is performed by specific mechanisms provided at different layer in isolation from each other, then the corresponding events are not aligned and the critical information is not adequately propagated across layers. This may lead to the situation where the source of the monitored problem is identified incorrectly; to the situation where the same problem results in separate events at different layers and then triggers different (possibly contradictory) adaptations, etc.

Consider an example, where due to a bottleneck in the service middleware the performance of the involved services goes down. This may be detected independently by the infrastructure-level monitor and by the composition monitor that observes the performance of the whole composition. As a result, different adaptation actions may be triggered at different layers (e.g., composition fragmentation and parallelizing of tasks at the SCC layer and resource re-allocation at the SI layer). Furthermore, the actions may be contradictory (e.g., the new resource allocation discards the expected effects of the parallelization).

The requirements that the cross-layer monitoring framework should satisfy in this case are the following:

- to provide necessary monitoring mechanisms at all the layers, where the problem of interest may be observed;
- to provide means to propagate the monitored information across the layers in order to properly diagnose the actual source of the problem;
- to align and correlate the monitored events across functional layer in order to avoid spanteneous and uncoordinated adaptation activities at different layers.

2.3.2 Lack of Adaptation Effectiveness

Another possible problem of monitoring and adaptation performed at different layers in isolation is that the adaptation activities may fail to achieve the expected effect, since they do not take into account the properties of other layers. For example, at the SCC layer in order to reduce execution the adaptation aims to execute independent tasks in parallel by delegating them to different services. In order to achieve this effect it is necessary that those services are independent also at the SI layer (which may not be the case in Grid).

In order to overcome the problem of this sort it is crucial

- to take into account features and properties of the whole SBA stack when the adaptation requirements are defined and the adaptation strategy is identified.

2.3.3 Lack of Compatibility

The problem of cross-layer compatibility of adaptation activities refers to the situation, where the adaptation performed at one functional SBA layer is not compatible with the requirements and constraints posed by the application design at other layers.

In order to take into account the issue of the cross-layer adaptation compatibility, it is necessary to consider how the identified adaptation activities influence the SBA as a whole. In particular, it is crucial to be able to

- define certain “boundaries” for each of the layers, and

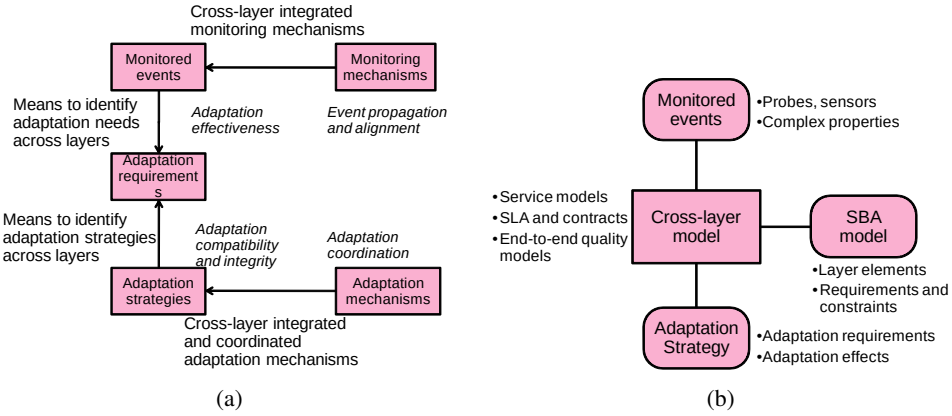

Figure 2.2: Cross-layer adaptation and monitoring framework: (a) problem types and required mechanisms within the conceptual A&M framework, and (b) conceptual cross-layer model relating the elements of the SBA layers with the adaptation and monitoring concepts.

- to check that the adaptation activities of other layers do not cross those boundaries. Indeed, this implies that both these elements are appropriately modelled and specified, and the corresponding analysis is done before the adaptation is executed.

2.3.4 Lack of Integrity

A problem of cross-layer adaptation integrity deals with a situation, where it is not enough to perform the adaptation at a particular layer only, but several actions at different layer should be performed. For instance, when the adaptation is performed at the BPM layer (e.g., a business process is changed), it is necessary to propagate the changes also to the other layers (e.g., change the compositions that manage corresponding process instances and/or perform some compensation actions; negotiate, bind, and adjust to the new formats and protocols at the SI layer, etc.).

To address this problem, novel mechanisms are necessary

- to identify, aggregate, and enact wide range of adaptation actions available at different functional layers. Indeed, these actions should be performed in a coordinated manner; some centralized mechanism should be able to control and manage isolated layer-specific tools.

2.4 Prospective Cross-layer Adaptation and Monitoring Framework

Figure 2.2(a) represents the placement of the requirements for the cross-layer adaptation and monitoring in the general adaptation and monitoring framework. It also shows the mechanisms and approaches necessary to achieve those requirements.

2.4.1 Required cross-layer adaptation and monitoring mechanisms

- *Cross-layer integrated monitoring mechanisms*. Cross-layer monitoring mechanisms built on top of the existing and new monitoring capabilities should provide a holistic, integrated infrastructure for the SBA monitoring. In particular, such infrastructure should allow for (i) expressing layer-specific properties and capabilities in an integrated and uniform manner, and (ii) relate events and mechanisms of different layers to each other to enable their correlation, aggregation and alignment. In this way, the problem of alignment of monitored events will be addressed.

- *Cross-layer integrated and coordinated adaptation mechanisms*. Cross-layer adaptation mechanisms should guarantee that the adaptation activities being triggered at different layers or across those layers are properly managed, i.e., control, validate, and coordinate them (thus supporting adaptation integrity).

- *Means to identify adaptation needs across layers*. Cross-layer mechanisms should be capable of properly identifying the necessary SBA adaptation requirements, when the changes concern not a single layer, but the whole application. That is, these mechanisms should provide ways to (i) properly identify the source of the problem and the corresponding requirements; (ii) map those requirements onto the relevant functional layers; (iii) identify the adaptation strategies at different layers that together achieve the expected goal. Such mechanisms would allow addressing the problem of adaptation effectiveness.

- *Means to identify adaptation strategies across layers*. To address the problems of the adaptation compatibility and integrity, the mechanisms for the identification and selection of the adaptation strategies should be able to (i) validate the adaptation strategies against the whole model of the application; (ii) foresee whether the adaptation strategies are sufficient to achieve the corresponding requirements; (iii) to identify search appropriate adaptation strategies when the previously selected strategies are insufficient or may in turn trigger some other adaptations.

2.4.2 Required principles for cross-layer adaptation and monitoring

To provide a basis for the cross-layer SBA adaptation and monitoring and to enable various mechanisms described above, it is necessary to provide ways to explicitly relate different conceptual elements to each other and across different layers. This vision is reflected by the conceptual model represented in Figure 2.2(b). Without loss of generality, the model shows that various specific elements of the application, of the adaptation mechanisms and capabilities, and of the monitoring mechanisms and capabilities should be related to a centralized, high-level model which defines some cross-cutting aspects of the SBA. Through this centralized model different layers, different adaptation strategies and monitoring mechanisms are related, thus providing a way to propagate and align events, to reason about cross-layer strategies, impacts, and dependencies, to control and coordinate adaptation mechanisms. This should include, in particular,

- *cross-layer representation of monitored events*. Cross-cutting aspects of the SBA functioning should be used to capture and represent relevant monitored events, monitoring mechanisms and information sources provided by the monitoring approach at a particular layer and to abstract from the low-level realization and specification details;

- *cross-layer representation of adaptation strategies*. To enable the analysis of adaptation integrity, effectiveness, and compatibility, the adaptation strategies and mechanisms of different layers should also be characterized in terms of the relevant cross-layer models;

- *cross-layer representation of the SBA model*. In order to be able to understand the relation and the impact of the adaptation activities on the elements of the application architecture, the latter should also be characterized in terms of some cross-cutting models.

2.5 Initial Approaches and Mechanisms for Cross-layer SBA Adaptation and Monitoring

The required mechanisms and principles identified before may be realized in a variety of ways, depending on a particular domain, problem, or goals of the application. In this section, we summarize several possible solutions identified and elaborated by the partners of the WP. Further details on the realization

of those solutions may be found in the respective research works (the corresponding papers of the WP members are attached to this deliverable). For each of the solutions we describe how they contribute to the prospective cross-layer adaptation and monitoring framework and to which required mechanisms they correspond.

2.5.1 Unified Model for Monitoring Design Evolution and Run-time Events

The increasing need for continuously available software systems has raised two key-issues: self-adaptation and design evolution of Service Based System. The former one requires service based systems to monitor their execution platform and automatically adapt their configuration and/or architecture to adjust their quality of service (optimization, fault-handling). The later one requires new design decisions to be reflected on the fly on the running system to ensure the needed high availability (new requirements, corrective and preventive maintenance). However, design evolution and self-adaptation are not independent and reflecting a design evolution on a running self-adaptive system is not always safe.

In many cases, if the monitoring is performed by specific mechanisms provided at different layer in isolation from each other, then the corresponding events are not aligned and the critical information is not adequately propagated across layers. This may lead to the situation where the source of the monitored problem is identified incorrectly; to the situation where the same problem results in separate events at different layers and then triggers different (possibly contradictory) adaptations, etc.

We propose to unify run-time adaptation and run-time evolution by monitoring both the run-time service-based-system and the design models (Orchestration, etc...). Thus, it becomes possible to correlate those heterogeneous events and to use Complex Event Processing engine to elaborate a pertinent decision for run-time adaptation. This work proposes a cross-layer monitoring mechanism that is able to relate models at BPM (design specifications of, e.g., business processes) to the other run-time events. The detailed description of the approach may be found at [35].

Contribution to Cross-Layer Adaptation and Monitoring

The presented approach provides a mechanism for correlating and aggregating monitored events across functional SBA layers (*integrated cross-layer monitoring mechanisms*). The main purpose of this mechanism is to unify the model evolution with the run-time adaptations by integrating the design decisions and modifications regarding the evolution of the higher level model of the SBA (e.g., the BPM model) with the lower level events of the SBA instances (which may be, e.g., infrastructure events). Furthermore, the ability to detect and differentiate evolutionary changes and the run-time events allows for better diagnosis of the potential problems and, therefore, for better capturing the requirements for adaptation. As so, the approach supports also the *means for identifying adaptation needs across layers*.

2.5.2 Integrated Run-time Monitoring Approach

In order to approach the problem of alligning and correlating monitored events across functional layers (i.e., to enable cross-layer SBA monitoring) there is a need to provide mechanisms to aggregate, evaluate, and propagate various layer-specific events in a uniform manner. In [6, 7] the authors present a monitoring framework that allows for integrating wide range of events into more complex properties, measure them, and even further exploit the result of such measurements as the basic events of some other monitored properties.

To achieve these capabilities, the framework integrates two monitoring approaches, namely Dynamo [5] and Astro [4]. The capabilities of the former are exploited to provide a way to define basic monitored properties and events, measured data, utility functions, placement of measurements, etc. In this way it is possible to define a wide range of probes and events that are specific to a particular functional layer (e.g., infrastructural measurements, events occuring in service compositions, etc.). The capabilities of the latter are exploited for the definition of more complex events and properties, e.g., temporal properties,

the events and properties characterizing not only a single instance of an application or a single execution of the component, but a series or classes of executions. In this way, one can express the advanced properties of the service composition or of the corresponding business processes. Furthermore, the extended recursive model of monitored properties as well as the capability to correlate and aggregate variety of events from independent sources enables the key required mechanisms for cross-layer SBA monitoring.

More details on the underlying model of monitoring properties and on the realization of the framework may be found in [7, 6]. While the works focuses on the events and properties pertaining to the service compositions, the approach may be further extended to cover other layers of SBA stack. Indeed, the formalism and notation used to define complex properties and events is agnostic to the specific types and semantics of the basic events underlying those properties. It is, therefore, possible to define various events and metrics specific to particular functional layer. These events may include, for example activities and events within the BPEL code representing the service compositon, business events corresponding to the initiation/termination of an instance of the business properties, contextual properties (accessible through external services), service characteristics (e.g., QoS properties), infrastructure events. The monitoring framework allows in this way for correlating events across different layers and even specifying properties that span different layers (e.g., properties related to the classes of process instances and expressed in terms of events at the service level).

Contribution to Cross-Layer Adaptation and Monitoring

This approach instantiates the *integrated cross-layer monitoring mechanisms*. The modular architecture underlying the approach enables definition of a variety of the event types corresponding to different functional SBA layers. The formalism underlying the specification of the complex properties allows for integration, unification, and correlation of those events, providing a uniform monitoring infrastructure.

2.5.3 Monitoring and Analysis of Influential Factors of KPIs

As discussed in Section 2.1, in the SBA Layering we distinguish between different types of metrics in the non-functional view: KPIs on BPM layer, PPMs on service composition layer, and QoS on service infrastructure layer. When business people define KPIs they are not only interested in monitoring of their values in a timely fashion, but also analyzing the causes (a.k.a. influential factors) why KPIs violate their target values. Thereby, influential factors can be situated on all three SBA layers, and are measured by KPIs, PPMs, and QoS metrics. For example, the typical KPI customer satisfaction on BPM layer can be influenced by process performance metrics such as order delivery time, but also technical QoS metrics such as the availability of the Web service used by the customer for placing of orders.

In [49] we describe an approach to monitoring and analyzing the influential factors of KPIs in a cross-layer setting based on machine learning. The approach supports both monitoring of KPIs, PPMs, and QoS metrics across layers and analysis of metrics dependencies using decision tree algorithms. The user defines the KPIs which are to be measured and a set of potential influential factors as PPMs and QoS metrics. PPMs are specified based on process events, e.g. events published by a BPEL engine, whereas QoS metrics are measuring technical characteristics of the underlying service infrastructure such as availability by dedicated QoS monitors. At process runtime, all those metrics are monitored and correlated in near-real time and shown to the user in a dashboard. If the user wants to analyze the influential factors of a KPI, a decision tree is created which shows which PPM and QoS metrics mostly lead to the KPI satisfying or violating its target values. We call this tree a depedency tree as it shows the main dependencies of the KPI on lower lever metrics.

The future work will consist of extending this approach to supporting cross-layer adaptation based on dependency trees. Thereby, the influential factors identified by the dependency tree can be used as a target for adaptation strategies.

Contribution to Cross-Layer Adaptation and Monitoring

The main contribution of this approach with respect to the cross-layer adaptation and monitoring is the ability to identify the factors influencing violation of the SBA requirements. As the factors may correspond to the metrics defined at different SBA layers, such a method provides a way for diagnosis the source of the problem in the SBA stack, and, therefore, provides a basis for the *means to identify adaptation needs across layers*.

2.5.4 SLA Contract for Cross-layer Adaptation and Monitoring

This part of the report aims at proposing a framework for contract-based monitoring and adaptation with respect to customer's goals seen as an important part of the contract (see [22] for further details). An SLA contract is a mutual agreement between service providers and users. We discuss about SLA as a possible candidate for cross-layer model to be applied for cross-layer monitoring and adaptation of service based applications. The need to consider customer parameters when evaluating business services has become increasingly noticeable. We believe that the value of a service is highly influenced by the business of the organization, by the customers who are going to use the services, and by the IT infrastructure. Business service parameters are not always appropriate to express the users' satisfaction. Therefore, we emphasized on the importance of the customers perspective and parameters as a complimentary measurement for parameters defined from business service perspective. For that reason, we introduce Key Goal Indicators (KGI) which are parameters that state how well services or processes achieve the customers' goals. we distinguish between the formulation and evaluation of the business service performance and the formulation and evaluation of the customers' goals.

From the business level, the output of service provider is evaluated by Key Performance Indicators (KPIs) that show the degree of performance of business services. On the other hand, from the users' perspective, the evaluation is done through Key Goal Indicators (KGIs) which show how well the services are successful in the achievement of the customers' goals. In this sense, the Output of the service provider is differentiated from the Outcome obtained by the service users. Typical examples of service and business indicators are response time, process duration, process cost and service availability while some parameters related to KGIs could be financial return, satisfaction, reputation and trust. There are parameters in common between service, business and user such as time and cost; however, they are observed from different perspectives. Besides, there are also domain-dependent business process parameters and users parameters.

Service parameters alone are not enough to express the user satisfaction, which is typically related to KGI parameters, and they do not consider the perspective of customers. It is worth to state that IT infrastructure factors has to be considered in the contract since they have properties that influence the parameters of users and services. Therefore, in order to have a comprehensive SLA contract, it is increasingly important to consider the three factors together, namely KPI, KGI and IT infrastructure, as long as they have a close inter-relation. As a result, parameters of an SLA contract should be a mix of KPI, KGI and IT infrastructure type. The SLA contract is a fundamental part for monitoring and adaptation of service based applications. In fact this contract will be checked in the monitoring phase to see if there is any deviation or violation from the predefined contract occurs at run time. Besides, SLA contract is continuously checked for the purpose of optimization. Such a violation could be due to the defiance of IT infrastructure, business service and user parameters.

Taking advantage of the comprehensive SLA contract, we propose a contract based framework for monitoring and adaptation of service based applications. Our approach consists of five major phases: (1) Identifying KPI, KGI and IT infrastructure parameters (2) Mapping of parameters into a Contract (creation of an SLA contract and contract set up through negotiation); (3) Evaluation and monitoring (4) Adaptation, (5) Updating of the contract. In the first phase, the corresponding parameters related to KPI, KGI and IT infrastructure are specified. This should be done through a requirement engineering phase with an early participation of users in order to understand their actual needs. The second phase

is the SLA contract creation which includes the aggregation of parameters defined in the first phase. The parameters in the contract are specified by parameters from service provider with respect to the customer's parameters and the limitations that IT infrastructure parameters cause. Therefore, the contract has parameters resulting from a mapping process combining business services, IT infrastructure, and user parameters.

Moreover, we specify Linking Rules in the contract to combine contracts elements. Let us consider the example of "availability". Service parameters, such as availability, are not sufficient to evaluate user satisfaction. Availability is generally defined as the percentage of the time that a service is available for use. From the user perspective, customers only care if the service is available when they want to use it. Therefore, service and business parameters are not always appropriate to achieve users' satisfaction. Suppose that there are two service types. One service is working for a month and then stops for one whole day. The other service works for about 8 hours and then there is a 15 minutes downtime. Although the overall availability of both services is equivalent, users may not have the same level of satisfaction for each service. The above example states that the availability parameter described from the service provider is not sufficient to evaluate the user satisfaction. In our contract model, a linking rule is defined to consider both parameters from service level and parameters related to customer satisfaction. Some of more appropriate parameters of availability relate to customers perspective includes: is the service up if the customer wants to use, the actual time that customer spend, maximum waiting time, maximum downtime, meantime between halt and meantime to restore. An integrity constraint could be apply to guarantee that the downtime of the online service shouldn't be higher than the maximum acceptable downtime that user specified in the contract.

The SLA contract is evaluated in the monitoring phase and checked for possible violations from the predefined values of any type of parameters, of the linking rules and of the constraint set up in the contract. According to the source of violation, detected via a diagnosis, an appropriate action is taken to adapt the violated condition to the new values, through an adaptation mechanism. Therefore, if SLA contract is not respected, an adaptation strategy should be taken (e.g., penalties are applied and the service provider is substituted) and new requirement-driven values are set up. The last phase of the method is the contract update which is the reformulation of the contract (or parts thereof) driven by the new conditions and requirements.

Contribution to Cross-Layer Adaptation and Monitoring

The approach provides a novel model of the service-level agreement that integrates elements of different functional layers (namely, BPM layer where the business goals and the user goals are expressed through KPIs and KGIs respectively, and the SI layer), and, therefore, is cross-cutting. This conceptual *cross-layer model* enables description of the contract between user and provider, as well as definition of monitored events and adaptation activities across the layers.

2.5.5 Self-* service infrastructure and the cross layer framework

Usually monitoring information on the service infrastructure layer is fine grained. Thus most of the details about the SBA can be found on the lowest layers. Higher layers usually depend on aggregated information; however, these layers might collect the information to aggregate on their own. This could result in different monitoring systems on the different layers. If the service infrastructure would offer the monitored information towards the higher layers, then these monitoring systems would be able to focus on the information required for their adaptation tasks. This would result in more efficient resource consumption by monitoring.

In case of self-* capable infrastructure layer the services are extended with local monitoring functionality. As a result, they could tell their current health state. The health state varies from service to service; however, the different measurements can be published for the upper layers. The service infrastructure layer collects the monitored properties of every service instance in the application. The upper

layers of the SBA specify then the properties that should generate events for their adaptation strategies.

After the monitoring system of the self-* infrastructure has been fitted in the cross layer framework, we discuss the result of the adaptation strategies initiated by monitoring events. In case of adaptation, a decision in the service infrastructure layer could heavily influence the behaviour of the layers built on top of service infrastructure. Therefore, the infrastructure layer should make adjustments on the infrastructure that are not interfering with the higher layers. This is a basic requirement against the self-* services detailed in JRA-2.3. If an adjustment is needed on the infrastructure layer that could affect the higher layers, then the service infrastructure should notify the upper layers and act according to their decision.

In [29] we introduce an architecture which is planned to be extended with self-* capabilities. This architecture covers the basic infrastructure layer functionality. We also introduce the service level agreements (SLAs) as the main steering forces of the infrastructure. As a matter of fact, SLAs in this architecture could solve the issue of compatibility with higher layers introduced in section 2.3. This way the SLAs sent to the service infrastructure layer cannot be broken unless they are re-negotiated with the service composition or business process management layers. Thus the infrastructure layer will let service instances use their self-* capabilities to make sure the agreed SLA can be met. In case service instances cannot cope with the negotiated SLA, then the infrastructure layer itself could initiate further service-deployments to avoid the violation of the SLA. However in extreme situations the service infrastructure layer could detect that it is not feasible to adapt the resources so that they can meet the SLA requirements. In such cases the infrastructure layer initiates a re-negotiation procedure with the higher layers. During this procedure the infrastructure specifies those SLA requirements that cannot be met, thus higher level adaptation is needed.

As an opposite to the previously mentioned scenario the SLA re-negotiation can also be initiated from the upper layers of the SBA. In such case adaptation is required by the upper layers, and the re-negotiation phase acts as a notification to the service infrastructure. As a result of this adaptation request the infrastructure could for example acquire new resources or even migrate previously used service instances.

Contribution to Cross-Layer Adaptation and Monitoring

The contribution of the architecture presented in this work with respect to the cross-layer A&M framework is twofold. First, it enables alignment of lower level infrastructural events with higher level monitoring systems, thus contributing to the *integrated cross-layer monitoring mechanisms*. Second, it allows for managing and adapting infrastructure components (e.g., the use of resources) according to the requirements posed by the higher levels in terms of SLAs. The adaptations are performed in a way not to change the SLA and, therefore, not to affect the upper layers. In this way, the approach contributes also to the *cross-layer integrated and coordinated adaptation mechanisms*.

2.6 Summary and Conclusions

This chapter defined a baseline for the integration of adaptation and monitoring principles, techniques, and methodologies across functional SBA layers. For this purpose, the functional architecture of the SBAs is described; the adaptation and monitoring approaches identified in the state-of-art survey are mapped to those functional layers. The results of this analysis demonstrate that (i) the research approaches are still fragmented and do not yet cover the adaptation and monitoring of all the elements of the functional SBA layers, and (ii) there is a lack of approaches and methods that integrate and coordinate the monitoring and adaptation activities across layers.

Based on a set of scenarios from the automotive case study [37], the document identifies the set of problems that the SBA monitoring and adaptation may face when the existing solutions are applied independently and in isolation from each other. With the help of scenarios potential problems are illustrated

and the required functionalities and features that the cross-layer approaches and methodologies should provide are identified.

The identified problems and the corresponding requirements have been mapped onto the adaptation and monitoring framework presented in [26]. We have also identified key mechanisms and principles that are necessary for addressing those requirements and, therefore, for providing cross-layer adaptation and monitoring.

Finally, this part of the deliverable summarized the initial research works of the WP participants that aim at addressing the identified requirements and at providing the necessary mechanisms and principles. While these approaches already target different types of the identified mechanisms (SLA-based cross-layer model, integrated cross-layer monitoring mechanisms, means to identify cross-layer adaptation needs, and a support for coordinated adaptation mechanisms), they are still insufficient for providing a holistic and comprehensive adaptation and monitoring framework across functional SBA layers. In the next phases of the project these approaches will be further extended in order to better suit the presented needs. Moreover, new research approaches will target the missing mechanism and approaches complementary to the presented ones. The first results of the integration of these techniques within the scope of the prospective framework presented in this chapter will be presented in the upcoming deliverables of the WP, in particular in Deliverable CD-JRA-1.2.4 “Integrated adaptation and monitoring principles, techniques and methodologies across functional SBA layers”.

Chapter 3

Survey of the HCI- and Context-aware Monitoring Approaches

The main objective of this report is to provide a review of the state of the art about monitoring techniques with respect to *context* and *human computer interaction (HCI)* monitoring. More specifically, we describe and analyze approaches and techniques that have been developed to support context and HCI monitoring.

The analysis of various context factors, models of contexts and HCI aspects has been carried out in the S-Cube Deliverable PO-JRA 1.1.3 [32]. In particular, that document focused on analysing existing context type approaches and culture definitions in order to create a knowledge model of Context and HCI aspects. In this chapter we complement this context and HCI research by i) studying different approaches that support monitoring of different context types and monitoring of human computer interaction with the system; and ii) extending the scope proposed in [32], by considering *passive* context awareness approaches (where context is relevant but not critical) in addition to *active* context awareness approaches (where context is critical to an application).

In this way, the presented survey complements also the survey presented in PO-JRA-1.2.1 [8] by identifying those monitoring approaches that concern about the change in context and HCI aspects and the impact of this change on the system.

3.1 Research Method

We have followed a systematic approach to develop the survey presented in this chapter. Our aim was to produce this survey as a complementary to the survey presented in PO-JRA-1.2.1 [8]. In other word we tried to cover the monitoring approaches that concern about the change in context and the impact of this change on the system that are not covered in PO-JRA-1.2.1. The systematic approach that we followed to produce this survey comprises the following simple steps:

1. We conducted a systematic review of major academic and industrial publications in the area of software monitoring and adaptation. From the reviewed articles we identified the approaches that mainly deal with context monitoring and human computer interaction monitoring. We tried to review publications appear in major conferences, journals, magazines or digital libraries in the domain of software engineering, human computer interaction, virtual reality and robotics that were published in or after year 1994. Specifically we reviewed articles in the following publications:

    - **Conference /Workshop Proceedings**: Intl. con. on Mobile HCI (MobileHCI), Intl. con. on HCI (CHI), Intl. con. on Humanoid Robotics (HUMANOID), Intl. con. on Intelligent User Interface (IUI), Intl. con. on Information Systems (ECIS), Intl. con. on Mobile Systems (MOBSYS), Intl. con. on Artificial Intelligence (IJCAI), Intl. con. on Communication and Networking (CHINACOM), Intl. Conf. on Human-Computer Interaction with

Mobile Devices (MobileHCI), Intl. conf. on Requirements Engineering (RE), Intl. conf. on Mobile Computing Systems and Applications (HotMobile), Workshop on User Interfaces for All (UI4ALL), Ten Competence Workshop (TENCompetence), Workshop on Software Engineering and Middleware (SEM), Workshop on Adaptive Design of Interactive Multimedia Presentations for Mobile Users, Symposium on Handheld and Ubiquitous Computing (HUC), Symposium on User Interface Software and Technology (UIST);

- **Journals/Magazines**: Journal of Computers in Human Behavior, International Journal of Human-Computer Studies (IJHCS), Journal of Communication Engineering (JCM), IEEE TSE;
- **Digital Libraries**: ACM Digital Library, IEEE Digital Library, Distrib Parallel Database;
- **Project Deliverables/Technical Reports**: S-CUBE, Service Centric Software Engineering (SeCSE), Irish Software Engineering Research Centre (Lero), Telecooperation Office (TecO), International Federation for Information Processing (IFIP).

2. We separated the identified papers in step (a) into two groups, namely context monitoring and HCI monitoring. We applied the context knowledge model and human factors presented in PO-JRA-1.1.3 to separate the papers into these two groups. We further provided a classification of the works presented in each group based on some classification categories where these classification categories are motivated by the monitoring and adaptation taxonomy presented in CD-JRA-1.2.2 [26].

3.2 Context Monitoring

Context is defined as "interrelated conditions in which something exists or occurs" [1]. It characterizes the state of a certain *entity* by the identification of all factors surrounding the entity. Several definitions and characterizations of context have been found in the literature and based on these definitions different context models have been proposed. In [18] operational definition and developing concerns are given, in [46, 45] a hierarchical model is proposed to structure the concept of context, in [11] an evaluation framework for context models and applications are presented, and in [15] different context types and models are analyzed. Additionally several classifications of context have been given in [25, 17, 34, 46, 45] mainly based on general aspects involving an entity (or user) and its environment. In what follows, we analyze different context monitoring approaches that are identified based on these definitions and characterizations of context. In this section we complement the survey presented in PO-JRA-1.2.1 [8] by identifying those monitoring approaches that concern about the change in context and the impact of this change on the system. We classify the identified approaches in term of (i) context acquisition, (ii) context modeling, (iii) adaptation of the system and (iv) architecture of the system.

3.2.1 Existing Context Monitoring Approaches

In [28] a context monitoring platform called Seemon is presented. The main aim of this platform is to reduce the expensive computation and communication costs in context monitoring for mobile devices. This reduction is achieved by translating high level context application queries into lower level queries in order to optimize the acquisition of relevant context. For example, consider the query "if person P is studying a subject X in a place Y at a time Z" – could be easily answered if no person is at the place Y. In other words in a context aware system, a single context type (in this case *place*) could determine an outcome of a query avoiding the consideration of all other contexts involved in the query. In this approach, context information is obtained through physical sensors measuring features such as energy, temperature, speed, or user's heart rate. The approach uses an intuitive monitoring query language that is similar to SQL and supports rich semantics for a wide range of contexts. It also contains a processor to

evaluate different queries over a continuous stream of sensor data and a *sensor manager* that dynamically controls sensors to avoid unnecessary data transmissions. Its job is to find minimal sets of sensors needed to answer a query. The communication and management between the platform and the applications are done trough a special components, called the *application broker*. The use of a query language capable of specifying different conditions in a SQL-type notation simplifies the generation of conditions when generating rules for the behavior, since the application broker automatically performs all the translation work. This is associated with the selection of sensors based on relevant context from a set of physical sensors in context monitoring. However, the use of physical sensors is not adequate for monitoring of non-physical context aspects such as the skills or knowledge of a user. Moreover, the use of several different sensors may cause complexity for the queries.

In [16] a comparison between pull and push approaches for context-aware tourism information systems, is proposed. The authors advocate that in pull approaches the user is aware and in control of the system's input and, therefore, the user is responsible to deliver contextual information. In push approaches, on the other hand, the user is relieved from the responsibility of delivering contextual information since this information is triggered by events produced by sensors. Although not explicitly mentioned in the paper, in push approaches, monitoring is performed in order to capture the events triggered by contextual change, while in pull approaches, the behavior of the system varies only with the user's context input. These approaches are not mutually exclusive. For example, the pull approach could help define general context types in order to use a push approach for each of these general context types. For instance, once the location is explicitly set by the user in a theme park, an advance touristic guide system could suggest attractions based on the attraction's estimated adrenaline levels, the user's heart rate, and the specified location in the park. While the pull approach monitoring assumes a simpler context acquisition (directly from the user); it is more prone to errors, for example, a user forgets to change his location.

In [40] a software platform called ContextPhone is described. The platform consists of four interconnected modules namely sensors, system services, communications and customizable applications. In this study the focus is on mobile devices and its interaction with the environment. It is assumed that context information can be sensed, processed, stored and transferred within the mobile device or outside it. This is done by physical and logical sensors grouped into specific types relevant to mobile devices such as location, user interaction, communication behavior and physical environment. This grouping of physical and logical sensors into specific relevant types is actually a context sub type classification that aims to characterize context relevant factors when developing a mobile device application. The sensors type classification is general enough to allow flexibility when developing applications. The platform allows a design, based on the available sensing capabilities of a device, which includes context monitoring capabilities. However, due to the nature of the sensing devices and the fact that interaction is focused on the device and the environment, monitoring is performed over environmental context types and no user context is considered.

In [9] an approach for monitoring distributed context to support Internet services is described. The approach is based on an architecture [2] consisting on three key entities, namely a user, an operator, and a service provider. For each of these entities a set of context parameters is specified in a defined notation, generating what is called a profile manager for each of these entities. It is also possible to define rules, in order to report when a change in context information has occurred for a defined entity. When a change in context happens, it is reported to a special component called context provider that evaluates the change and contacts the service provider by using pre-defined policies. The approach not only manages distributed context but establishes, trough a series of mechanisms, when to perform an action based on defined policies. A change of context that according to the rules does not affect the service is just ignored. For each of the profile managers a monitor is in charge of processing the entities events. Also for each profile manager the same type of monitor operates using defined parameters and notation. However the rules established for each monitor cannot be applied to different ones since they

are based on a specific context, even if the rules are expressed in the same language. Therefore the reuse of rules for similar situations, but different contexts is not possible. The paper does not describe how context information can be obtained or measured. In order to apply this approach it is necessary to have a complete knowledge of the involved context.

In [24] a lightweight approach to context-sensitivity considering the middleware limitations in dynamic mobile environments is described. The approach proposes general guidelines that should be followed in a context-aware application development. The work aims to provide application developers with *access* to context information through simplified interfaces that facilitate the programming tasks and construction of context-aware systems in resource-constrained devices. In the approach, context information is accessed through two components namely sensing and *sensor monitoring*. The approach assumes the existence of a special element called host, which contributes to context information in a network, with concrete monitors and sensors. The sensing component allows software systems to communicate with sensing devices connected to a host, while the sensor monitoring component keeps a registry of monitors available on the hosts.

The idea behind this is to make the services available on a host accessible when building monitors, providing unified functionality and methods to get values and react to changes. Thus a developer could use available monitors or create his own. When using the available monitors it is not possible to specify the sensing device from which the information is obtained. In this case, all the potential devices delivering information should be considered at development time. In this approach context monitoring is analyzed from the developer's point of view. It simplifies the process dealing with context awareness offering developers a "familiar" way to deal with monitors. No classification or distinction is mentioned referring to the different context types and since all of the sensing devices mentioned in the study were related to positioning, we assume that the focus in the approach was oriented towards a location context.

In [33] the authors present an approach to support context recognition and forecast. The work is based on the analysis of mobile devices equipped with different types of sensors, such as time, brightness, and Bluetooth. In this work, context is not obtained directly from the sensors, but from states of an abstract state machine. These states represent the result of the different sensor readings at a defined moment. For example for a certain time t_1 a state s_1 is available based on the monitored sensors, if in a subsequent time t_2 a change in a sensor occurs, a new state s_2 will be active. More specifically, the extraction of context starts from the monitored raw data obtained from the sensors. This data is represented as a vector that is classified into clusters (classes), representing common patterns, in a multi-dimensional feature space. The approach uses the notion of "class clustering" that allows grouping several situations, determined by sensors and represented by vectors, into defined classes according to the probability that the feature vector belongs to a defined class. Forecast is executed by predicting other vectors base on the current one. A vector could be assigned to several classes; therefore it would be helpful to assign descriptive names to the classes and to classes' combinations in order to ease the identification of context at development time. This is a manual labeling process. The main drawback is that the inclusion of a new sensor could trigger significant changes, modifying not only the labeling but the cluster as well. In the approach, monitoring is performed continuously in order to detect changes in sensors.

A study focused on wireless networks characterized by dynamicity, heterogeneity and mobility is presented in [38]. In order to manage a distributed network in such heterogeneous environment the authors propose the use of context awareness to allow self-adaptation of the network based on distributed analysis performed in each node. The study considers a distributed architecture that allows efficient, scalable, and distributed management operations (based on *Echo Pattern* [43]) across a network. An administrator is in charge of context sensors deployment, which monitor the network on each node. The context sensors used in the approach are self-contained highly flexible software components that can monitor specific context types such as QoS or status in a node. In this approach, the context sensors perform both

activities of sensing and monitoring. Finally context sensors are not possible to be modified. Every time, a re-configuration for a context sensor is required, a new context sensor replaces the original.

In [42] an approach is proposed to represent and reason about physical contextual variability (mainly location) and its impact on the requirements of a system where the contextual variability may require the system to adapt in the changed context to meet the system requirement. The work is based on the "problem" frame notation [27, 19] for describing a problem in three different sets: a description of the context in which the problem resides in terms of domain properties, the required properties of the domain, and what a system must do to meet the requirements. Context of the problem, required properties, and expected behavior are related in different ways depending on the variables involved (e.g. a variation in context may cause a requirement to be no longer satisfied by a specification of what the system must do). To capture the behavior in a problem state machines are used, where a satisfied requirement is presented as a final states and an unsatisfied requirement is presented as an error state. The approach considers a scenario with separate physical domains where context variations occur. However, in scenarios where domains do not restrict to physical components the approach may not be suitable.

A proposal for discovery and execution of web services by using contextual agents is presented in [10]. In the proposal each agent is able to monitor the user context and the services capabilities so that relevant web services are made available, or suggested to users. A service for a user is made available or suggested by an agent based on rules defined in a *user profile*. These rules deal with three main domains related to context: *user role* that describes the nature of the user's task, user action type and information that describes a specific action, and the *information type* that delivers information of the local user. An agent searches for services and information relevant to the users, based on the user profile. In the approach, rules are either defined by the user or predefined by the agents based on rules for other users. Profiles are updated based on user's preference when a particular service capability is not defined. While this is adequate for discovery and execution of web services, it is not well suited for monitoring, since a considerable number of irrelevant preferences could arise. Despite the problem with the rules generation, the approach deals with context definition and acquisition in a novel manner. More specifically the contextual agent collects information from the user's interaction with the system, correlates the collected information to generate a query. For example if a user is repeatedly searching a product X in the Internet, the agent can generate a query based on different information of the product X (e.g. price, color) and execute the query to find a web service that provides relevant information of the product X. Moreover, to the best of our knowledge this is the only approach dealing with web services context monitoring.

3.2.2 Classification

In this section we provide a classification of the context monitoring works presented above. We base our classification on the various implementation aspects and the application of the surveyed approaches. More precisely we discuss the works according to i) Context Acquisition, ii) Context model, iii) Adaptation and iv) Architecture. It should be noted that these categories are motivated by the monitoring and adaptation taxonomy presented in CD-JRA-1.2.2 [26]. For example, categories *context acquisition* and *architecture* comply with the *How* dimension presented in the monitoring taxonomy in CD-JRA-1.2.2 and the category *context model* comply with the *What* dimension presented in the monitoring taxonomy in CD-JRA-1.2.2. Again, the category *adaptation* complies with the *What* and the *How* dimensions presented in the adaptation taxonomy in CD-JRA-1.2.2. In the end of this subsection, we present a table summarizing the classification of the context monitoring approaches with respect to categories (i) to (iv) above.

- *Context Acquisition*: This category refers to the mechanism that has been used to collect context information for the monitoring process. It is found in the literature that context information can be collected broadly in tow ways and these are,

Table 3.1: Classification summary of context monitoring approaches

Approach	Context Acquisition		Context Model		Adaptation		Architecture	
	Physical Sensor	*Logical Sensor*	*Attr./Val. Pair*	*Structured Language*	*Adaptation of System*	*Adaptation of Monitor*	*Distributed Middleware*	*Non-distributed*
[28]	X			X		X	X	
[16]	X	X	X		X		X	
[40]	X	X	X					X
[9]	X	X	X		X		X	
[24]	X	X	X				X	
[33]	X	X		X	X			X
[38]		X	X			X	X	
[19, 42]	X			X	X			X
[10]		X	X				X	

  - Physical sensors are hardware devices that collect context information directly from the environment of the monitored system (e.g. temperature, brightness etc.) [28, 19, 42]
  - Logical sensors are implemented as software module that may compute context information based on the context information collected by the physical sensors (e.g. average temperature) or collect context information by polling system parameters (e.g. battery level of PDA), or collect context information from user profile [10, 38]. However, some context monitoring approaches apply both methods (i.e. physical and logical sensors) to collect context information [9, 16, 24, 33, 40].

- *Context Model*: This category refers to the technique that has been used to define and store context information in the presented works. Context can be modeled in many ways including,
  - Simple attribute/value pairs with predefined semantics of the attributes and possible set of values. This approach allows to express logical conditions on the attributes [9, 10, 16, 24, 38, 40],
  - Structured language that based on some formalism (e.g. predicate calculus based language can support application of Boolean algebra) [28, 19, 42, 33].

- *Adaptation*: This category refers to system support for any type of adaptation due to context change. Two types of adaptation are found in the literature and these are,
  - Adaptation of the monitored system where the monitored system can adapt itself according to some predefined set of policies while a change in the context is identified by the monitor. For example in case of continuous video streaming, if the monitor detects a fall in available bandwidth then the system can set the video rendering quality to a lower level [9, 16, 33, 19, 42].
  - Adaptation of the monitor where the monitor can adapt itself with respect to a change in the context. For example the monitor may start monitoring a new set of rules when the room temperature rises above a certain threshold [28, 38].

- *Architecture*: We use this category to discuss the implementation architecture of the presented work. A context monitoring system can be implemented in many ways including,
  - Middleware between application and the environment of the application, where the middleware collects context information from the environment and evaluates the context conditions set by the application and returns the evaluation result to the application [28, 10, 24, 9, 16, 38].
  - Non distributed architecture where context information is gathered by the application directly from the environment through sensors or input devices [33, 40, 19, 42].

3.3 HCI Monitoring

In this section we present a survey on the Human Computer Interaction (HCI) monitoring. Through this survey we complement the survey presented in PO-JRA-1.2.1 [8] by identifying those monitoring approaches that concern about the interaction of the user with the system. We classify the identified approaches in terms of (i) event acquisition, (ii) user modeling, (iii) adaptation of the system and (iv) architecture of the system.

3.3.1 Existing HCI Monitoring Approaches

In [20] a novel method using time series analysis for detecting interaction styles in a human-robot relation is presented. It consists of an algorithm, based on a clustering method for the extraction of relevant information, that recognizes on real time pre defined tactile interaction styles between a human and a robot, based on time serialized input signals. Styles are defined as behaviors between the user and the machine (e.g. rudeness in the interaction or frequency), and the algorithm is able to recognize and classify such behaviors in a reduced period of time. The relevant information extraction is possible due to the fact that as signals occur they are grouped into a defined length set for pattern recognition using a previous set. Finally it is claimed by the authors that the algorithm could enable real time adaptation of machines to interaction styles. Although it is not explicitly mentioned monitoring, is being performed constantly every time a set of grouped signals is compared to another one. Even more, the statement made by the authors implies that monitoring of the user-machine behavior is constantly performed. The main drawback with this approach is its need of constant human machine tactile interaction. A more sporadic interaction could be not appropriate for the clustering method and, even worse, a single signal, or event, would be almost impossible to recognize based on the patterns.

HCI monitoring is used to design systems in such a way that system may adapt itself based on the interaction of the user with the system and assists the user to accomplish his task more conveniently. In [21] an automated lecture facility system, called Classroom, is presented. This system watches and listens to its user (i.e. a lecturer in a classroom) and when appropriate assists the user. Classroom also produces video feed suitable for distance learning. For example, if the lecturer starts writing on the board the Classroom automatically focuses its presentation camera on the words and figures been written. In this approach the Classroom represents the activities taking place as a *process* that contains a sequence of actions and the Classroom keeps its process set synchronized with what is actually happening by comparing the sequence of actions in the process against the set of events collected through the sensing systems.

In [31] an architecture, called SUITOR, is presented that monitors user actions in a system and provides appropriate suggestions to the user. For example, user A accessing a web based application that helps to compare prices of different products. If SUITOR monitors that A is always comparing prices of the products only from company X and Y, ignoring all other companies, then SUITOR may infer that A is interested only in the products of X and Y, and thereby the application can show only product information from companies X and Y to A, and hide product information from other companies. In this approach a user is modeled as a list of keywords where keywords are derived from user interactions (e.g. keyboard input, user email, web pages read) and then a list of keywords is produced by determining the relative frequency of the original keywords. The system also gathers information from the world outside of SUITOR (e.g. observing other running applications). This information is compared with the user model and rated according to how much it overlaps with the user model.

In [30], a user interface agent, called Letizia, is presented that assists a user to browse the World Wide Web. Letizia tries to approximate the user's interest by tracking users' browsing behavior. It applies a set of heuristics to model users' browsing behavior. For example if a user follows a link and spends

a considerable time in the document then an indication of interest to that document could be assumed, while if the user leaves the document immediately then an indication of disinterest can be assumed. Again if a user visits a document repeatedly then it indicates the user's interest to that document. By analyzing the user interest Letizia complies a set of recommendations for the user that the user can choose either to follow or reject.

In [13] a system that monitors information requested and accessed by users in browser applications has been proposed. The main goal of the monitor is to analyse user's interaction with the browser and proactively suggest the user some useful resources. The information sources as well as the applications involved are connected to the monitor system via APIs, and special adaptors for each application are available in order to exchange information. The approach considers context with respect to tasks performed by a user. More specifically, information that has been used by a user could serve as a guide for another user. This however involves a privacy violation since the information required and accessed by a user is kept in the system and made it available for other users.

HCI monitoring is also exploited to produce indicative feedback for the learners in informal learning environment (e.g. web based educational blogging or collaborating writings) [23]. The approach proposes a four layer architecture to generate feedbacks. In the first layer a sensor service monitors the learner's interaction with the system. In the second layer an objective view is created based on the learner's actions. In the third layer this objective view is contextualized regarding the learning situation and learning process and in the fourth layer the generated feedback is reported to the learner.

Various applications of implicit HCI is discussed in [44]. Implicit HCI is defined as any action performed by the user where the user is not intended to interact with the system but the system can consider such action as an input along with the explicit input to the system. For example a system may automatically switch on the light when a user is walking in a dark corridor. Here the user is walking in the corridor and not intended to interact with the system but the system considers this user action as an input along with other explicit input (i.e. brightness in the corridor) from the light sensor. As described in this work, such systems relies on two major concepts, i) perception, i.e. the ability to perceive the environment and circumstances using sensors and ii) interception, i.e. the mechanics to understand what the sensors see, hear and feel. The paper argued that implicit HCI can be applied to improve the input/output capabilities of small appliances (e.g. PDA, mobile phones) as such appliances often suffer to offer optimal input/output capabilities due to lack of spaces.

In [3] the approach described analyzes the user interactions, compares them with the expected model of the user activities and reports the corresponding feedback. The expected model of the user activities is represented as a task model that specifies the hierarchical and sequential structure of tasks that should be performed to achieve user's goal. Task models are specified using a description language that provides formal syntax and semantics for creating task models. The framework contains an event database and a handler that manages user events from instrumented applications. These events can be low level system events (e.g. a mouse click or move) or higher level application events (e.g. selection of a menu item). A task monitor receives events from event handler and monitors the users' progress by matching the events to the user task model. As a consequence of this monitoring process the task monitor notifies the user level services of user task related events e.g. starting or finishing a task.

In [39] a study is presented to generate, at run-time, homogeneous and coherent user interfaces independent of the device utilized by the user for the interaction. In the approach user interface (UI) components representing small widgets are associated with business components which contain application logical parts. The construction of an interface is based on the merging/unmerging of (UI) components dynamically according to the business components assembly. This allows to adapt the interface according to the business constraints. The components for interaction and an interaction server are prepared by an

interface service. The interaction server is in charge of binding/unbinding components. Thus when a business component is used, the interaction service detects it and generates the adequate interface.

In [14, 45, 48] the authors propose a framework for multi-target user interface (UI) design. A tool (ARTStudio) is proposed to serve as an instrument to help designers and developers to structure the development of plastic interactive systems. The idea is to design/develop models being used in current practice and improve them according to the variations in the "context" (e.g. visualization from a mobile phone, visualization in a PC). For this purpose different descriptions are made specifying concepts models, task models, platform models, environment models, and user models. Concepts models describe the entities and relation between entities the user manipulates. Task model describes how the user reaches his goals. The platform, environment and user model define the platform environment and user the UI should cover. During the development process these descriptions are referenced. The development process is a combination of vertical reification and horizontal translation. A vertical reification is applied, between the different models, for a particular target while translation is used to express bridges between the descriptions for different targets (e.g. a mobile phone and a palm). Rules can be expressed in order to deal with presentation issues occurring at run-time.

In [12] a study is performed in the area of physiological monitoring, in particular face expression recognition. Its aim is to establish the relation between the user affective states (positive or negative) and a defined task to be performed composed of different activities with different levels of complexity. The monitored events are captured using specialized sensors for muscles recognition during a defined task. The study is presented as a complement when monitoring affective responses, since user's facial recognition is part of physiological monitoring in HCI.

3.3.2 Classification

In this section we try to provide a classification of the HCI monitoring works presented above. We consider almost the same categories that we used in the previous section to discuss context monitoring approaches, except the categories context acquisition and context model. Instead of the categories context acquisition and context model we use the categories *event acquisition* and *user model* respectively. This is because HCI monitoring process compares the events produced through the user's interaction with the system against the expected model of the user activities. A table summarizing the approaches with respect to the classification is presented in the end of this subsection.

More precisely we discuss the works according to the following categories:

- *Event Acquisition*: This category refers to the mechanism that has been used to collect runtime events for the monitoring process. In case of HCI monitoring runtime event stream is produced by users interacting with the system and the monitoring process may collect the events through various means including,
  - By instrumenting the application or applying an adapter to the application the user is interacting with [3]
  - Physical sensors can be used to collect user activities (e.g. movement of a user in a room, eye gaze of user) [12, 20, 21, 23, 31, 44],
  - Events can be captured from the interactions a user makes through the input devices of the system (e.g. keyboard typing, or mouse click) [13, 14, 30, 39, 47, 48]. However some approaches apply both the physical sensors and input devices to collect runtime events [31].
- *User Model*: This category refers to the technique that has been used to define the activities that the user should perform to achieve the goal. User model mainly specified in two ways and these are,

Table 3.2: Classification summary of HCI monitoring approaches

Approach	Event Acquisition			User Model		Adaptation		Architecture	
	Application Instrumentation	*Physical Sensor*	*Logical Sensor*	*Keyword List*	*Structured Language*	*Adaptation of System*	*Feedback to User*	*Cleint Server*	*Agent-based*
[21]		X			X	X			
[31]		X	X	X			X		X
[30]				X			X	X	
[13]			X	X			X		X
[23]		X			X		X	X	
[44]		X			X	X		X	
[3]	X				X		X	X	
[39]			X		X	X			
[47, 48]			X		X	X		X	

  - List of simple keywords, where each keyword signifies a user activity. In this approach key words are defined in terms of the basic user interactions (e.g. keyboard input, user email, web pages read) and then the lists are created by analyzing the relative frequencies of the keywords [30, 31, 13].
  - By using structured description language that specifies formal syntax and semantics to express user activities [3, 14, 21, 23, 39, 44, 47, 48].

- *Adaptation*: This category refers to system support for any type of adaptation due to user's interaction with the system. It is found in the literature that based on the HCI monitoring result a system can offer two types of adaptation and these are

  - The system can adapt itself to assist the user in achieving the goal. For example in an automated distant learning setup when the monitor detects that the lecturer is about to start speaking the system can turn on the audio transmission, or a system may adapt its graphical interface based on the user interaction [14, 21, 39, 44, 47, 48],
  - The system can suggest the user a set of activities that the user may be interested in to do next. For example, if a user repeatedly visits a web page then the system may suggest similar web pages for the user based on the content of the current webpage [3, 13, 23, 30, 31].

- *Architecture*: We use this category to discuss the implementation architecture of the presented work. A HCI monitoring system can be implemented in many ways including,

  - Client server architecture where the client provides a front end to the user and collects the user interaction and the server acts in the backend as the reasoning engine that receives the events and compares the events with available user models [3, 14, 23, 30, 44, 47, 48]
  - Agent based shared repository architecture where different agents collect runtime events and store in the shared repository and the monitoring agents access the shared repository to perform the monitoring process [13, 31].

3.4 Summary and Conclusions

In this chapter we presented a survey of the monitoring techniques with respect to *context* and *human computer interaction* (HCI). It should be appreciated that both HCI and context monitoring are most of the time strongly related, thus their differentiation could be some times quite hectic. In this survey we separated the approaches into two groups, namely (a) context monitoring and (b) HCI monitoring. In the first group, we described approaches concerned with different types of context. More specifically, we considered the types of context in the context knowledge model presented in deliverable PO-JRA-1.1.3 [32] (computing, physical, time, and location context types) together with user context type. In the

second group, we described approaches concerned with monitoring of the interaction of the user with the system.

Context monitoring and HCI monitoring can facilitate the overall objective of SBAs monitoring to a great extent. This is because, monitoring of different types of properties (e.g. security, or reliability) of SBAs often depends on the surrounding situation of the SBA system or its user (e.g. in ATM systems security measures need to be tightened at the evening hours). Appropriate characterization of SBA context and monitoring of SBA context may help to identify surrounding situations of the SBAs or its user. Several definitions have been proposed in the literature to characterize context [25, 17, 46, 45]. These proposals consider a general classification for general context types, including categories dealing with *location* and *user* context. However in this general classification the different proposals overlap or are incompatible with each other; for example in [45] there is a clear differentiation between physical environment (e.g. location, surrounding resources for computation, temperature) and human factors (e.g. habits, group dynamics, goals), while in [25] location is included in the user context. Moreover each of the proposals tend to define a reduced set of different context types covering all possible situations, making it too general to be applied to a specific situation (e.g. browsing a web page from a desktop or a mobile would not be much different according to general context types). Therefore a general classification of context as found in the literature is most likely unfeasible for SBAs due to the wide range of situations covered by SBAs.

Automatic identification of relevant context of SBAs and its user should be a major concern in case of monitoring SBAs. Manual identification of relevant context of SBAs would force the users to express all the information relevant to a given situation, which would be difficult and tedious for the users. Moreover users may not know which information is potentially relevant to a situation. Most of the approaches discussed above ignore the automatic identification of relevant context of a system apart from the approach in [17]. Still in this approach automatic identification of relevant contexts depends on a predefined set of contexts, which is not suitable for SBAs due to its wide scope.

Adaptation of monitoring process due to change of context of SBA or its user is another issue that should be considered in case of SBA monitoring. For example, if the monitor detects that the available memory drops below a certain level then a new instance of memory can be deployed to a remote machine. Only a few approaches [28, 38] found in the literature concerned about this type of adaptation and that reveals a gap between the objective of this work package and the existing approaches.

An appropriate user model plays a significant role in case of HCI monitoring, as the user model allows for representing the properties of a user. Most of the approaches covered in this survey, model users in terms of a list of simple keyword (i.e. the activities should be performed by user), or user profile which is simple description of users' preferences and history. However effective monitoring of HCI may require that a user model should contain more detail information (e.g. user's level of expertise, skills etc. and their inter relation) about user which is not addressed in the literature.

Bibliography

[1] Merriam-webster online dictionary http://www.m-w.com/.

[2] Alessandra Agostini, Claudio Bettini, Nicolò Cesa-Bianchi, Dario Maggiorini, Daniele Riboni, Michele Ruberl, Cristiano Sala, and Davide Vitali. Towards highly adaptive services for mobile computing. In *Mobile Information Systems, IFIP TC 8 Working Conference on Mobile Information Systems (MOBIS), 15-17 September 2004, Oslo, Norway*, pages 121–134, 2004.

[3] B. P. Bailey, P. D. Adamczyk, T. Y. Chang, and N. A. Chilson. A framework for specifying and monitoring user tasks. *Computers in Human Behaviour*, 22:709–732, 2006.

[4] Fabio Barbon, Paolo Traverso, Marco Pistore, and Michele Trainotti. Run-Time Monitoring of Instances and Classes of Web Service Compositions. In *IEEE International Conference on Web Services (ICWS 2006)*, pages 63–71, 2006.

[5] Luciano Baresi and Sam Guinea. Towards Dynamic Monitoring of WS-BPEL Processes. In *Service-Oriented Computing - ICSOC 2005, Third International Conference*, pages 269–282, 2005.

[6] Luciano Baresi, Sam Guinea, Raman Kazhamimakin, and Marco Pistore. An integrated approach for the run-time monitoring of bpel orchestrations. In *ServiceWave*, 2008.

[7] Luciano Baresi, Sam Guinea, Michele Trainotti, and Marco Pistore. Dynamo + astro: An integrated approach for bpel monitoring. In *ICWS'09*, 2009.

[8] Salima Benbernou, editor. *State of the Art Report, Gap Analysis of Knowledge on Principles, Techniques and Methodologies for Monitoring and Adaptation of SBAs*. S-Cube project deliverable, July 2008. S-Cube project deliverable: PO-JRA-1.2.1. http://www.s-cube-network.eu/achievements-results/s-cube-deliverables.

[9] Claudio Bettini, Dario Maggiorini, and Daniele Riboni. Distributed context monitoring for the adaptation of continuous services. *World Wide Web*, 10(4):503–528, 2007.

[10] Brian Blake, Daniel R. Kahan, and Michael Fitzgerald Nowlan. Context-aware agents for user-oriented web services discovery and execution. *Distrib. Parallel Databases*, 21(1):39–58, 2007.

[11] Cristiana Bolchini, Carlo A. Curino, Elisa Quintarelli, Fabio A. Schreiber, and Letizia Tanca. A data-oriented survey of context models. *SIGMOD Rec.*, 36(4):19–26, 2007.

[12] Pedro Branco and L. Miguel Encarnacao. Affective computing for behavior-based ui adaptation. In *IUI workshop*, 2004.

[13] Jay Budzik, Shannon Bradshaw, Xiaobin Fu, and Kristian J. Hammond. Supporting on-line resource discovery in the context of ongoing tasks with proactive software assistants. *Int. J. Hum.-Comput. Stud.*, 56(1):47–74, 2002.

[14] Gaelle Calvary, Joelle Coutaz, and David Thevenin. Embedding plasticity in the development process of interactive systems. In *In 6th ERCIM Workshop User Interfaces for All. Also in HUC (Handheld and Ubiquitous Computing) First workshop on Resource Sensitive Mobile HCI, Conference on Handheld and Ubiquitous Computing*, 2000.

[15] G. Chen and D. Kotz. A survey of context-aware mobile computing research. Technical report, Hanover, NH, USA, 2000.

[16] Keith Cheverst, Keith Mitchell, and Nigel Davies. Investigating context-aware information push vs. information pull to tourists. In *In Proceedings of Mobile HCI 01*, page 2001, 2001.

[17] A. K. Dey and G. D. Abowd. Towards a better understanding of context and context-awareness. In *Proceedings of the Workshop on the What, Who, Where, When and How of Context-Awareness*, 2000.

[18] Anind K. Dey. Understanding and using context. *Personal Ubiquitous Comput.*, 5(1):4–7, 2001.

[19] M. Salifu et al. Using problem descriptions to represent variability for context-aware application. Technical report, Limerick, Ireland: Lero, 2007.

[20] D. Francois, D. Polani, and K. Dauenhahn. Towards socially adaptive robots: A novel method fro real time recognition of human-robot interaction styles. In *8th International Conference on Humanoid Robots*, 2008.

[21] David Franklin, Jay Budzik, and Kristian Hammond. Plan-based interfaces: Keeping track of user tasks and acting to cooperate. In *In International conference on intelligent user interfaces*, pages 79–86. ACM, 2002.

[22] Mariagrazia Fugini and Hossein Siadat. SLA Contract for cross-layer Adaptation and Monitoring, 2009. to be submitted.

[23] C. Glahn, M. Specht, and R. Koper. Towards a service-oriented architecture for giving feedback in informal learning environments. In *2nd TenCompetence Workshop*, 2007.

[24] Gregory Hackmann, Christine Julien, Jamie Payton, and Gruia-Catalin Roman. Supporting generalized context interactions. In *Workshop on Software Engineering and Middleware (SEM 2004)*, 2004.

[25] T. Haedrich and T. Priebe. Supporting Knowledge Work with Knowledge Stance-Oriented Integrative Portals. In *Proceedings of the Thirteenth European Conference on Information Systems*, 2005.

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

[27] Michael Jackson. *Problem frames: analyzing and structuring software development problems*. Addison-Wesley, 2001.

[28] Seungwoo Kang, Jinwon Lee, Hyukjae Jang, Hyonik Lee, Youngki Lee, Souneil Park, Taiwoo Park, and Junehwa Song. Seemon: scalable and energy-efficient context monitoring framework for sensor-rich mobile environments. In *Proceedings of the 6th International Conference on Mobile Systems, Applications, and Services (MobiSys 2008)*, pages 267–280, 2008.

[29] Attila Kertesz, Gabor Kecskemeti, and Ivona Brandic. An sla-based resource virtualization approach for on-demand service provision. In *VTDC 2009 - The 3rd International Workshop on Virtualization Technologies in Distributed Computing. In conjunction with the 6th International Conference on Autonomic Computing and Communications Barcelona*, 2009.

[30] Henry Lieberman. Letizia: An agent that assists web browsing. In *International Joint Conference on Artificial Intelligence*, pages 924–929, 1995.

[31] Paul P. Maglio, Rob Barrett, Christopher S. Campbell, and Ted Selker. Suitor: an attentive information system. In *IUI '00: Proceedings of the 5th international conference on Intelligent user interfaces*, pages 169–176, New York, NY, USA, 2000. ACM.

[32] Neil Maiden, editor. *Codified Human-Computer Interaction (HCI) Knowledge and Context Factors*. S-Cube project deliverable, March 2009. S-Cube project deliverable: PO-JRA-1.1.3. http://www.s-cube-network.eu/achievements-results/s-cube-deliverables.

[33] R. Mayrhofer, H. Radi, and A. Ferscha. Recognizing and predicting context by learning from user behavior. *Radiomatics: Journal of Communication Engineering, special issue on Advances in Mobile Multimedia*, 1(1):30–42, May 2004.

[34] K. Mitchell. *Supporting the Development of Mobile Context-Aware Computing*. PhD thesis, Department of Computing, Lancaster University, 2002.

[35] Brice Morin, Thomas Ledoux, Mahmoud Ben Hassine, Franck Chauvel, Olivier Barais, and Jean-Marc Jezequel. Unifying Runtime Adaptation and Design Evolution. In *CIT*, 2009.

[36] E. Di Nitto, D. Karastoyanova, and A. Metzger et al. S-Cube: Addressing Multidisciplinary Research Challenges for the Internet of Services. In et al G. Tselentis, editor, *owards the Future Internet: A European Research Perspective*, 2009.

[37] Elisabetta Di Nitto, Valentina Mazza, and Andrea Mocci, editors. *Collection of industrial best practices, scenarios and business cases*. S-Cube project deliverable, March 2009. S-Cube project deliverable: CD-IA-2.2.2. http://www.s-cube-network.eu/achievements-results/s-cube-deliverables.

[38] R. Ocampo, L. Cheng, K. Jean, A. Gonzalez-Prieto, A. Galis Z., and Lai. Towards a context monitoring system for ambient networks. In *IEEE CHINACOM*, 2006.

[39] Anne-Marie Pinna-Dery, Jérémy Fierstone, and Emmanuel Picard. Component model and programming: A first step to manage human computer interaction adaptation. In *Human-Computer Interaction with Mobile Devices and Services, 5th International Symposium, Mobile HCI 2003*, pages 456–465, 2003.

[40] M. Raento, A. Oulasvirta, R. Petit, and H Toivonen. ContextPhone: a prototyping platform for context aware mobile applications. *IEEE Pervasive Computing*, 4(2):51–59, 2005.

[41] Raman Kazhamiakin and Marco Pistore and Asli Zengin. Cross-layer Adaptation and Monitoring of Service-Based Applications, 2009. to be submitted.

[42] Mohammed Salifu, Yijun Yu, and Bashar Nuseibeh. Specifying monitoring and switching problems in context. In *RE*, pages 211–220. IEEE, 2007.

[43] Bill Schilit, Norman Adams, and Roy Want. Context-aware computing applications. In *In Proceedings of the Workshop on Mobile Computing Systems and Applications*, pages 85–90. IEEE Computer Society, 1994.

[44] Albrecht Schmidt. Implicit human computer interaction through context. Technical report, Personal Technologies, 2000.

[45] Albrecht Schmidt. *Ubiquitous Computing - Computing in Context*. PhD thesis, University of Lancaster, 2002.

[46] Albrecht Schmidt, Michael Beigl, and Hans w. Gellersen. There is more to context than location. *Computers and Graphics*, 23:893–901, 1998.

[47] D. Thevenin. Artstudio: Tool for multi-target ui design. Technical report, National Institute of Informatics, Hitotsubashi, 2002.

[48] David Thevenin and Joelle Coutaz. Adaptation and plasticity of user interfaces, 1999.

[49] B. Wetzstein, P. Leitner, F. Rosenberg, I. Brandic, S. Dustdar, and F. Leymann. Monitoring and Analyzing Influential Factors of Business Process Performance. In *Proceedings of the International Enterprise Distributed Object Computing Conference (EDOC)*, 2009.