Context-aware Web-service Monitoring

Ricardo Contreras

Professor Andrea Zisman (supervisor)
Professor Neil Maiden (co-supervisor)

A thesis submitted for the degree of
Doctor of Philosophy (PhD) in Computer Science
of the City University London

CITY UNIVERSITY LONDON
SCHOOL OF INFORMATICS
DEPARTMENT OF COMPUTING

February, 2013

Contents

List of Figures

List of Tables

Acknowledgements

First, I would like to extend my deepest gratitude to my supervisor, Professor Andrea Zisman, for all her support, constant encouragement, assistance and motivation during my research at City University. Her inspiration, specially in times of self-doubt and frustration is something that I will - without any doubt - be grateful for the rest of my life. Her insistence on details and clearness was always accompanied by a critic analysis - sometimes painful - but nevertheless necessary. I am deeply honoured to have worked with my supervisor during these four years for which I am sincere grateful. Finally, I believe it would have been extremely difficult, if not impossible, to finish this work without her continuing support and guidance.

Second, I would like to thank my co-supervisor Professor Neil Maiden and Professor George Spanoudakis for their support, feedback and advices during the whole my research. I would also like to thank the School of Informatics at City University, the S-Cube project, and all the involved partners.

I would also like to thank my colleagues and friends, specially Maria Crociani, Angela Kounkou, Adina Sirbu, Rafael Borges, Marco Comuzzi, Gilberto Cysneiros, Mark Firman, Davide Lorenzoli, Khaled Mahbub, Luca Pino, Rafael Roque, Icamaan da Silva, Igor Siveroni and Theocharis Tsigkritis. Finally, I would like to thank my family, my parents, my sister, and my niece and nephew.

Declaration on consultation and copying

The following statement is included in accordance with the Regulations governing the ‘Physical format, binding and retention of theses’ of the City University London

I grant powers of discretion to the University Librarian to allow this thesis to be copied in whole or in part without further reference to me. This permission covers only single copies made for study purposes, subject to normal conditions of acknowledgement.

Ricardo Contreras

Abstract

Abstract

Monitoring the correct behaviour of a service-based system is a necessity and a key challenge in Service Oriented Computing. Several efforts have been directed towards the development of approaches dealing with the monitoring activity of service-based systems. However, these approaches are in general not suitable when dealing with modifications in service-based systems. Furthermore, existing monitoring approaches do not take into consideration the context of the users and how this context may affect the monitor activity. Consequently, a holistic monitor approach, capable of dealing with the dynamic nature of service-based systems and of taking into consideration the user context, would be highly desirable.

In this thesis we present a monitor adaptation framework capable of dealing with changes in a service-based system and different types of users interacting with it. More specifically, the framework obtains a set of monitor rules, necessary to verify the correct behaviour of a service-based system, for a particular user. Moreover, the monitor rules verifying the behaviour of a service-based system relate to properties of the context types defined for a user.

The main contributions of our work include the general characterisation of a user interacting with a service-based system and the generation of suitable monitor rules.

The proposed framework can be applied to any service composition without the need of further modifications. Our work complements previous research carried on in the area of web service monitoring. More specifically, our work generates a set of suitable monitor rules - related to the user context - which are deployed in a run-time monitor component. Our framework has been tested and validated in several cases considering different scenarios.

Chapter 1

Introduction

Monitoring the behaviour of a software system is essential to ensure that the system executes as expected. However, monitoring is a complex process that involves verification of the system by checking specific properties describing the behaviour of the system. Additionally several elements including logical, physical, and even environmental factors can lead to unexpected behaviours or errors during the execution of the system if they are not visualized and dealt with in advance. As a consequence the monitor activity becomes quite challenging, especially when dealing with dynamic and evolving systems.

Many approaches and techniques have been proposed regarding monitoring of a software system, surveys focusing on different types of systems have been conducted in [20][106][135][136][161][164][229]. These monitoring approaches and techniques can be classified either as static analysis or dynamic analysis. Although it is argued that the two types of analysis can complement each other, there are differences between them. An approach is classified as static analysis when the behaviour of a system is evaluated without the execution of the system. Static analysis operates by building a model of the state of the system and then determining how the system reacts to this state [90][107]. On the other hand, an approach is classified as dynamic analysis when the behaviour of a system is evaluated during its execution. Dynamic analysis is performed by observing the outputs of the system during its execution [31][90]. The work in this thesis focuses in the category of dynamic analysis.

Our interest is centred on monitoring the behaviour of those software systems encompassed in Service-Oriented Computing (SOC). SOC is an emerging computing paradigm that utilizes *services* as basic building blocks to support the development of rapid, low-cost compositions of distributed applications. SOC relies on a Service Oriented Architecture (SOA) which is an architectural style based on loosely coupled interacting software components that provide services [89][214][227]. A service is a self-contained independent autonomous unit able to perform operation(s) on behalf of a user, an application, another service or a set of services. Services can be described, published, located, and combined with other services on a single machine or on a large number and variety of devices distributed over a network. Additionally, the information related to the implementation details of a service is usually unavailable and the user of the service has no control over it.

The composition of distributed applications using services and how they are coordinated in order to accomplish a certain task is called a Service-based Application (SBA) [9]. Likewise, a system that considers a Service-based Application, the partner services involved, and an underlying infrastructure for the application and the partner services, is called a Service-based System (SBS) [9]. We are concerned with the activity of monitoring Service-based Systems (SBSs), where information about the execution of a SBS is collected at runtime and used to verify whether the system is behaving correctly according to a specified set of properties. It is important to note that in our work we take into consideration the partner services involved, but do not address the underlying infrastructure, e.g. networking, for the application. Even considering this restriction, the evolving nature of a SBS, e.g. at some point a service participating in the composition might be replaced, makes it difficult to come up with a stable model of the system and its current state.

Monitoring a SBS involves additional challenges when compared with traditional systems. While in the latter the composition of the system is fixed, in SBS the different components, i.e. the services, are independent autonomous units that, during runtime, are combined to accomplish a certain task. Services can fail, become unavailable or even be replaced by other services, e.g. a faster and cheaper service becomes available. Furthermore, environmental contextual factors may lead the system to provide incorrect results.

Among the contextual factors that may influence the correct behaviour of a SBS, it is possible to identify those related to the user and the user interaction with the system. User context and user interaction have had very little attention when dealing with SBS monitoring. Furthermore, even when dealing with SBS designing and development, only a few approaches, e.g. [29][49], have addressed these issues.

1.1 Focus and Motivation

The focus of the work described in this thesis is *monitor adaptation* based on contextual factors related to users. More specifically, we define *monitor adaptation* as adaptation of the monitor rules (also known as monitor properties) used by a monitor component to verify the correct execution of a SBS. The adaptation of the monitor rules for a SBS involves: *i)* the identification of adequate monitor rules whenever they exist, *ii)* the modification of existing but not completely adequate monitor rules, *iii)* the creation of monitor rules when neither identification nor modification is possible, and *iv)* the removal of monitor rules that become obsolete.

Whether a monitor rule is adequate or not depends on different factors, including the composition and logic of the SBS being analysed, the correspondence between the elements defined in the monitor rules and the identifiable elements during the execution of the SBS, and the involved contextual factors.

Our work takes into consideration the previous factors in a dynamic scenario by modifying the SBS composition or by modifying the context factors from one execution to the next one. Furthermore, it considers monitor rules suitable to verify the correct behaviour of a SBS according to the user context characteristics, and capable of dealing with the user interaction in the system. More specifically, our work concentrates on context and HCI-aware monitor adaptation in which changes in the monitor rules are based on users interaction with a SBS and different types of user context.

In what follows, topics related to SBS, Monitoring, and User Context & User Interaction are further discussed.

1.2 Service-based Systems

A Service-based Systems (SBS) is a composition of different independent units, i.e. services or other service compositions, put together using a specialized language to perform a certain task.

The composition of a SBS is performed according to the approach by which the different elements are put together. There are two approaches by which a service composition can be accomplished, by *orchestration* and by *choreography*. A service composition thought as an orchestration, considers a central controller that defines how the participant services interact with each other and covers the business logic and execution, including conditions and exceptions, for a process [190]. In a service composition thought as a choreography, specifications are created determining the behaviour of each participant in the process. Unlike orchestration, the overall behaviour of the process emerges from the interaction of individual pieces [190]. In this work we centre our attention on the monitoring activity of those systems where the sequence of the process is specified by a central controller, i.e. orchestration.

There have been several proposals regarding the language used to describe service compositions including, among others, the Business Process Execution Language (BPEL) [120] and its extension the WS-BPEL Extension for People (BPEL4People) [140], the Web Services Choreography Description Language (WS-CDL) [133], and the Web Service Definition Language (WSDL) [65]. While BPEL and BPEL4People focus on the orchestration of a service composition, WS-CDL and WSDL focus on the choreography of a service composition. Nevertheless, although their focus is different, they can complement each other [168]. In this work we have focused on those SBS where the composition has been specified using the Business Process Execution Language (BPEL). The choice of BPEL is based on the following reasons.

- Its wide acceptance as the *de facto* orchestration language standard for executable process specification [185]
- Its expressiveness when describing the composition behaviour [134]
- The support of the language on different engines, e.g. Apache ODE [12], Oracle BPEL Process Manager [181], WebSphere Process Server [121]
- The support from different communities, including the academic one

Regardless of the task the SBS performs and its composition, it is always possible to identify the following three components, the *provider*, the *client* and the *registry* [186]. Services are made available by a third party, a provider who, from an architectural perspective, hosts and controls access to services. It is important to note that the provider might not have ownership over individual services or information regarding their implementation. The client is the entity that is looking for, and subsequently invoking, a service. The registry is a searchable directory where service descriptions can be published and searched.

In addition to the above, if the SBS is specified using BPEL, it is possible to distinguish five main parts: message flows, control flows, data flows, process orchestration and fault and exception handling [186]. Message flows are related to the invocation and response of an operation on web services. Control flows are related to the sequence of steps required to make up a given process. Data flows are concerned with the exchange of messages between partners. Process orchestration establishes consumer/provider relationships. Finally, fault and exception handling deals with errors that might occur when services are being invoked.

Probably one of the most distinguishing characteristics of a SBS is the the fact that the owner of the SBS does generally not own the component services. Similarly, this is often the case for the service provider regarding the offered services. Furthermore, the control in services execution is beyond the owner of the SBS or the service provider [9]. As a consequence, the introduction of monitoring mechanisms, that are able to verify the correct execution of service compositions are essential.

1.3 Monitoring

Monitoring is the activity concerned with the constant verification of a system. The monitoring activity is based on the use of a set established properties, i.e. rules. These rules are used to check whether the events occurring in the system behave as they are expected to. The monitoring activity usually involves a component[1], the *monitor*, in charge of verifying the collected data against a set of defined rules. The monitor is

[1] In some approaches, e.g. [40][46], the monitoring activity may involve several monitors

usually a separate component, without relations with the system being analysed. In most cases, it is limited to the run-time interception of exchanged messages; however in some approaches, e.g. [174][175], it can be strongly related with the adaptation of the system.

There have been several studies dealing with the activity of monitoring. In the area of SBS, existing monitoring approaches can be classified according to the system's behaviour, e.g. [25][192][220], the quality of services (QoS), e.g. [69][137] [153], or the contextual information of the services participating in the system and the system itself, e.g. [35][47][58]. The common characteristics in all these approaches is that the monitor rules, used for checking the correct execution of a SBS, are predefined and conceived for the particular system being monitored. Furthermore the studied approaches do not consider the fact that modifications in SBSs may cause monitor rules to be no longer suitable (e.g. a service participating in a SBS may become unavailable or malfunctioning with respect to certain QoS aspects, requiring the service to be replaced). In fact, when it comes to monitoring, the verification mechanisms are relegated to a second place once they fulfil their task, e.g. should a modification be performed, it would only concern the service composition (see section 2.4).

Users interacting with a SBS is another issue that has been, in most cases, neglected when performing the monitoring activity. User interaction, nevertheless, may also cause monitor rules to be no longer suitable, e.g. different user configurations may trigger the replacement of service components, making the previous associated rules no longer suitable.

Also in relation with SBS monitoring, it has been observed that monitor rules specification is usually a manual task, involving the participation of someone familiar with the logic of the SBS and the language used for representing the rules. The above implies that unless all possible monitor rules have been defined a priori, a very unlikely case, an expert with the knowledge of the system logic and language used for specifying monitor rules, will be needed.

In this thesis we tackle the problem dealing with monitor adaptation. More specifically, we address the adaptation focusing on the monitor rules and the user context information. As previously mentioned, contextual information could trigger changes in the rules without causing necessarily a change in the composition of the SBS. In order to apply the proper rules for verifying the correct execution of a system, it is essential to take into consideration the current composition of the SBS along with the user context.

The research we have performed over the last years shows that it is possible, at least up to a certain extent, to identify, modify, create, and even remove monitor rules based on the user context and the part of the system involved during execution. Furthermore, even in those cases were no modification is made in the SBS; the identification, creation, modification, and removal of monitor rules may still be conditioned and triggered by other factors, such as the user context. In what follows we introduce the concepts related to user context and user interaction.

1.4 User Context and User Interaction

User context is the information regarding characteristics of a user. User interaction, is concerned with the different ways a user communicates with a system. Both are strongly related, for example user context is a subject of study in the field of Human-Computer Interaction[2] (HCI), which focuses on the interaction between people and computers. In other words, user context involves user interaction.

[2] HCI is an interdisciplinary field including knowledge related to computer science, social science, and communication theory among others

While it is true that efforts have been made to include contextual information in the activities related to the design, deployment, and operation of SBSs, e.g. [49][48] [62], little or no importance has been given to the user context. Although in principle this may seems odd, specially considering the inclusion of human interaction in commercial products, see [140], it can be explained by the fact that human activities are treated as special types of basic activities within a process. The encapsulation of a human interaction as a special activity, reduces the complication when dealing with it. Unfortunately such encapsulation does not includes associated contextual concepts/models.

Regarding user context and monitoring, as far as we are aware, no studies have been conducted analysing the relation/impact that user context might have in the monitoring activity. While it is true that some studies have focused context and its impact on the adaptation of the process, e.g. [51][94][245], or on the use of context for the identification of the appropriate services for a composition, e.g. [132][152][239], no study tackles the existing gap regarding user context and the monitor adaptation.

1.5 Hypotheses and Objectives

In this section we present the hypotheses and objectives of the work developed over the last years and described in this thesis report. The general hypotheses of the work is.

> *It is possible to adapt monitor rules for Service-based Systems due to users interaction with the system, the different types of user context, and changes in the system itself in order to verify its behaviour.*

The above general hypothesis can be broken down into the following sub-hypothesis:

- It is possible to automatically identify monitor rules that should be used to monitor SBSs due to changes in different user context types and user interaction with the system.
- It is possible to create and modify monitor rules that should be used to monitor SBSs due to changes in different user context types, user interaction with the system, and service composition specification.
- It is possible to remove monitor rules that are not relevant to the SBS due to changes in different user context types, user interaction with the system, and in the service composition specification.
- It is possible to use the automatically identified, created, or modified monitor rule in a monitor component to verify the correct execution of the system and notify that there has been violation in the behaviour of a system that requires changes in the system.

Given the above hypothesis, the general aim of our work is to assist with monitoring activities of SBSs by automating the process of identifying, modifying, creating, and identifying obsolete monitor rules (so they can be removed), to be used during the verification of the system. This process will be executed due to changes in user context, interaction of the user with the system, or changes in the system itself.

The above general aim can be broken down into the following measurable objectives:

1. To provide a literature review and state of the art of the work performed in the area. More specifically, we will focus on user context, human computer interaction, SBS monitoring, monitoring adaptation, and their relations.
2. To identify the distinctive user context types likely to affect the execution of a SBS and to require adaptation of monitor rules.

3. To develop a model for the representation of the distinctive user context types.
4. To provide a classification for the various monitor rules with respect to the different user context types.
5. To provide a formalism to represent the various monitor rules for the different context types.
6. To develop techniques to support the identification, creation, modification, and removal of monitor rules.
7. To implement a prototype tool to support objectives 5 and 6 above.
8. To evaluate the results of the work in case studies and in a suitable monitor component.

1.6 General Overview

In this section we provide an overview of the methodology used to develop the work reported in this thesis, an account of the context in which the work was developed, and a description of the various phased taken to develop the project.

1.6.1 Methodological Approach

We explain the methodological approach in this work following to the classification given in [211].

Regarding the *research settings* we focus on *Feasibility*, *Characterisation*, and *Method*. Regarding *feasibility*, we examined the potential for success of a an approach dealing the monitor adaptation when involving user context. More specifically, *i)* we analyse the advantages (strengths) and the disadvantages (weaknesses) that underlie existing monitoring and adaptation approaches, *ii)* we identified user context types and analyse whether the existing approaches considered the impact these contexts during the monitor activity, and *iii)* we proposed a viable solution covering the existing gap in context-aware web service monitoring and identified the benefits of such proposal. Regarding the *characterisation*, we identified a set of characteristics which are important for the successful monitoring of service-based system when involving human interaction. This included: the type of service-based systems, the types of users, and the properties that could be specified and verified. Regarding the *research methods* we used, we started with the literature review, followed by the development of a descriptive model that aimed to cover the existing gap in the area of service-based systems monitoring and user context. More specifically, we studied existing approaches and proposals dealing with run-time verification, analysed existing service-based systems, and found out that no work has been carried out focusing on the run-time verification, system modification, and user characteristics. As a result, we proposed, developed and implemented procedures and techniques to carry out tasks in order to deal with the adaptation of the monitoring process.

In relation to the *validation techniques* our work is prototype-based: we have developed a prototype as proof-of-concepts and to support evaluating the work. The results show that our proposal is able to deal with the monitor adaptation of service-based system when considering human characteristics. In our evaluation we conceived and developed a set of experiments for validating our results. More specifically, we developed a prototype to demonstrate and evaluate the work developed in this thesis. The evaluation itself considered different criteria including a *descriptive model*, which was used when dealing with different configurations, and an *empirical model*, used to measure the performance and scalability under different loads and configurations. We performed an analysis based on an empirical model - simulation - in which the results were predicted in a controlled situation (statistical analysis). The evaluation was executed in two case studies in order to demonstrate the ability and efficiency. The first case study was about users interacting in a web organiser. The second case study considered users interacting in a air traffic control systems (chapter 6). Overall, the results show the ability and efficiency of our approach when addressing the problems regarding monitor adaptation and user in the loop. Furthermore, the automation of the whole process improves the performance when adapting the monitor - conceived most of the times a manual task - and reduces potential errors due to human intervention.

1.6.2 Context of the work

The work conducted in this thesis covers the area of Service Systems Engineering, with particular focus on service monitoring. The work carried out in this thesis was developed under the EU Network of Excellence S-Cube project [33]. As part of the project, our results complement the research carried out by the associated partners in related areas of Service Systems Engineering, including service discovery and service level agreement negotiation. Furthermore, as a result of the collaborations among the different partners, we proposed and published an initial approach integrating dynamic monitoring and service discovery features. This proposal represents the joint work of City University, London, UK and Fondazione Bruno Kessler, Trento, Italy.

1.6.3 Phases

Our work started - in an initial phase - by analysing the state of the art in the areas of web-service monitor adaptation and the human *in the loop* in service-based systems. After the corroboration of the existing gap in the area of context-aware web-service monitoring, we focused - in a second stage - in the identification of user context characteristics that could affect the monitoring activity. We created a model for these characteristics. In a third stage, we studied different existing approaches and techniques that could support our work. More specifically we studied proposals related to the generation of monitoring properties and the selection based of pre-conceived properties. We developed our own pattern strategy approach for the specification of monitor properties. In a fourth stage we implemented a prototype tool and conducted sets of experiments to evaluate our work. Finally in a final stage we analysed the results and provide an overview of our work.

As mentioned in section 1.2 there are two approaches by which a service composition can be accomplished *orchestration* and *choreography*. In this work we focused on *orchestration* because it is clearer in terms of control, i.e. while orchestration describes a process flow between services from the perspective of a centralized control, choreography tracks a sequence of messages from multiple parties (a decentralized control). Additionally, fault handling is easier in orchestration as the execution is controlled, this is not the case with choreography. Finally, services can be easily and transparently replaced in case of orchestration while it is more difficult in case of choreography.

1.7 Contributions

In this thesis we propose a holistic framework that combines monitoring, adaptation, and user context for supporting the continuous verification of a system's behaviour. The relation between these elements is depicted in Figure 1.1.

According to Figure 1.1 changes in user context and SBSs can trigger the monitor adaptation. Furthermore, the monitoring activity may trigger changes in the service composition (e.g. a service replacement) that, likewise, may trigger changes in the monitor. Our work takes into account these existing relations[3] and offers a solution

[3] Our work does not take into consideration those relations beyond the monitoring activity, such as

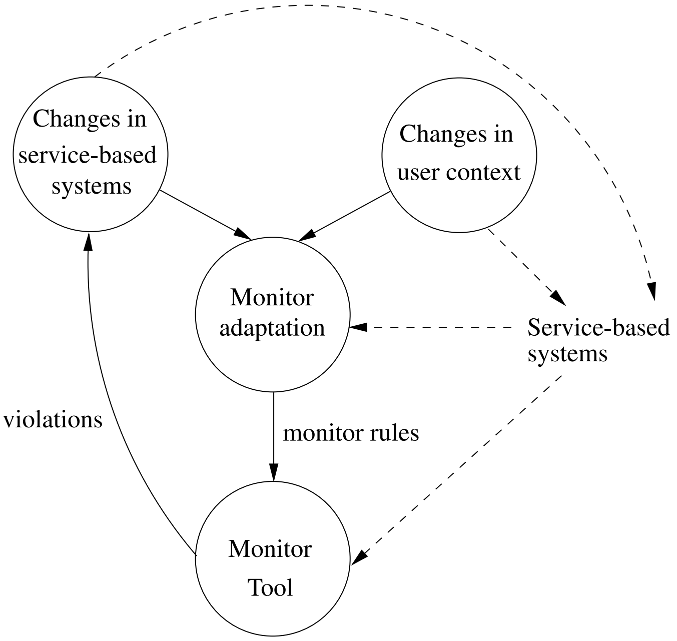


Figure 1.1: Relation between Context, Monitoring, Adaptation and SBSs

to the monitor adaptation problem. The contributions of our work include:

- Identification of user context types, general enough to be applicable to most SBSs.
- Proposal of a context model focused, mainly, on the user context types.
- Creation of an easily expandable pattern-based approach for representing the various types of monitor rules for the different user context types.
- Development of a framework based on pre-defined patterns for the automated identification, creation, modification, and removal of monitor rules. This framework is capable of dealing with different user context configurations and different SBSs.

The contributions of this work have been published in the papers listed below:

the one between context and SBS

- *A Pattern-based Approach for Monitor Adaptation*, 2010, IEEE International Conference on Software Science, Technology and Engineering [71].
- *Identifying, Modifying, Creating, and Removing Monitor Rules for Service-oriented Computing*, 2011, Proceedings of the 3rd International Workshop on Principles of Engineering Service-Oriented Systems [72].
- *A Framework for Dynamic Monitoring of Adaptable Service-based Systems*, 2012, Proceedings of the 4rd International Workshop on Principles of Engineering Service-Oriented Systems [70].
- *A Context-based Monitor Adaptation Framework*, 2013, Automated Software Engineering Journal (current under review).

1.8 Outline of this Thesis

The thesis is organized in 7 chapters as follows.

In Chapter 2 we analyse existing approaches dealing with monitoring and service-based systems, context, and human interaction. We also analyse the existing approaches dealing with context modelling, and adaptation in service-based systems and in the monitor.

In Chapter 3 we present our context model for service-based systems. Our model includes different dimensions when dealing with user context. More specifically, the model represents context by means of a set of context types, we present the rationale and benefits behind our model. In this chapter we also present a strategy for dealing with user interaction in SBSs, introduce our framework for obtaining monitor rules, and explain the main components of the framework.

In Chapter 4 we introduce the formalism used to specify monitor rules and present a pattern approach for the specification of monitor rules. More specifically, we start with a description of Event Calculus (EC), a language, based on first-order logic, capable of representing the behaviour of dynamic systems. Then we present our pattern strategy, based on EC, which serve as a template for the specification the monitor rules for the different user context types, previously described in Chapter 3.

In Chapter 5 we describe the adaptation process. More specially, we describe the strategy, based on the use of annotations, for the identification of the part of the system related to a context type. We also present an example covering the different activities (i.e. identification, creation, modification, and removal) when obtaining a set of monitor rules.

In Chapter 6 we present the results of the experiments conducted to evaluate the performance and correctness of the framework. We present the configuration as well as the scenarios used in the evaluation, and analyse and explain the results obtained from the framework.

Finally, in Chapter 7 we present the conclusions. We discuss the approach and expose the limits of the current approach along with the future work.

Chapter 2

Background - State of the Art

2.1 Overview & Methodology

The following sections present the state of the art, covering the topics previously introduced in Chapter 1. More specifically, in section 2.2 descriptions of existing works dealing with the monitoring activity are presented. These descriptions cover different issues and have been grouped according to three different perspectives: *Service-based Systems (SBSs)* perspective, *Context* perspective, and *Human Computer Interaction (HCI)* perspectives. In section 2.3 we focus on those approaches dealing with context modelling for SBSs. In section 2.4 we centre our attention on those approaches dealing with SBS adaptation. This is followed by section 2.5 which provides a description of monitoring adaptation approaches. Section 2.6 gives a summary of the above exposed approaches and highlights our concerns regarding HCI-context aware monitor adaptation.

In order to perform an objective, concise, and critical analysis, we performed the literature review of the issues above exposed. Initially, we focus on the main issue: user context-aware monitoring, and studied the existing work. More specifically, we identified the different areas and fields that could contribute to our research. We classified them by topic, e.g. monitoring, monitoring adaptation, and established whether our main concerns and hypotheses were - at least up to some extent - covered by the existing works. Regarding the methodology, we focused on recent work performed in the different areas. This included general papers, journals, conferences, and theses. We also studied existing tools (e.g. [24][175]), models (e.g. [42][105]), and projects (e.g. [25][33]) from the last five years. Initially, we covered each area separately, focusing on string matching for a particular topic to retrieve the relevant work. We expanded the search by considering the conferences focused on our areas of interests, as well as related work from other authors referenced by third parties. Most of our sources of information were obtained from digital libraries (e.g. springer, ACM), although we also considered material ranging from technical reports, to *de facto* standards.

2.2 Monitoring

As described in Section 1.3, monitoring is the activity concerned with the verification of the behaviour of a system according to a set of rules. The monitoring activity however, presents significant differences according to what is being monitored or how the system is being monitored. Furthermore, when performing the monitoring activity on a specific subject, e.g. when focusing on context monitoring, differences can also be appreciated from one approach with respect to another. On the other hand, when classifying monitoring approaches according to a certain taxonomy, as in [33], it is possible to observe some correspondences between the different categories. This exemplifies just how many different views can be, when performing the monitoring activity. Since our interest focuses on HCI, context, and SBS, in what follows we proceed to analyse each one of them in relation with the monitoring activity.

2.2.1 Service-based System

Monitoring of SBS has been the subject of research for the last several years; various approaches have been proposed to support this activity. Approaches for SBS monitoring can be discussed from different perspectives, including the specification language used to express monitoring rules or the mechanism applied to perform the monitoring process. In what follows we present and discuss some of the approaches.

In [93], the authors propose an approach to support monitoring of Service Level Agreements[1] (SLAs). The approach uses ecXML, a formalization based on XML and Event Calculus (E.C.) [143] to represent contract rules. A reasoner called Event Calculus State Tracking Architecture (ECSTA) is used to analyse the behaviour of the contract rules expressed in ecXML. In this approach, the monitor is a separate system that analyses contract rules, but does not interfere with the execution of the system being monitored. The approach is based on pre-defined contract rules for the services involved in an SLA and does not support changes and adaptation of the rules to be monitored. Moreover, monitoring of context aspects is restricted to those aspects defined in the SLAs of the involved services. The approach does not support monitoring of users interactions with the system.

The approach described in [199] attempts to minimize human intervention in the monitoring process. It uses a flexible formalization for SLA in which quality metrics (e.g., security, response time) are associated with formalizations describing services (WSDL) and process flow (WSFL, XLANG). The formalization of the SLA is made based on its constraint and clauses contained in the service level objectives (SLO). In this approach, agents are in charge of monitoring SLAs between web services exchanging measurements and protocols based on the formalizations. For each web service there is an agent managing the relationships of the service that interacts with

[1] A SLA is a contract between the provider of a service and a user of that service, specifying the level of service that will be provided.

other agents responsible for other sites services. The instance data required for the agents for monitoring is modelled in a high performance database that is updated for every transaction instance. The approach does not consider monitoring of user and context. In addition the rules used in this approach can not be changed.

The work in [37] is based on the use of an algebraic specification language to describe service data, service operations, and properties and semantics of the operations. The approach focuses on conversational services specified in BPEL and the monitoring of the functional behaviour of these services. In this work, monitoring is performed over a client-service interaction where run-time behaviour of the services is checked against their expected behaviour defined in the algebraic specification. The checking is based on symbolic execution of the algebraic specifications and uses term rewriting techniques. The component responsible for the checking (i.e., the Monitor) contains a symbolic state generator and an interpreter of the formal specification. Although the focus of the work has not been on context or HCI monitoring, we believe that the algebraic specifications of the services could be extended to support context representation.

In [23][26] an approach for dynamic monitoring of the BPEL process based on monitoring rules is presented. More specifically, the approach supports dynamic identification and execution of monitoring rules specified as WS-CoL (Web Service Constraint Language). These rules are weaved into BPEL processes during deployment time, allowing an explicit and external definition of the monitoring rules. This supports separation of business (BPEL process) and control (rules) logics. The original BPEL process is not modified and the approach uses a copy of this process where the monitoring rules are added. In order to define which rules should be considered at monitoring time for a service, a monitoring definition file is used. This file provides general information of the BPEL process to which the rules will be included, values associated with process execution, and the monitoring rules. The latter considers not only the expressions of the rules, but it also considers the locations and parameters of the rules. In this approach the focus is on monitoring of non-functional aspects (e.g. security, performance). However, the approach does not consider monitoring of context and HCI aspects.

The work in [29] proposes BP-Mon (Business Process Monitoring) a query language for monitoring business process that allows users to visually define monitoring tasks using a simple intuitive interface similar to those used for designing BPEL processes. In the approach tasks are monitored over defined activities representing the flow of processes. These activities are composed of atomic activities that are related to basic operations (e.g. messaging passing, value assignation). Activities are represented as a direct acyclic graph (DAG), where two nodes define an activity and edges represent dependency between the activities. The process instances to be monitored in the activities are specified by a graphical query language using execution patterns that extend regular expressions (In XML) to the DAG. The queries are compiled into the BPEL process specification, whose instances perform the monitoring task. The user interaction is considered in the selection of the compound activities (e.g. selection of a buyer or seller process). Context however has not been considered. Finally queries are defined for specific activities and are not able to be modified.

In [159][160][219], the authors present a framework for monitoring functional and non-functional requirements (run-time, SLA monitoring) of web services. The approaches are based on the use of Event Calculus (EC) [210] for the specification of requirements to be monitored. The requirements could include behavioural properties of the system that are automatically extracted from the specification of its composition process. Some basic transformations are given from BPEL activities to EC formulas. Requirements can also be expressed by system providers in order to monitor a specific behaviour. Special requirements can also be created, for example, to monitor SLAs. Requirements are checked against the events generated by the system being monitored. Evaluation is performed by the monitor without interfering with the system being monitored. The approaches do not consider the adaptation of the monitoring rules.

Similarly in [91] another monitoring approach, based on the use of predefined policies is presented. In this approach the requirements, to be monitored, are specified in a WS-Policy4MACS language. This language extends WS-Policy by defining new types of requirements and policies. The novelty in this approach is that, in addition to the specification of policies, the language proposed also allows for the specification of predefined adaptation actions, triggered by policies violations.

The work in [184] proposes the use of agents to perform the monitoring activity. In this approach, agents verify the correct execution of a system according to defined policies. More specifically, the extracted information from policies is used to identify, configure and instantiate management agents that will be used to monitor the system. One of the main challenges deals with the identification of relevant information and the instantiation of agents according to the scope and constraints of the monitored system.

The work described in [6][182][183] deals with web services monitoring, although its architecture has been designed to support - via independent interfaces - any kind of services. The focus is on SLA, more specifically, on the fulfilment of the service level objectives (SLOs) derived from the SLAs. The architecture includes an *analyser* component, which is responsible of checking whether the set of SLOs are fulfilled, and a *monitor services* component, which is responsible of retrieving the values of quality metrics of the different participating services. In order to retrieve the quality of service (QoS) from the participating services the framework allows for two strategies: *passive monitoring*, which collects data from the interaction between provider and client, and *active monitoring* where an engine invokes a service in a systematic manner. The novelty in this approach is the flexibility in the retrieval of QoS and the possibility of an easy integration with other systems. Furthermore, according to the authors the approach has been integrated with other strategies including the ranking of web services [53].

Numerous architectures/frameworks are found in the literature to realize runtime monitoring. These frameworks observe the current state of a running system and compare the observed state with the expected state of the system specified in a specification language. In [173] a model-driven methodology for a top-down development including monitoring capabilities is presented. It is based on a model driven orchestration design based on three layers, namely computer independent models (CIM), platform independent models (PIM), and platform specific models (PSC). In this approach, monitoring is based on quantitative process performance indicators (PPI) which are defined and evaluated on the basis of business events. The business process and the PPIs specifications are initially specified in two different models: a functional and a monitoring model, respectively. This separation is made to treat business process and PPI specifications separately. A business process definition meta-model (BPMN) is used for modelling both functional and monitoring models. In the first layer (CIM), the PPIs and the functional model are independently defined in BPMN. In the second layer (PIM), a PPI monitor model and an orchestration model are generated. In the third layer (PSC), a specific instrumented orchestration model is generated based on the orchestration and the PPI monitor model. The BPEL instrumentation, which includes the PPIs, is extended by monitoring sensors to a monitor infrastructure which is based on the generated monitor model in PIM. User interaction is considered if the orchestration involves human participation. Such interaction is treated as a special task inside an activity in the functional model. Context is not taken into account in this approach.

The work in [21] proposes an approach to monitor conformance between expected specified behaviour of a service and the actual behaviour of the service. The behaviour aspects of a service to be checked are concerned with time-outs, runtime and violation of functional contracts. These aspects are described as monitoring rules specified as comments in BPEL processes. The authors use C# and CLIX to represent monitoring rules (i.e., comments in the BPEL processes). In the approach, the content of an invoked service is serialized as an XML fragment and sent together with the associated rules to be monitored. The approach uses a monitor for rules specified in C# and another one for rules specified in CLIX. These monitors have been implemented as web services. The work does not consider adaptation of rules defined in the BPEL process for the three monitoring criteria. Moreover, the approach does not offer support for context and HCI monitoring.

The work in [11] is aimed to support fine grained identification of the causes of incorrect behaviour (i.e. exceptions) in Web Services. It is based on A.I. Model-Based Diagnosis (MBD) useful for reasoning on possible faults on software systems. Each composed service is associated to a model, called local diagnoser, and all of them are connected to a general global diagnoser. Local and global diagnosers are themselves web services. Each local diagnoser generates a local hypothesis (based on the MBD) which explains the exceptions of a service from a local point of view as they occur. This is done using special WSDL operations defined for diagnosis. The local diagnosers communicate to the global diagnoser that keeps track of each of the local hypotheses. Thus if a local diagnoser needs extra information to explain a failure that might be involved with another service it communicates through the global diagnoser. The hypothesis generated in the local diagnosers are not modifiable.

Classification

Approaches for SBSs monitoring can be discussed from different perspectives including the specification language used to express monitor rules, the aim of the monitor, or the degree of intrusiveness of the monitor approach.

In any monitoring process, monitor rules need somehow to be expressed. The characteristics of the rule specification language can affect the computational complexity of the monitoring process or constrain the expressiveness when formulating rules. There are several works in the literature focusing on the development of rule specification languages aiming to facilitate the SBSs monitoring process, e.g. [108][232][24].

Depending on the monitor approach and the specification language rules may be formulated, for example, aiming to monitor the Service Level Agreements (SLAs) or quality metrics, e.g. security, response time. Monitor approaches also include, most of the times, some kind of formalization for the representation of rules. Such formalization could be expressed in a logic-based language suitable to support behaviour representation of dynamic systems, e.g. [210], or in some type of algebraic specification language describing service data, operations and properties/semantics of the operations, as in [37].

Another important aspect of monitoring approaches deals with their level of intrusiveness. A monitor approach can be classified as *intrusive* if it interferes with the execution of the SBS, e.g. rules weaved into the process specification of a system during deployment time, or *non-intrusive* if it does not interfere in the execution of the SBS.

From the analysed monitoring approaches we have observed that,

- None of them really focused on the user interaction when monitoring a SBS.

- Context, and in particular user context, has not been considered as an influential factor when formulating monitor rules.

- Monitor rules were assumed to be pre-defined

- None of the approaches dealt with changes in the monitor activity, e.g. creation of new rules.

We believe that automatic retrieval of context information is another major concern when monitoring a SBS. This is based on the fact that manual identification of relevant context for a SBS would force the users to express all the relevant information for a given situation, which could end up being a difficult and tedious task for the users. Moreover users may not know which information is potentially relevant for a situation. None of the above approaches involving SBS monitoring, considers automatic identification of relevant context for a system.

2.2.2 Context

Context can be defined as “interrelated conditions in which something exists or occurs”. It characterizes the state of a certain entity by the identification of all factors surrounding the entity. In fact, several definitions and characterizations of context can been found in the literature and, from these definitions, different context models have been proposed.

Context models are classification structures, e.g. [75][79][171] [204][206][207], with different categories (also called dimensions) that altogether try to cover the characteristics of an entity, e.g. an individual, the environment, an object. Some context models are formulated as hierarchical models, e.g. [206][207], where main context types, i.e. the main classes, are expressed at the top level of the structure and further fine-grained specification is done at sub levels of the main context types. In fact, several context models in the literature can be viewed as a tree-shaped structure where the root node defines the main context types and each additional child node represents a sub context of a parent node, the ancestor of all nodes - the root - would be related to the main context types. There is no overlapping between context types.

It is also possible to find context model studies focusing on the operational definitions, e.g. [80], the evaluation of the context models and applications, e.g. [42], and the analysis of the different context types and models, e.g. [60], checking the overlapping between the different context types. In what follows, different context monitoring approaches are presented.

A context monitoring platform, Seemon, is presented in [131]. The platform reduces the expensive computation and communication costs in context monitoring for mobile devices. The reduction is achieved by translating high level context application queries into lower level queries in order to optimize the acquisition of relevant context. In other words in a context aware system, a single context type, e.g. location, could determine an outcome of a query avoiding the consideration of all the involved contexts in the query. Context information is obtained through physical sensors measuring features such as energy, temperature, speed, or users heart rate. The approach uses a query language, similar to SQL, and supports rich semantics for a wide range of contexts. A processor evaluates different queries over a continuous stream of sensor data. A sensor manager dynamically controls sensors avoiding unnecessary data transmissions. Its job is to find minimal sets of sensors needed to answer a query. 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 behaviour. However, the use of physical sensors is not adequate for monitoring of non-physical context aspects such as the knowledge of a user. Moreover, the complexity for the queries is directly proportional to the amount of sensors used.

In [64] a comparison between pull and push approaches is presented. The authors advocate that in pull approaches the user is aware and in control of the systems 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 behaviour of the system varies only with the users 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, in a tourism information system, once the location is explicitly set by the user in a theme park, an advance tourist guide system suggests attractions based on the attractions estimated adrenaline levels, the users 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, e.g. a user forgets to change his location.

A software platform, ContextPhone, is described in [193]. The focus is on mobile devices and its interaction with the environment. It is assumed 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 including location, user interaction, communication behaviour, and physical environment. The grouping into specific relevant types is actually a context sub-type classification that aims to characterize all the relevant context factors when developing a mobile 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 [35] an approach is described for monitoring distributed context to support internet services. It is based on an architecture [2] consisting on three key entities, namely a user, an operator, and a service provider. For each entity a set of context parameters is specified in a defined notation, generating what is called a profile manager for each of these entities. Rules can be defined, in order to report when a change in context information has occurred for a defined entity. When a context change occurs, it is reported to a special component 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. 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. Rules established for each monitor cannot be applied to different ones since they are based on a specific context. Hence the reuse of rules for similar situations, but different contexts is not possible.

A lightweight approach to context-sensitivity is presented in [111]. It considers the middleware limitations in dynamic mobile. 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 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. However, when using available monitors it is not possible to specify the sensing device from which the information is obtained. In this approach context monitoring is analysed from the developers 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 the focus in the approach was oriented towards a location context.

In [166] 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 labelling process. The main drawback is that the inclusion of a new sensor could trigger significant changes, modifying not only the labelling but the cluster as well. In the approach, monitoring is performed continuously in order to detect changes in sensors.

In [179] a study focused on wireless networks characterized by dynamicity, heterogeneity and mobility is presented. The authors propose the use of context awareness in order to manage a distributed network in a heterogeneous environment. This allows for a self-adaptation of the network, based on distributed analysis. The study considers a distributed architecture that allows efficient, scalable, and distributed management operations 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, 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.

An approach to represent and reason about physical contextual variability and its impact on the requirements of a system is presented in [202]. It considers the system may be required to adapt in order to cope with changes in the context and fulfilment of the system requirements. The work is based on the problem frame notation [126][201] 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. State machines are used to model the behaviour; a satisfied requirement is presented as a final state and an unsatisfied requirement is presented as an error state. The approach considers a scenario with separate physical domains where context variations occur.

However, the approach may not be suitable in scenarios where domains go beyond physical components, e.g. user context.

In [40] a proposal is presented for discovery and execution of web services by using contextual agents. In the proposal each agent monitors 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 pre-defined rules in a user profile. These rules deal with three main domains related to context: user role, user action type and information and information type. User role describes the nature of the users task. User action describes a specific action. Information type delivers information of the local user. An agent searches for services and information relevant to the users, based on the user profile. Rules are either defined by the user or by the agents based on rules for other users. Profiles are updated based on users preference when a particular service capability is not defined. While the approach is adequate for discovery and execution of web services, it is not well suited for monitoring; a considerable number of irrelevant preferences can arise. The approach deals with context definition and acquisition in a novel manner. More specifically the contextual agent collects information from the users 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, colour, and execute a query to find a web service that provides relevant information of product X.

Classification

In what follows we provide a classification of context monitoring approaches. 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 described in [115]. More specifically, categories *context acquisition* and *architecture* comply with the *How* dimension presented in the monitoring taxonomy in [115]. The *context model* category comply with the *What* dimension presented in the monitoring taxonomy in [115]. The *adaptation* category complies with the *What* and the *How* dimensions presented in [115].

i) *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 two ways and these are,

   a) Physical sensors, which consist of hardware devices that collect context information directly from the environment of the monitored system (e.g. temperature, brightness) [131][166][201].

   b) Logical sensors, which are implemented as software modules that may compute context information based on the context information collected by the physical sensors (e.g. average temperature), collect context information by polling system parameters (e.g. battery level of a PDA), or collect context information from user profile [35][37][40][179]. However, it is also possible for some context monitoring approaches to apply both methods (i.e. physical and logical sensors) to collect context information [111][166][193].

ii) *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 modelled in many ways including,

   a) Simple attribute/value pairs, with predefined semantics of the attributes and possible set of values. This approach allows expressing simple logical conditions on the attributes [37][193].

b) Structured language that based on some formalism, e.g. predicate calculus based language can support application of Boolean algebra [17].

iii) 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,

a) 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 (e.g. for video streaming a fall in the available bandwidth sets video quality rendering to a lower quality) [37].

b) Adaptation of the monitor where the monitor can adapt itself with respect to a change in the context (e.g. a monitor may start monitoring a new set of rules when a room temperature rises above a certain threshold) [131].

iv) 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,

a) 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 [37][111][131].

b) Non distributed architecture where context information is gathered by the application directly from the environment through sensors or input devices [166][202].

In all of the analysed approaches the categories dealing with Context Models, Adaptation and Architecture allowed a single classification (e.g. either simple attribute/value or structured language for Context Model) or were non-applicable for the approach (e.g. adaptation undefined for the system and for the monitor in Adaptation). For the category of Context Acquisition, on the contrary, around half of the approaches considered both classifications (i.e. logical and physical sensors) to collect context information.

2.2.3 Human Computer Interaction

HCI monitoring is used to design systems in such a way that a 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 what follows we present a survey on the Human Computer Interaction (HCI) monitoring.

A novel method for detecting interaction styles in a human-robot relation using time series analysis is presented in [98]. 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. Interaction styles are defined as behaviours between the user and the machine, e.g. rudeness, frequency, and the algorithm is able to recognize and classify such behaviours 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. The authors claim the algorithm could enable real time adaptation of machines to interaction styles. Although not explicitly mentioned, monitoring is constantly performed, i.e. every time a set of grouped signals is compared to another one. The main drawback with this approach is its dependency of a 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.

In [99] *Classroom*, an automated lecture facility system, is presented. The 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 and is able to automatically focus on the point of interest during a presentation, e.g. if the lecturer starts writing on the board, Classroom automatically focuses its presentation camera on the words and figures been written. Classroom represents the activities taking place as a process that contains a sequence of actions and 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. The approach is highly dependable on the physical interaction of the user and does not consider user contexts.

In [157] an approach that monitors user actions in a system providing appropriate suggestions to the user is presented. The approach infers, from the user interactions, the interests from the users. The feedback provided to the user considers only the information considered as of interest to the user and exclude non interesting information. In this approach a user is modelled as a list of keywords. Keywords are derived from user interactions (e.g. keyboard input, user email, web pages read) and then a list is produced by determining the relative frequency of the original keywords. The system also gathers information from the outside world (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 [151] a user interface agent, *Letizia*, is presented. It assists users browsing the World Wide Web according to the users interest. The above is accomplished by tracking users browsing behaviour. The approach applies a set of heuristics to model the users browsing behaviour. The approach relies on strategies such as assigning a degree of interest according to the amount of time a user spends for a link or the continuous visits from a user for a specific link. By analysing the user interest, the approach complies a set of recommendations that the user can either to follow or reject. The main drawback in this approach is that no external factors are considered when the user interacts with the system. Furthermore the approach assumes a constant level, e.g. attention, when interacting with the system that remains unchanged during the whole interaction.

In [52] a system that monitors information requested and accessed by users in browser applications has been proposed. Monitoring in this case is used to analyse users interaction with the browser and, based on it, pro-actively 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 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. The approach in [104] proposes a four layer architecture to generate feedbacks to the user. In the first layer a sensor service monitors the learners interaction with the system. In the second layer an objective is created based on the learners actions. In the third layer the objective is contextualized regarding the learning situation and learning process. Finally in the fourth layer, the generated feedback is reported to the learner.

Various applications dealing with implicit HCI are discussed in [205]. 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, e.g. a system may automatically switch on the light when a user walks in a dark corridor. A user walking in the corridor may not intend to interact with the system, nevertheless the system considers this user action as an input. These type of systems rely 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 capture. The paper argues 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. However since interactions do not directly depend on the user, an undesired behaviour can be easily triggered.

In [17] user interactions are analysed and compared with expected models of user activities. 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 users 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 higher level application events, e.g. selection of a menu item. A task monitor receives events from event handlers and monitors the users progress by matching the events to the user task model. The task monitor notifies the user level services of user task related events e.g. starting or finishing a task. In the approach a correct execution would involve the correct execution of all tasks related to the specified goals. The approach does not consider any user factors with exception to those related to the interaction.

In [78] a study is presented to generate, at run-time, homogeneous and coherent user interfaces with independence of the device used by the user for the interaction. 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 dynamic merging/separation of UI components. An interface adaptation is possible according to the business constraints. An interface service prepares the components for interaction and an interaction server. An 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. The approach deals with adaptation of the interface, however it does not consider user characteristics in the process.

A framework for multi-target user interface (UI) design is proposed in [54][228] [226]. It includes a tool which serves as an instrument to help designers and developers structuring 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 context, e.g. visualization from mobile devices, visualization from a PC. In order to accomplish the above 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 models describe 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 the previous descriptions are referenced. The development process considers the combination of different models for a particular target and the creation of bridges between the descriptions for different targets. User context in this case is limited to the specified in the models, however these models seem to focus on the interface, rather than the user.

Studies have been performed in the are of physiological monitoring. More specifically in [45] a study for face expression recognition is presented. It aims to establish the relation between user affective states, i.e. positive and negative states, and a defined task to be performed. Tasks are 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 users facial recognition is part of physiological monitoring in HCI. The approach is innovative but limited to the specific context involving face expression recognition.

Classification

In this section we provide a classification of the HCI monitoring approaches. We consider almost the same categories we used in section 2.2.2 to discuss context monitoring approaches, except for *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 users interaction with the system against an expected model of the user activities. We discuss the works according to *i)* Event Acquisition, *ii)* User Model, *iii)* Adaptation, and *iv)* Architecture.

i) *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,

   a) By instrumentation of the application or applying an adapter to the application the user is interacting with [17].

   b) Physical sensors can be used to collect user activities (e.g. movement of a user in a room, eye gaze of user) [45][98][99][104][157][205].

c) Events can be captured from the interactions a user makes through the input devices of the system (e.g. keyboard typing, or mouse click) [52][151]. However some approaches apply both the physical sensors and input devices to collect runtime events [157].

ii) *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 is mainly specified in two ways and these are,

a) 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 analysing the relative frequencies of the keywords [151][157].

b) By using structured description language that specifies formal syntax and semantics to express user activities [17][99][205].

iii) *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,

a) The system can adapt itself to assist the user in achieving the goal. For example in an automated distant learning configuration 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 [54][99][78][205][226].

b) 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 web page [52][104][151][157].

iv) *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,

   a) Client server architecture where the client provides a front end to the user and collects the user interaction and the server acts in the back end as the reasoning engine that receives the events and compares the events with available user models [17][104].

   b) 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 [52][157].

Again, in all of the analysed approaches the categories dealing with User Model, Adaptation and Architecture allowed a single classification. For the category of Event Acquisition, the use of more than one mechanism to collect runtime events for the monitoring process was considerable reduced when compared to context monitoring, in fact few of the approaches considered different mechanisms to collect runtime events.

2.2.4 Correspondences

From the previous classifications it can be observed that those approaches dealing with HCI and context are, most of the time, strongly related and that their differentiation can be, some times, quite hectic. This problem becomes clearer when trying to analyse approaches dealing with HCI and the monitoring activity, and context and the monitoring activity separately. Furthermore, approaches dealing with SBSs monitoring are not exempt from this relation. In fact, as long as a SBS envisages human interaction, e.g. [37][173], correspondences can be found among the system, HCI and context.

As a consequence of the above, the approaches previously described in sections 2.2.1, 2.2.2, and 2.2.3 can be further categorised according to different criteria. In what follows we provide a general classification of the previous analysed approaches based on their similitude. More specifically, the classification considers approaches dealing with *i)* HCI and context acquisition and processing and *ii)* HCI and context taxonomy.

The group of approaches dealing with HCI and context acquisition and processing, e.g. [35][45][98][131][173][193], is characterized by the use of widgets and/or user pre-established configurations or preferences (e.g. threshold levels). The contextual information in these scenarios is obtained from the surroundings using dedicated sensors, e.g. location sensors, and processing it according to a default configuration, or a specific configuration defined by the user. These approaches are characterized by their focus on a particular environment, with specific context types and associated pre defined conditions; making them, most of the time, unsuitable even for slightly different scenarios.

The second group, dealing with HCI and context taxonomy, aims to classify context types from a high level perspective by proposing general context models (see 2.3) and taxonomies, e.g. [37][49][42][206][201][207]. In these approaches the goal is to categorize, if not totally, partially, the surrounding context of interest for a system. A common characteristic in these studies is the initial and generic context classification. This classification consists of a reduced number of context types (usually no more than four or five), expressed in natural language. On one hand, generality of a context type definition allows for a wide range of factors (or other sub-context types) to be part of the definition. On the other hand generality carries the problem it may get to be too abstract, or inadequate, for specific scenarios.

2.3 Context Modelling

Perhaps one of the most recent and widely accepted context definitions is the one given in [116], where context is any *information that is computationally accessible and upon which behavioural variations depend*. However, besides this, several other definitions and characterizations have been given for the term and, along with these definitions and characterizations, different context models have been proposed. These context models aim to provide a better understanding of the properties, aspects, categories, or dimensions related to context. It has been noticed that most of the approaches dealing with context models or context modelling are general in order to cover different scenarios. The approaches described in this section address context modelling in SBSs. They propose new context models to cover issues ranging from requirements elicitation to the development of context aware applications.

In [49] a context model has been proposed for the formalization of the most relevant aspects characterizing a Service-based System (SBS). It consists of an XML representation of the main context components for SBSs featuring six main dimensions that are used to describe the status of an application. These dimensions include: *i)* time, referring to the information about the time in which the system is accessed; *ii)* ambient, related to space factors (e.g. address) or environmental conditions of the user; *iii)* user, concerned with the privileges, roles, and preferences of users; *iv)* service, related to information about the services in the system; *v)* business, which takes into account business application factors; and *vi)* computing, which specifies the available software and/or hardware characteristics.

An ontology-based context framework is presented in [18] and [19]. The work in [18] aims to facilitate the development and deployment of context-based web service applications. The work in [19] aims to dynamically integrate context model-based constraints into web service processes. The context model is composed of four context types, namely Functional, QoS, Domain, and Platform context. Functional context describes the operational features of services in terms of *i)* syntax: input and output parameters; *ii)* effect: pre and post-conditions; and *iii)* protocol: rules and data flow. QoS context deals with the needs, explicitly declared by the user and requirements not known to users. It includes *i)* runtime attributes: measurable properties; *ii)* financial/business attributes: assessment of a service from a financial perspective; *iii)* security attributes: whether the service is compliant with security requirements; and *iv)* trust attributes: relationship between clients and providers. The Domain describes each application in its own context including *i)* semantics: concepts and properties; *ii)* linguistics: the language used; and *iii)* measures and standards. The Platform context category describes the technical environment. It involves *i)* devices: computer/hardware platform, and *ii)* connectivity: network infrastructure.

An integrated context model for business process management is presented in [242]. The approach aims for an integrated view of context data belonging to context-aware services, workflows, human tasks and their interrelations. The model consists of three parts: workflow model, service model, and task model. Each part is represented in terms of classes with sub-classes in some cases. For each class there are attributes describing states and context. The workflow model is concerned with the control flow of the application, including state of execution, processed data, and generated tasks. The service model is concerned with services, including supported operations, access protocol, and performed operations. The task model is concerned with human tasks, including action, duration, origin, and destinations.

A framework for context information provisioning is proposed in [16]. The framework relies on context service deployment on the cloud and use of context brokers to mediate between context consumers and context services using a publish/subscribe model. A multi-attribute decision algorithm is used for the selection of potential context services that can fulfil contexts consumers requests for context information. The selection is made based on the QoS and quality of context information (QoC) requirements expressed by the context consumer. Context information is processed according to who, where, when, what, and why a service was invoked. This is enough to respond to different situational circumstances, e.g. the identity of client who invoked a service, the activity that the client is carrying out at the time it invokes a service, and the device a client is using to invoke the service.

A service-based approach to develop context-aware automotive telematics systems is presented in [230]. Telematic systems are automotive technologies combining advanced communications and vehicle technologies. The approach utilizes services as a means to acquire context and assist telematics in adapting contexts. A four-layer-architecture enables the separation of development of telematics and management of context adaptation. The first layer supports the operation and integration between hardware and sensors. The second layer wraps functions (from the first layer) as physical context services that expose standard WSDL. This layer also manages discovery and binding of external services provided by a third party, e.g. location services. The third layer includes social context models that represent relationships and interaction constraints between entities, as well as how interactions are affected by physical context facts. Each social context is modelled using a role-oriented adaptive design composition consisting of functional roles, interactions constraints, and organizer role. A social context model is implemented as a service that exposes WSDLs interfaces corresponding to its functional role. The fourth layer includes context-aware telematics that use context services (social and physical) to “sense” context and adapt themselves in response to changes of context.

In [213] a context-based model for access control in mobile web services is proposed. The work combines semantic web technologies with context-based access control mechanisms. An ontology is used for modelling and reasoning about context, and specifying access control policies. The context model consists of four different context types covering access control: subject contexts, object contexts, transaction contexts, and environment contexts. In the case of subject contexts, a subject is an entity that takes action on an object or resource. Subject contexts define the specific subject-related contexts that must be held by a subject to obtain rights to use an object or resource. In the case of object contexts, an object is an entity that is influenced by a subject. Object contexts are any object-related information that can be used for characterizing the situation in which an object was created and information about its current status. In the case of transaction contexts, a transaction involves the user, the mobile platform, the specific resource or service, and the physical environment. Transaction contexts specify particular actions to be performed in the system. In the case of environment contexts, an environment describes the operational, technical, and situational environment at the time a transaction takes place.

In [55] the work proposes a framework, a computational context modelling framework (CCMF), to extend software in order to integrate context. The framework relies on the reuse of artefacts and tools to automate analysis and development activities related to the making of context-aware web applications. The instantiation of the framework considers two different cases *i)* integration of a computational context modelling diagram (CCMD), which allows the creation of diagrams to model concepts related to computational context (it includes six different context dimensions) and *ii)* the embedding of ontologies to support the representation of context structures and the generation of context aware mechanisms. Both, *i)* and *ii)*, intend to enable the development of context-aware web applications. According to the authors, use of an ontological approach leads to a reduced framework when compared to the CCMD approach. Whether adopting CCMD or ontologies as development framework depends on what is the intended context information source.

In [122] a formal definition of services with context-dependent contracts is presented. The approach proposes a composition theory of services with context-dependent contracts taking into consideration: functional, non-functional, legal and contextual information. A formal verification approach transforms the formal specification of service composition into extended timed automata that can be verified using a model checking tool. The key concept in the approach is a package, called *ConfiguredService*, in which service functionality, service contract, and service provision context are bundled together. The *ConfiguredService* consists of two essential elements contract and context. The context part of the *ConfiguredService* includes *i)* contextual information and *ii)* contextual rules. Contextual information is specified using three dimensions: where, when, and who, associated to location, temporal information, and subject identities respectively. Contextual rules define information related to the service requester; contextual rules are defined as constrains in a subset of time computation tree logic (TCTL).

2.4 Service-based System Adaptation

In [51] a framework that allows to model dynamic, adaptable and context aware service-based applications is presented. In this work, processes are modelled as Adaptable Pervasive Flows (APFs) [50], which are an extension of the traditional workflow language (i.e. WS-BPEL). The use of APFs makes processes more suited for adaptation and execution in dynamic environments. In addition to the classical workflow language constructs, APFs add the possibility to model abstract activities. An abstract activity is defined at design time in terms of the goal it needs to achieve and is dynamically refined at run-time into an executable process, considering the set of available services, the current execution context and its goal. The context model is described as a set of context properties, each modelling a particular aspect of the application domain. Every context property is modelled with a context property diagram, which is a state transition system capturing all possible property values and value changes. A specialized engine manages the execution and adaptation of the processes. The approach supports two different dynamic adaptation mechanisms: vertical process adaptation by automatic service composition, and horizontal process adaptation by context-aware re-planning. The framework supports two different dynamic adaptation mechanisms: vertical process adaptation [216], i.e. by automatic service composition, and horizontal process adaptation [51], i.e. by context-aware re-planning. Probably the main drawback in this approach is that all participating services and the defined goals need to be manually annotated, so they can be used in the state transition systems.

In [59] a plug-in architecture for self-adaptive web service composition is presented. In this approach adaptation features for SBSs are modularized as aspect based plug-ins (the approach makes use of the aspect-oriented programming paradigm). An aspect-aware orchestration engine takes care of managing all active plug-ins and their dependencies. The orchestration engine can be extended by self-adaptation plug-ins using extension points. Each plug-in follows a well-defined objective and consists of several aspects and infrastructural services. A plug-in is developed by domain experts, e.g. an administrator and can be deployed to the orchestration engine at run-time. Finally, inside plug-ins two types of aspects are used *i)* monitoring aspects, which collect information and decide based on it whether adaptation is needed, and *ii)* adaptation aspects, which handle the erroneous situations and events detected by the monitoring aspects.

The work in [49] considers context, and its evolution, as the triggering element in the adaptation of SBSs and proposes an approach to design and develop adaptable SBSs. The approach is based on the life-cycle proposed in [9] that highlights the typical design iteration cycle along with a second iteration cycle, at run-time, that is undertaken whenever adaptation is necessary. The approach includes a context model that includes the most relevant aspects characterizing a SBS and consists of six general dimensions including: time, ambient, user, service, business, and computational context. In this model each dimension can be further refined. The context-aware design process consists of three phases: *i)* context modelling, *ii)* modelling adaptation triggering and requirements, and *iii)* construction of contextual monitors and adaptation mechanisms. In each phase the contextual dimensions are exploited in order to cover the possible influential factors in the system behaviour that could lead to the need for adaptation.

In [142] the authors propose an adaptation approach focusing on business interfaces and protocols adapters, this proposal is based on their observations where many of the differences between business interfaces and protocols are recurring. The approach relies on the use of patterns for capturing the recurring differences and providing solutions to these differences. The approach leverage mismatch patterns for service adaptation with two approaches, *i)* by developing stand-alone adapters, which consists on developing a third service that mediates the interactions between two incompatible services and *ii)* via service modification using an aspect-oriented approach. Guidelines are provided to help developers to decide on situations in which strategy, *i* or *ii*, is preferable. Mismatch patterns provide a simple and effective abstraction for capturing and resolving differences. They include, information regarding the type of difference captured, the needed information when instantiating an adapter and sample usage among others.

The work in [66] proposes a framework that allows users to specify the QoS parameters they require and undertakes the task of location and invoking suitable services. The framework complements the BPEL execution with features that involve *i)* specifying an execution policy, which comprises of restrictions for QoS attributes and defining ranking criteria in terms of QoS characteristics *ii)* the dynamic choice of the best available service according to a given policy, and *iii)* automatic exception handling in the presence of system faults. The approach does not use pre-determined paths, instead it selects services dynamically from a registry. In order to perform the latter, the framework includes two additional modules: an alternate service operation binding (ASOB) and a preprocessor. The ASOB *a)* selects the best matching operations operations according to specified QoS, *b)* transforms messages and results to tackle syntactical differences between services, and *c)* intercepts exceptions and resolve them by invoking equivalent operations. The preprocessor transforms the BPEL scenarios allowing direct invocations, it also allows to include, in each invocation, all the necessary information for selecting the best available operation.

In [245] a cross-layer adaptation manager (CLAM) is proposed. It tackles the problem associated to the isolation of different layers when performing adaptation. In other words, most of the existing approaches focus on a particular layer, e.g. application, excluding the impact this adaptation may have in a different layer, e.g. infrastructure. The approach proposes a platform that integrates and coordinates different adaptation approaches, focused on different aspects of a SBS. The adaptation manager is based on a comprehensive high-level model of the application and of the layers behind it. The approach also includes for each model element[2] *i)* a set of analysers, verifying whether there is a need for adaptation, *ii)* a set of solvers for identifying possible solutions, and *iii)* a set of enactors to apply the solutions on an element. The approach consists of five main parts: *i)* a rule engine, where rules are implemented for the overall supervision, *ii)* checkers, related to the capabilities plugged in the platform, *iii)* model updater, in charge of the system configuration, *iv)* tree constructor, where results of the cross-layer adaptation are continuously updated, and *v)* strategy ranker, where the tree is traversed to output validated and ranked adaptation strategies.

Another multi-layer adaptation approach is proposed in [13] and [14]. The ap-

[2] These elements depend on the layer, for example for the application layer an elements can be a *process activity*

proach proposes a proactive adaptation of service-based compositions by *i)* the use of techniques predicting QoS aspects, *ii)* the analysis of dependencies between the different services, and *iii)* the consideration of groups of operations instead of single operation for a replacement. Proactive adaptations occur when *i)* problems arise, causing the composition to stops its execution, *ii)* an improvement in the composition is possible, *iii)* there is emergence of a new requirement *iv)* a better service becomes available.

In [77] a theoretical framework is proposed to cope with unplanned exceptions in a service composition. The work includes the implementation of a process management system (PMS) which features a set of techniques to deal with unplanned exceptions. In the work, the attention is centred on highly pervasive scenarios. In order to provide an automatic adaptation in these scenarios the approach relies on the use of situation calculus [195], and automatic planning. Also, an interpreter, IndiGolog [103], is used to support on-line planning and plan execution in dynamic and incompletely known environments. Adaptation is synthesized automatically without relying on the intervention of domain experts or the existence of specific handlers to cope with exceptions. Monitoring the behaviour, in order to establish when to perform adaptation, is based on previous works which consider situation calculus agents.

In [246] a dynamic process reconfiguration is presented. The approach consists in the replacements of failed services by new ones, taking also in consideration the process still meets the QoS constraints specified by the user. The approach relies in the use of an iterative structural inspection algorithm designed to perform reconfiguration when a failure occurs. As a consequence, when one or more services fail at run-time the approach tries to replace only the malfunctioning services, however if a replacement for some failed service cannot be found, adjacent services may gradually be taken into consideration until a satisfactory solution is found. Services candidates are identified for each participating service based on syntactic and semantic matching.

In [10] and [22] approaches toward the self-healing for service composition are proposed. The work in [22] proposed an adaptation based on monitor rules and established reaction strategies. Similarly, in [10] monitor and recovery actions are used for a system adaptation. The main issue in this approaches deals with the fact monitor rules and recovery actions must be predefined.

In [15] a framework is described to perform run-time system adaptation. More specifically, the framework allows for the development and the deployment of adaptable applications, which consume and provide services, targeted to mobile resource-constrained devices in an heterogeneous network. This is accomplished by using the *chameleon programming model* that, extending the JAVA language, permits developers to implement services in terms of *generic code*. This code, opportunely preprocessed, generates a set of different Java components that represent different ways of implementing a provider/consumer application. The framework focuses on the fulfilment of non-functional requirements and specific context of use.

The work in [88] deals with models associated to non-functional properties and adaptation triggered by rules associated to the models. It proposes the creation of models their constant update at run-time. The idea is to provide a better representation of systems in dynamic environments where value of parameters change over time. The analysis of an updated model at run-time allows for the detection or prediction of a property violation. This may trigger automatic reconfiguration or recovery actions aiming at guaranteeing established rules are not violated. The approach relies on the use of a Bayesian estimator for the collection of data at run-time.

In [169] the authors propose the proactive adaptation of a system triggered by prediction of failures in the process. The approach relies on a proactive adaptation process, i.e. an adaptation occurs before a violation actually happens, based on a future prediction. The approach involves augmenting the service monitoring process with online testing, to produce failure prediction with confidence. The work covers an area that has not been fully addressed, which deals with the problem of whether to adapt a system based on a predicted future failure.

Classification

Approaches related to the adaptation of SBS can be classified according to different categories including, among others, *i)* the type of strategy, i.e. proactive or reactive, *ii)* whether the approach aims for a single layer or multiple layers, i.e. application, service and infrastructure layers (see [163]), in the adaptation process or *iii)* components involved in the adaptation.

i) Strategy

   a) *Reactive Adaptation*: in this case the adaptation process is triggered as a response to a known, past event.

   b) *Proactive Adaptation*: in this case the adaptation process is usually triggered by inference. This inference can be the result of some logical evaluation of observed events or based on statistical analysis.

ii) Layers tackled in the adaptation

   a) *Single layer Adaptation*: the adaptation focus on a specific layer, e.g. application layer.

   b) *Multi-layer Adaptation*: the adaptation focus on the different layers and evaluates how a change in one layer can affect the another.

iii) Components

a) *Variable*: in this case components related to the adaptation may correspond to modularized plug-ins, which may change even at runtime.

b) *Static*: in this case the components related to the adaptation correspond to pre-defined self-adaptive features.

2.5 Monitor Adaptation

Most of the existing approaches addressing adaptation of SBSs focus, as expected, on the adaptation of the application using service re-composition mechanisms. Typically, adaptation approaches make use of *monitor components* to support the identification of problems in the service-based system triggering the need for adaptation. These *monitor components* are responsible of verifying the behaviour of a service-based system with respect to some pre-defined properties and requirements. An important problem is concerned with the support for the adaptation of the *monitor component* itself. For example, the support for changes in the monitor rules due to changes in the system, what exactly needs to be monitored at a certain time, or even changes in the monitor component are issues normally excluded from the adaptation process. In what follows, we describe some approaches concerned with monitor adaptation.

A run-time monitor architecture for web services is presented in [46]. The work aims to provide a holistic monitoring framework by enabling the integration of different verification tools. The architecture is capable of integrating different monitoring approaches and it was designed with the intention of being pluggable to support multiple concurrent monitors for different monitoring aspects. The work described in [46] concentrates on the behaviour verification of services with respect to their advertised specification during run-time. The approach is based on stream x-machines (SXM) to represent the behaviour of web services. SXMs are special instances of x-machines capable of representing both data and control of a system. An integrated tool in the monitoring architecture is used to process the requests/responses to the SXM. The outputs of both SXM models and web services are compared. If there is a match with the outputs, the service behaved as expected; otherwise a deviation occurred. In this work, monitor adaptation is concerned with SMX models since a new web service in the service composition implies the creation of a new SMX. However, it is not clear whether the creation of a SXM is a fully automated process in the approach.

In [198] the authors present a component-based framework for monitoring and managing features of composite SOA applications. Their proposal relies on the use of components responsible for each activity, namely monitoring component, SLA analysis component, decision taken component, and execution of actions component. In the framework, the different components are attached to each service being managed in order to provide the required information. For example, the monitor component collects, stores, and filters information. The framework works by monitoring data from each individual service and calculating a set of metrics for them. The list of available metrics is exposed by the monitoring component and used by the SLA analysis component which can read the metrics and check if the specified conditions are being fulfilled or not. In the case in which a condition is not fulfilled, or has some risks of not being fulfilled, the decision component is activated and decides on the actions to be taken. When a set of actions are identified, they are passed to the executor component to be executed over the managed service. The work supports the addition and removal of different components at runtime. For example, a service to which no monitoring information is required may not need the monitoring component and may only have an execution component to modify some parameter of the service. The monitor component can also be adapted with respect to the use of appropriate sensors depending on input data.

The approach in [231] discusses the use of autonomic workflows (AWs) to self-manage processes based on service composition. An AW is an extended workflow that contains semantic information about the process to be executed, its objectives, and all related data and constraints that may be useful for the formulation of the process. The approach supports adaptation of autonomic workflows based on their life-cycle, i.e. inception, binding, and execution. In the work, the adaptation is based on the use of policies and reactions to process anomalies. Policies are pre-defined at a high level language as event-condition-action (ECA) rules and are kept in a knowledge base. The approach manages the workflow during all its life-cycle by collecting and organizing the information from the operating environment. The collected information is used together with semantic descriptions of services for adapting, reacting, and improving the workflow at run-time using the ECA rules. The approach also uses a Manager component to support some of the monitoring activities (e.g., checking execution, handling anomalies) of a workflow. When there are changes in the workflow, the Manager checks the new workflow using the ECA rules in the knowledge base.

In [76] a high-level model for adaptation is proposed. It focuses on the adaptation of a service composition taking into consideration global constraints, e.g. the execution of a service-based application within a time constraint. The approach relies on the creation of an organization model (OM) based on a BPEL specification. The OM involves the use of agents (each agent is bounded to a different service), models of complex collaborations between different agents, and specification of rules triggering adaptation. In the work, agents are used to establish the correct execution of a service in a service composition and to trigger the adaptation of the service composition. This is done by checking the information of the agents and verifying whether rules are being satisfied. When a rule is violated the agent is replaced (i.e., the replacing agent is bounded to a different service). In the approach, agents also contain monitoring mechanisms that allow them to pro-actively decide to stop participating in a service composition, e.g. an agent can monitor its SLO and predict than an SLA will be violated. When an agent decides not to participate in a service composition, a different agent is used to replace it. The approach assumes the existence of additional agents (bounded to services) for replacement. Monitoring in this approach uses information from the participating agents and the rules for triggering adaptation.

The work in [39] tackles automated evolution, repair, and tuning of services compositions. More specifically, it focuses on the automation of important aspects of a service-oriented application, including resource and service discovery, binding and composition, deployment, and monitoring. The authors propose a roadmap outlining selected research opportunities in the context of self-organizing SOAs. The work also suggests the concept of self-organizing service that is capable of managing its life-cycle (i.e. discovery, composition, and execution) in an autonomic way. The work uses an infrastructure to support self-organizing SOA to allow a user to specify desired QoS characteristics at the level of the service composition. These QoS characteristics can be automatically broken down into requirements for individual services.

An approach to achieve highly adaptable Web services through context-adaptable web service policies is presented in [243]. In this work, policies are sets of one or more monitoring rules. The work assumes that ,both policies and rules are adaptable based on context information. It extends the Web Service Policy Language (WSPL) to allow the specification of context at both policy and rule levels. The extended policies are woven into the service composition. The approach uses three operations for extending the policies, namely: *i)* a context specification method for specifying policy context, *ii)* a policy translator method for translating a policy to a required format, and *iii)* a policy integration method for applying context-based policies to web services.

An event-based framework for specifying and reasoning about monitoring properties is presented in [244]. The approach tackles the problem associated with the definition of suitable monitoring properties at the design phase (i.e. properties known in advance) by allowing them to be defined during design or execution times. The approach builds upon an event-based declarative composition design that serves as a unified framework to bridge the gaps among process design, verification, and monitoring. The framework has four main stages including: *i)* composition design, which involves the composition specification; *ii)* instantiation and verification, which involves finding a solution (or identifying conflicts) in the composition; and *iii)* execution of the process and composition monitoring, which involves verification and recovery. The framework is based on event calculus (a language based on first-order logic) that allows specifying and reasoning about monitoring properties in terms of events and fluents. The approach allows definition of functional and non-functional properties, identification of violations, and calculation of the effects that a violation may have on the overall process execution.

In [188] a proposal to provide a solution to service compositions is presented. It is sketched in three parts. In the first part user requirements are represented as a goal model; in the second part functional specifications and supervision directives are obtained from the goal model; and in the third part execution is supported through a suitable runtime infrastructure. The proposal assumes the adoption of a live global model that will be able to change at runtime. Goals operations can be automatically derived. This facilitates the derivation of the functional specifications, and allows for an automatic derivation of the supervision directives that must be applied at runtime. The proposal also considers complementing traditional goal models with approaches tailored to self-adaptive systems.

In [146] a business centric monitoring framework is proposed to bridge the gap between the business and service levels in complex business applications. The approach uses business information invariants (i.,e., fields of information in composite applications that remain unchanged) to define one or more monitor sets . Monitor sets are defined as collections of attributes and mappings from each attribute to one or more business item attributes in order to associate the service activity with the business composition execution. The user selects the components to be monitored from a business centric view. This allows the user to select and specify monitor sets for a business composition by binding a monitor set with inputs/outputs of a business component. Monitor models check the execution of business compositions using events generated by the service components.

The approach in [222] uses a monitor manager on top of existing monitoring tools to provide a policy driven interface for these tools. The policies describe how the monitoring infrastructure should react, e.g. selection of a particular monitor rule, when a modification occurs in the system. However, The rules specified in the monitoring tools cannot be modified.

Monitoring Adaptation Overview

We have noticed that existing approaches, which *grosso modo* can be classified as monitor adaptation, rely on the occurrence of events in a SBS as the triggering mechanisms for the adaptation of the monitor component. However, strictly speaking, these approaches do not consider a modification of the monitor component, but rather:

- Replacements of the monitoring mechanisms for (another) pre-defined monitoring mechanisms.
- Run-time model creation, e.g. based on state machines, for the verification of generic properties of a service composition.

Furthermore, given that the adaptation process is triggered by events related to the service compositions, the process ignores relevant and influential factors such as user context.

In general, the studied approaches rely - at least up to a certain level - on the use of pre-defined conditions and solving conflict strategies/techniques (e.g. use of agents) to perform the monitoring activity. Moreover, in the studied approaches the monitor adaptation is more of a means to an end than the end itself, i.e. SBS adaptation. Furthermore, it can be observed that approaches related to monitor adaptation focus on specific sets of factors and adaptation triggering mechanisms which are, most of the time, inapplicable even in similar scenarios.

2.6 Summary

From the previous sections it can be observed that, although there is a considerable amount of proposals dealing with SBS monitoring and SBS adaptation, there are still open issues that have not been thoroughly addressed. The following points summarise, in a broad manner[3] the existing gaps in the state of the art, we address in our research.

i) Depending on the scenario, there are different context factors to consider when modelling a SBS. These context factors can be represented as context models, where defined dimensions are used for organising contextual characteristics. Although there has been a considerable amount of research involving context, e.g. in terms of location or time, only a few have partially addressed the importance of the *user context* in SBSs.

[3] Detailed classifications have been provided at the end of each one of the previous subsections

ii) Overall adaptation approaches emphasise adaptation of service-based systems and do not tackle the issue of monitor adaptation.

iii) User characteristics and user interaction in service-based systems have not been considered important factors for monitoring of service-based systems.

Our work focuses on the HCI context aware monitor adaptation. It is concerned with the human interaction (*iii)* above) in a SBS and takes into consideration user characteristics that are likely to be present in a SBS (*i)* above) when verifying the execution of the system. It assumes the adaptation of the monitor component (*ii)* above) on its own, and as a consequence of the adaptation of the service-based system. Figure 1.1, from chapter 1, depicts the above relations.

In the following Chapter we identify a set of user context types. We present a model we created for the representation of user context types and we also address the issue regarding *human interaction* in SBSs. The information from the user is a key component for the monitor adaptation.

Chapter 3

Overview of the Approach for Service-based Systems

This chapter introduces the user context model we have developed to represent some of the main characteristics of a user when interacting with a SBS. It takes into consideration key aspects that should be considered when modelling context, provides a set of suitable context dimensions centred on the user, and presents the rationale behind our proposal. In this chapter we also describe a Web Organiser Service-Based System (Wo-SBS) scenario, which will used throughout the thesis to illustrate our work. We finish the chapter presenting our framework and explaining its different components.

3.1 Context Model

In section 2.3, we presented different approaches dealing with context modelling. Providing a classification of existing context models can, very easily, become quite a complex task. As stated in [43], since applications are so different, context modelling should be addressed independently according to the specific objectives. In fact, this claim is sustained by the existing approaches. In [149][194][221][241], for example, hierarchical structures are proposed to deal with context modelling. The aim in this proposal includes, besides providing a general taxonomy, further extensions of the current model according to the pre-defined classes. The latter allows for a further level of granularity of the model. A different type approach regarding context modelling has been proposed in [156][200], where context-dependent functionality is encapsulated in software modules.

In addition to the above, and according to [217], there are some key aspects that should be considered when modelling context, including:

- System boundaries, which are usually arbitrarily established.
- Definition of the context and dependencies the system has on its environment.

Taking the above into account, we provide a general context model (section 3.2) capable of *i)* being further extended and *ii)* be applicable to most SBS involving user interaction. It is important to notice that the main concern in this model is centred on, but not limited to, the user.

3.2 Model Dimensions

The model is grounded on previous research related to context modelling [1], including [81][110][162][196][212]. The main contributions of this context model are *a)* the focus on the user, providing a clear identification of those characteristics that - from our perspective - depend entirely on the user, and their separation from those characteristics that are related - but not entirely dependent - on the user, *b)* creation of a taxonomy that can be easily extended/complemented with additional context types.

[1] Including some of the approaches previously described in section 2.3

It is important to note, at this point, that the proposed model is - under nor circumstance - exhaustive, and can be further extended. In fact, we believe that relations might exist between different context types. This claim is substantiated by the research carried on in other fields (e.g. perhaps one of the most influential works dealing with human cognition has been given in the area of psychology, see [234]) where different user configurations may trigger different actions, or cause different behaviours. How these context types might be related goes beyond the scope of this work, nevertheless it is an issue to consider in the expansion of a user centred context model.

Our model proposes two general context type categories: *i)* direct user context types and *ii)* related user context types.

i) The direct user context types represent information of the characteristics of the users and include role, skills, need, preferences, and cognition context types.

ii) The related user context types represent information that may influence user information and include time, location, and environment context types.

In tables 3.1 and 3.2 we provide a description of the direct and related user context types respectively.

The aim of our context model is the formalization of the most relevant aspects characterizing a user interacting with a SBS. Nevertheless, aware a context taxonomy - based exclusively on the user - can be argued to be not realistically practical[2], we proposed two general dimensions. These two dimensions allow to focus on the user, i.e. the direct user context types, and the relevant user-related context types, i.e. the related user context types. Furthermore, we believe, as stated in [236], that

[2] For example, it is highly plausible a user can be associated to a specific location, providing thus extra contextual information

Context	Meaning
Role	It signifies a social behaviour of an individual within the domain of a SBS. The roles of an individual can be concerned with the accessibility to the system, occupation of the user, privileges that the user may have to the system.
Skills	It signifies the level of expertise of an individual with respect to a SBS. The skills of a user are directly related to the user knowledge and experience with the system. The skills can be defined in terms of the level of expertise of the user (e.g., beginner, average, advanced) or the years of experience.
Preferences	It signifies an individuals choice over pre-established alternatives of computational resources, of a SBS. Examples of these preferences are concerned with security, reliability, response time, and availability characteristics of a SBS.
Needs	It signifies what an individual wants or requires from a SBS.
Cognition	It signifies individuals characteristics associated with the process of thought. It is concerned with the way that individuals think, feel, or react. Examples of these characteristics are perception, user attention level, and user comprehensive ability.

Table 3.1: Direct User Context Types

the separation of context concerns helps dealing with hierarchical decomposition, avoiding thus overlapping or cross-cutting issues among different context types.

3.3 Model Specification - Ontology

In our work we created an ontology to represent the different user context types and their relation with a user.

An Ontology can be defined as a formal specification of terms, along with the existing relations among these terms, in a given domain [109]. An ontology can be

Context	Meaning
Time	It signifies all possible types of information related to the moment when the user interacts with a SBS such as hour, date, day, week, or season.
Location	It signifies information related to the place where the user interacts with a SBS such as coordinates, city, and country.
Environment	It signifies information concerned with the environment where the SBS is being used. This context includes information such as temperature, traffic conditions, or climate.

Table 3.2: Related User Context Types

used to represent any type of information, including unstructured (e.g. text), semi-structured (e.g. web pages) and structured (e.g. database) data [155]. It can also be used to represent the relations between the data. Furthermore an ontology allows to reason about the information it contains.

The use of ontologies for the specification of context models is not new; in fact there is a considerable amount of research that has been done in the area of context modelling, e.g. [55][144][177]. In our work, we developed an ontology using protégé [102][178] where the context types, previously described in section 3.2, are represented as classes.

A graphical representation of the ontology is shown in Figure 3.1. In the figure the different context types are represented as classes, with subclasses in cases of preferences and environment context types, and are associated with a central class representing the user. These associations indicate relationships between the different attributes of a context type, e.g. occupation for context class *role* and the *user* class. For each class their attributes and respective data types are presented inside the class.

The user class represents information about the user ranging from unique identification (i.e. user ID, user name) to profile information (i.e. sex, language, address), and the associations between a user class and the other context type classes.

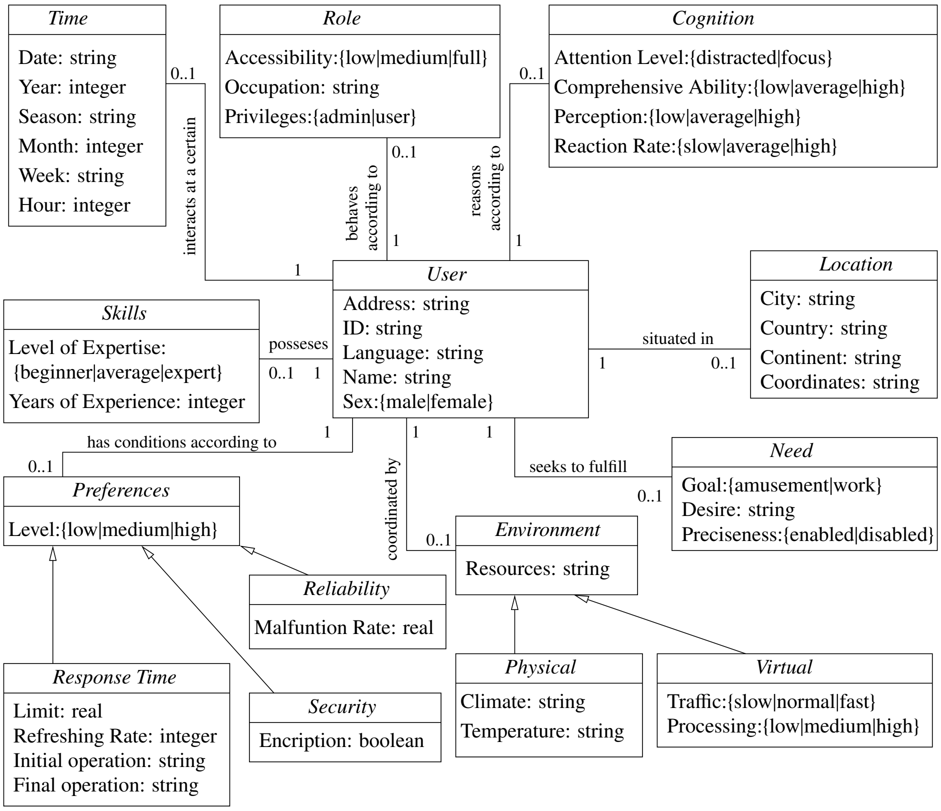


Figure 3.1: User Context Ontology

Context	Value
Cognition	reaction:{slow, medium, fast}, comprehension:{low, average, high}, perception:{low, average, high}, attention:{focus, distracted}
Environment	resources:string *Physical:* climate:string; temperature:string; *Virtual:* traffic:{slow, normal, fast} processing:{low, medium, high}
Location	coordinates:string; country:string; city:string
Time	year:integer; month:integer; hour:integer; minute:integer; second:integer; date:string; day:string; week:string; season:string
Need	desire:string; goal:{amusement, work}
Preferences	level:{low, medium, high}; *Response Time:* refreshing rate: integer; *Reliability:* malfunction acceptance: real; *Security:* encription: boolean
Role	occupation:string; accessibility:{low, medium, high}; privileges:{admin, user}
Skills	experience level:{beginner, average, expert}; years of experience:integer
User	name:string; sex:{male, female}; language: string; address:string; id: string

Table 3.3: Attributes and Values for the Different Context Types

Some of the attributes in the ontology are defined as symbols of values (e.g., low, medium, high; male, female; slow, normal, fast; beginner, average, expert), while other attributes support definition of specific values represented as string, integer, or real data types. Table 3.3 describes the different attributes for the different context types. Note that the context types *environment* and *preferences* include sub-types. The physical and virtual sub-types are related to the *environment* context type; while response time, reliability and security sub-types are related to the *preferences* context type.

3.4 Rationale, Compatibility and Benefits of the Model

As mentioned before, the use of ontologies for context modelling is not new. For example, in [61] an ontology for context-aware pervasive computing environments is presented. This ontology is centered on general concepts including people and places, and defines a set of properties and relationships associated with these general concepts. The main difference with respect to our ontology is that *i)* all the elements are defined according to a specific scenario and *ii)* most of the identified user context types are related to physical attributes. It is even possible to find ontologies that have been formulated considering the user as the main element, e.g. [105][176]. Similar to our model, in these ontologies a central class represents the user profile and is associated to the classes concerned with other user characteristics such as skills or abilities. However, our ontology contains more specific user context types. From an overall perspective, our ontology:

- Allows for a clear scenario-independent classification between those context types dependent on the user, and those that are not user-dependent.
- In relation to the user, provides a set of context types that are not only independent to the scenario but, given their generality, likely to be applicable to almost any user.
- Regarding compatibility, we found our ontology was semantically compatible with previous proposed ontologies, e.g [113]. As defined in [49] an ontology is semantically compatible with another ontology if the terms are supposed to mean the same thing in both ontologies.
- Regarding its evolution, our ontology is by no means exhaustive. As in [172] we consider the ontology to evolve based on further identification of context types.

3.5 Wo-SBS Scenario

In order to illustrate our work, we present a web-organizer service-based system (Wo-SBS) that will be used throughout the thesis. The Wo-SBS provides access to user email accounts, allows message exchange among different users logged in the Wo-SBS, and allows users to schedule activities in virtual agendas. The Wo-SBS can be accessed from different devices, e.g. desktops, PDAs, mobile phones. The following scenario presents the typical interactions that a user can perform in the Wo-SBS.

Mary is a 32 years old living in London and the director of a conference support company that helps with the organization of conferences in different parts of the world. Mary has been commissioned to organise a conference that will take place in one month in Oxford. Mary is going on holidays for a week in Italy, which she has organised four months ago. Given the new commissioned project, while away Mary wants to be able to monitor and coordinate any necessary activities for the forthcoming conference for which Mary is in charge. In addition, while being away Mary wants to be able to have access to her personal emails. In order to allow all the above requirements from Mary while in Italy, she subscribes to Wo-SBS application. Mary has not used this application before, but she heard from a colleague how good and helpful it is. During the time Mary is away she uses Wo-SBS to assist her with the organisation of the conference (as a "personal manager") and also to send emails to her family members and friends (as a "personal user"). Moreover, given that Mary is using the Wo-SBS application for the first time during the trip, her skills with the application are very basic. However, after three days of using WO-SBS during her holidays, Mary started to have a better understanding of the application and used advanced functions in the application such as creation of alarm events, link of these events with mobile devices, and cross checking information in different documents. At a certain point during Mary's holidays she needs to decide about the food to be *served during the banquet event of the conference. At this moment, her role as a user of the application changes to an "event coordinator". She receives quotes for five different types of possible menus and needs to cross check these quotes with the overall budget of the conference. However, although Mary's skills with the WO-SBS application is quite advanced at this stage, she is deciding about the banquet dinner for the conference in the evening after several glasses of wine and spending a whole day on a walking tour in the heat. Mary is very tired and her cognition level is "low", slowing down the coordination process. At this time, the meeting schedule service used by Wo-SBS is unavailable and a new meeting scheduler service is used by the system.*

Note that for the above scenario it is very unlikely that all the monitor rules necessary to verify the correct behaviour of the Wo-SBS, have been be pre-defined or are known in advance. For example, it is not possible to know monitor rules that are relevant to the new meeting scheduler that is used by the system. Similarly, it may be necessary to remove monitor rules that are specific to the original meeting scheduler used by the system that may become obsolete when the system is replaced. It may also be necessary to modify existing monitor rules to support the fact that the user, Mary, is using the system at a certain time when her *cognition* level is "low" (after having some wine and being on a walking tour in the heat during the day); or to automatically identify the relevant monitor rules when Mary uses the system for work or for personal communication with her family and friends.

3.6 User Model Example

From the ontology user models are obtained, described in an XML format, specifying user context types. We consider the concept of user models as defined in [96], i.e. "models that systems have of users that reside inside a computational environment".

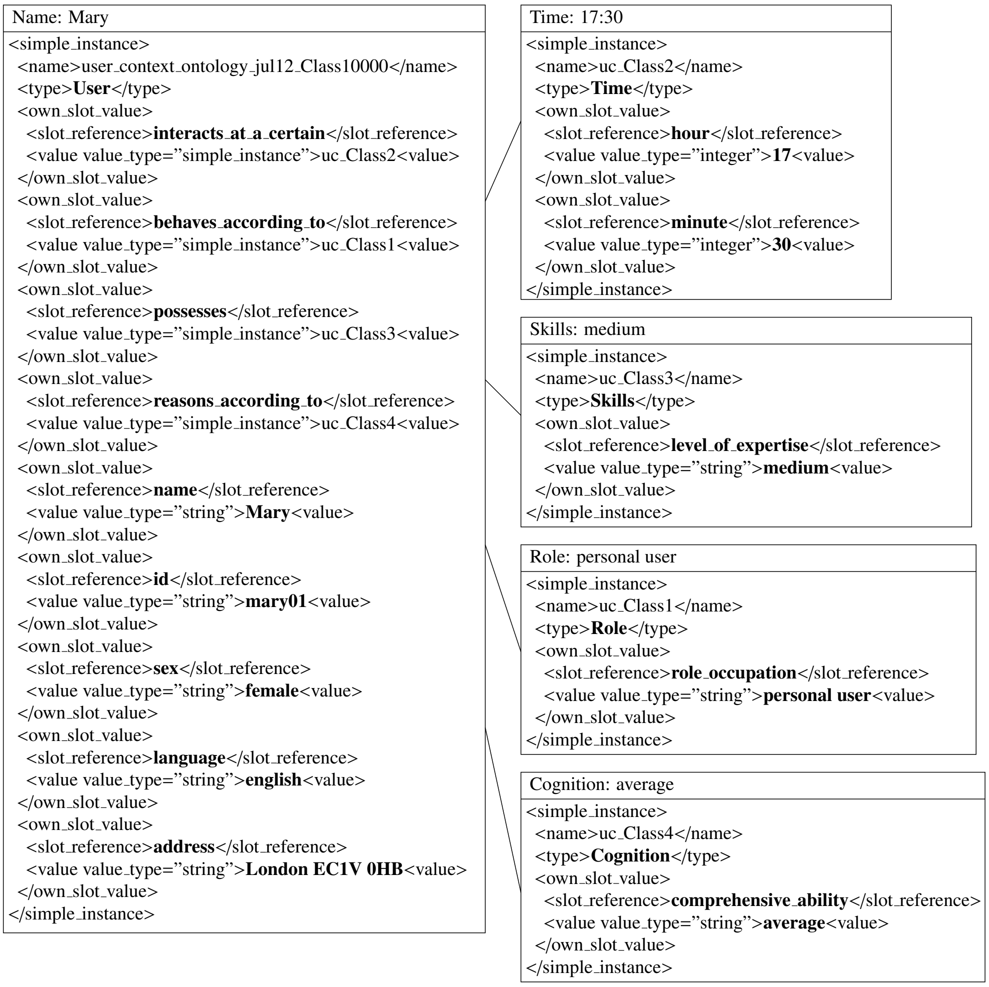
An example of the use of the ontology shown in Figure 3.1, to represent a user model for a Wo-SBS application (see Section 3.5), is depicted in Figure 3.2. In the user model, a user, *Mary*, is characterised by her name; her id: Mary01; her sex: female; her language: english; and her address: Northampton Square London EC1V 0HB. These attributes are represented in the *User* class. In addition, Mary has characteristics related to different context types which are represented in separate classes. More specifically the *Time* class represents the time Mary interacts with the Wo-SBS: at "17:30", and is related to the *User* class by the reference *interacts_at_a_certain*.The *Skills* class represents Marys level of expertise: "medium", and is related to the *User* class by the reference *possesses*. The *Role* class represents the role of Mary with respect to the Wo-SBS: "personal user", and is related to the *User* class by the reference *behaves_according_to*. Finally, the *Cognition* class represents Marys comprehensive ability with respect to the Wo-SBS: "average", and is related to the *User* class by the reference *reasons_according_to*. Note that each attribute has a specific data type. As shown in the ontology, all classes corresponding to user context types are related to the central *User* class.

It is important to note that the information provided by the context types is essential for the specification of monitor rules (Chapters 4 and 5). Monitor rules are used to verify the correct execution of a SBS according to the defined user context types.

3.7 User Interaction

Another aspect that needs to be taken into account, along with the user context, is the user interaction.

User interaction, or more specifically human computer interaction (HCI), is a field in computing dealing with the design, evaluation, and implementation of interactive computing systems for humans [114]. Due to its nature, it is very common to



Name: Mary

```
<simple_instance>
 <name>user_context_ontology_jul12_Class10000</name>
 <type>User</type>
 <own_slot_value>
  <slot_reference>interacts_at_a_certain</slot_reference>
  <value value_type=”simple_instance”>uc_Class2<value>
 </own_slot_value>
 <own_slot_value>
  <slot_reference>behaves_according_to</slot_reference>
  <value value_type=”simple_instance”>uc_Class1<value>
 </own_slot_value>
 <own_slot_value>
  <slot_reference>possesses</slot_reference>
  <value value_type=”simple_instance”>uc_Class3<value>
 </own_slot_value>
 <own_slot_value>
  <slot_reference>reasons_according_to</slot_reference>
  <value value_type=”simple_instance”>uc_Class4<value>
 </own_slot_value>
 <own_slot_value>
  <slot_reference>name</slot_reference>
  <value value_type=”string”>Mary<value>
 </own_slot_value>
 <own_slot_value>
  <slot_reference>id</slot_reference>
  <value value_type=”string”>mary01<value>
 </own_slot_value>
 <own_slot_value>
  <slot_reference>sex</slot_reference>
  <value value_type=”string”>female<value>
 </own_slot_value>
 <own_slot_value>
  <slot_reference>language</slot_reference>
  <value value_type=”string”>english<value>
 </own_slot_value>
 <own_slot_value>
  <slot_reference>address</slot_reference>
  <value value_type=”string”>London EC1V 0HB<value>
 </own_slot_value>
</simple_instance>
```

Time: 17:30

```
<simple_instance>
 <name>uc_Class2</name>
 <type>Time</type>
 <own_slot_value>
  <slot_reference>hour</slot_reference>
  <value value_type=”integer”>17<value>
 </own_slot_value>
 <own_slot_value>
  <slot_reference>minute</slot_reference>
  <value value_type=”integer”>30<value>
 </own_slot_value>
</simple_instance>
```

Skills: medium

```
<simple_instance>
 <name>uc_Class3</name>
 <type>Skills</type>
 <own_slot_value>
  <slot_reference>level_of_expertise</slot_reference>
  <value value_type=”string”>medium<value>
 </own_slot_value>
</simple_instance>
```

Role: personal user

```
<simple_instance>
 <name>uc_Class1</name>
 <type>Role</type>
 <own_slot_value>
  <slot_reference>role_occupation</slot_reference>
  <value value_type=”string”>personal user<value>
 </own_slot_value>
</simple_instance>
```

Cognition: average

```
<simple_instance>
 <name>uc_Class4</name>
 <type>Cognition</type>
 <own_slot_value>
  <slot_reference>comprehensive_ability</slot_reference>
  <value value_type=”string”>average<value>
 </own_slot_value>
</simple_instance>
```

Figure 3.2: Example of a User Model

```
...
    <bpel:invoke name="Login" partnerLink="login" operation="opSelectFeatureUserOperation" portType="ns:login"
    inputVariable="loginRequest" outputVariable="loginResponse"></bpel:invoke>
...
    <bpel:invoke name="Check Access" partnerLink="check access" operation="accessChecker"
    inputVariable="check accessRequest" outputVariable="check accessResponse"></bpel:invoke>
    portType="ns:CheckAccess" inputVariable="check accessRequest"
    outputVariable="check accessResponse"></bpel:invoke>
...
```

Figure 3.3: Extract from a BPEL Specification Including two Operations: *Login*, involving User Interaction and *Check Access*, not Involving User Interaction

see HCI related to other areas such as psychology or social science. As a result, it is also possible to find several definitions for HCI dealing with human interaction.

Regarding service-oriented architecture and computing, user interaction has been - principally - studied from the design process perspective. Different proposals can be found in the literature addressing HCI and SOA, e.g. [28][203][30]. Furthermore, there has also been an interest from the industrial community to introduce human interaction in SOA, e.g. [140]. Despite these efforts however, no standard has been yet agreed.

Because of the above, in our work we rely on a specific syntax, for the identification of those operations involving user interaction with the SBS application. More specifically, operations involving user interactions, and only those operations involving user interactions, are specified by a name including the prefix *"op"* and the suffix *"UserOperation"*.

Figure 3.3 shows an extract of the BPEL[3] specification containing two operations: *accessChecker* and *opSelectFeatureUserOperation*. According to the prefix and suffix in the operations; operation *opSelectFeatureUserOperation* corresponds to a user interaction, while operation *accessChecker* does not.

[3] As stated in chapter 1, BPEL is the *de facto* specification language for business processes

There have been approaches dealing with user interaction in service-based systems. In [44], for example, a meta-model was proposed to support the user interaction at run-time. The proposal, however, involves the extension of BPEL specification to include input and output user interactions. Our approach, on the other hand, does not involve further modifications to the BPEL specification, apart from the names of the operations.

Our proposal for identifying user operations in the BPEL specification has the following characteristics:

- It is simple.
- Relies on the name of the operations for the identification of user interactions. User operations must start with the prefix *"op"* and finish with the suffix *"UserOperation"*.
- It is based on a *de facto* language (BPEL) and does not require further modification.

3.8 Framework

The overview of the process associated with the monitor adaptation activity is shown in Figure 3.4 [4]. As shown in Figure 3.4 the process is iterative; changes in the SBSs or changes in the context types of the users accessing and interacting with the systems, trigger the need to identify, create, or modify monitor rules. These monitor rules will be used by a monitor tool to verify the correct behaviour of the SBSs. The identification of violations of the rules in the SBS also triggers the need to adapt the systems, which may require the creation, modification, or removal of monitor rules.

[4] Note that the diagram was previously introduced in Chapter 1

The framework can support different types of changes in SBSs specifications. Examples of these changes are: *i)* replacement of an operation, or a set of operations, by another operation or set of operations; *ii)* replacement of the operation types (e.g. a user operation is replaced by a service operation, or vice versa); *iii)* addition or reduction of the functionalities offered by the system represented in terms of operations, and *iv)* changes in other parts of the workflow of the SBS that are different from service operation replacement, e.g. addition of a condition.

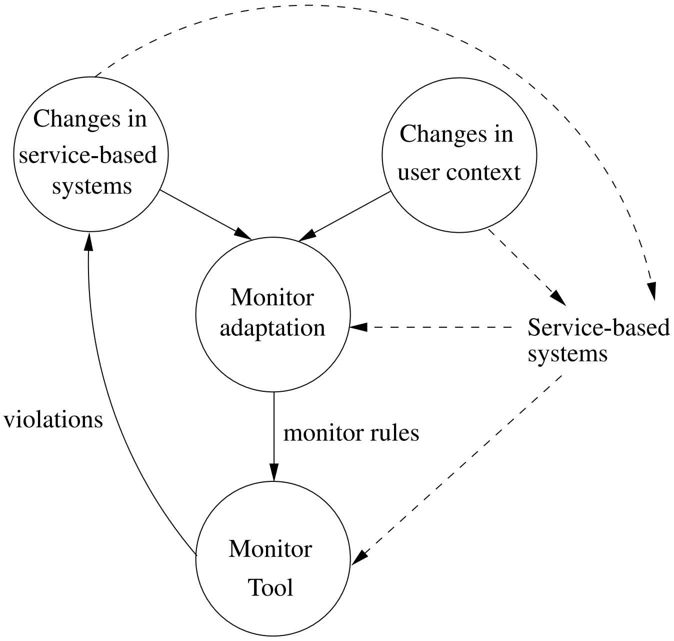


Figure 3.4: Relation between context, monitoring, adaptation, and SBSs

The monitor adaptation process is triggered by an event representing contextual information of a user. Based on this contextual information, the framework identifies relevant rule patterns representing the various context types of a user (see Section 4.2) and instantiates these patterns using the SBS specification, and time constraints represented in Service Level Agreements or historical execution time data for a service. The instantiated patterns are compared against monitor rules that may exist in a rule repository for a user of a certain SBS application. The framework assumes a different rule repository for each user accessing a system. Based on the comparison of monitor rules in the repository with the instantiated patterns, the framework *a)* identifies the monitor rules to be used if such rules exist; *b)* modifies existing rules in the repository that match the overall structure of the instantiated patterns, but have a mismatch with the time constraints in the pattern; *c)* creates new rules in the repository when there are no rules that match the instantiated patterns or the existing rules cannot be modified to match the instantiated pattern; or *d)* removes existing rules in the repository when they are no longer suitable. This last activity is executed by traversing the SBS specification and identifying the rules that do not match the operations in the system specification.

When a set of monitor rules suitable for the user and the SBS is identified, created, or modified these rules are used to monitor the system. It is also possible to have other monitor rules previously created for a system that are not concerned with the different context types relevant to our approach (e.g., general monitor rules regarding a functionality of the system). These rules are maintained in the repository and are not considered during the adaptation process.

Figure 3.5 shows an overview of the framework to support the monitor adaptation process. As shown in Figure 3.5, the main components of the framework are *Rule Adaptor*, *Path Identifier*, *Rule Verifier*, and *Monitor*. The framework also uses *Rule Patterns*, *Semi-instantiated Patterns*, *Monitor Rules*, *User Models*, *Service-based System (SBS) Specification*, *Annotations*, and *Service Level Agreements (SLAs)* or historical data.

In the framework we assume SBS specifications represented as BPEL [140] due to its widely use and acceptance; user models represented as an XML-based ontology; monitor rules, rule patterns, and semi-instantiated patters represented in Event Calculus [210]; annotations represented in an XML-based format that we have devel-

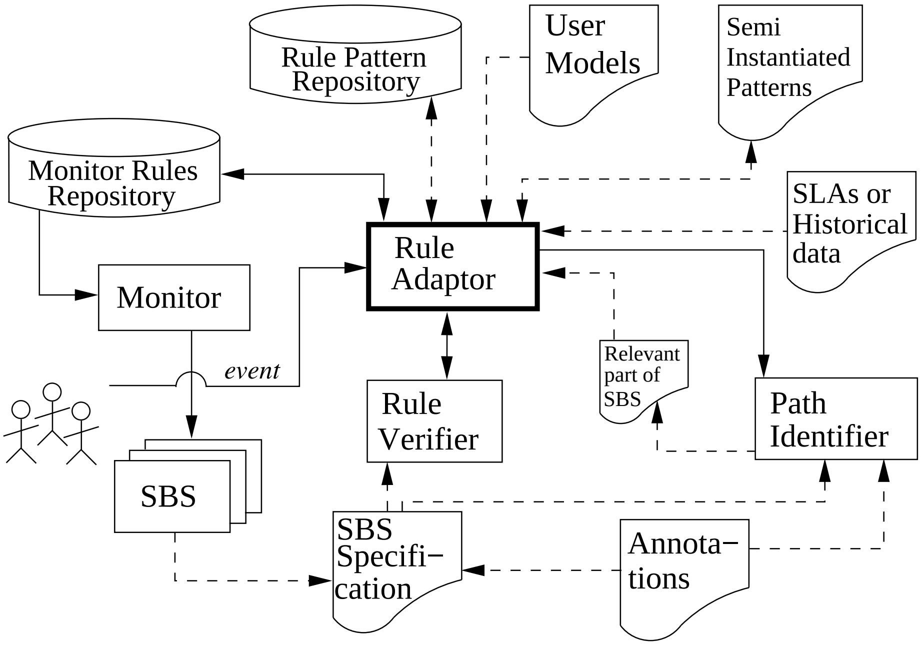


Figure 3.5: Framework architecture overview

oped; and SLAs represented in one of the SLA formalisms (e.g., WS-Agreement [8], SLang [147], WSLA [137]). The components of the framework are described below.

The *Rule Adaptor* is responsible for the identification, modification, creation, and removal of monitor rules. It receives events about changes in the context characteristics of the user or changes in SBS, and invokes the *Path Identifier* to identify paths in the specification of the SBS that are relevant to the received events.

The *Path Identifier* identifies and retrieves the parts in the specification that are related to the context types represented in the events. This is represented in Figure 3.5 as *Relevant Parts* of SBS. The identification of relevant parts of a SBS for certain context types is executed based on the use of *Annotations*. The *annotations* are special files containing information about context types, their instances, and the parts in the SBSs related to these context types. *Annotations* are created by developers, based on the requirements and domain of the system before the system is deployed. The *an-*

notation files are changed in the case of adaptation of SBSs due to new requirements or removal of existing requirements. For example, when there are new functionalities available for a certain context type, or even when there is a new instance of a context type for the system (a new role is created for the system). We provide more details about *annotations* in Section 5.2.

The *Rule Adaptor* uses context types from the events to identify relevant *Rule Patterns*, and instantiates these patterns with the identified information from the SBS specification and the *User Models*, representing characteristics of the users. As a result, *semi-instantiated rule patterns* are specified. The *semi-instantiated rule patterns* are patterns with some defined values *(events and fluents)*, but with undefined values for time variables or time gaps. The *Rule Adaptor* uses information provided by *SLAs* (or historical execution time data for a service, when available) to define time values. The assumption that SLAs will be available for participating services is not unrealistic since *SLAs* are currently used to establish business agreements between service providers and consumers. Moreover, the response times of a service or operations are attributes that appear in *SLAs*.

The *Rule Adaptor* uses the *semi-instantiated rule patterns* to identify monitor rules in the repositories. In the case where monitor rules that totally match the *semi-instantiated rule patterns* are identified, these rules are either used as they stand by the *Monitor* component or have their time values updated, when necessary, and subsequently used by the *Monitor*. In the situation in which no rules that match the *semi-instantiated rule patterns* are identified, new monitor rules are created based on the *semi-instantiated rule patterns*. In the case in which there are monitor rules that match invariant parts of the *semi-instantiated rule patterns*, the *Rule Verifier* checks if these rules are still valid for the SBS. In positive case, these rules are kept in the repository. Otherwise, these rules are removed from the repository and new rules based on the *semi-instantiated rule patterns* are created. The newly created rules are added into the repository and used by the *Monitor* component to verify the behaviour of SBS. Details about this process are described in Section 5.1.

In the framework, we use the monitor tool described in [220]. However, our approach can be used with other monitor tools that use monitor rules represented in Event Calculus [210]. The monitor tool receives requests from a service requester to verify, at regular intervals, the satisfiability of properties (represented as monitor rules) of a SBS. It intercepts run-time messages exchanged between a SBS and its services and verifies the satisfiability of the properties against these messages. It contains *a)* a service client that is responsible to invoke a service in a SBS; *b)* an event collector that is responsible to gather information during the execution of a SBS and the services deployed by the service based system, or information exchanged between the service client and its respective services; and *c)* an analyser that is responsible to check the satisfiability of the properties.

3.9 Summary

In this chapter we introduced our context model. The context model is based on the user context types we have identified. User context types are useful for the characterisation of a particular user. The characterisation for each user context type is performed in terms of general classes and attributes, which are likely to change from one user to another, as well as from one service composition to another.

We also described some user-related context types, which may provide complementary information. We introduced our Web Organiser Service-Based System (Wo-SBS) scenario and presented our solution for the identification of operations involving user interactions in service-based systems. Finally we presented and explained our framework and its different components.

In the next chapter we introduce the formalism, *Event Calculus (EC)*, for the specification of monitor rules. This is followed by the specification, for each user context type, of a set of monitor rule *patterns*. These *patterns* represent templates used for the specification of monitor rules and focus on a specific type of user context.

Chapter 4

Monitor Rules Specification: Event Calculus & Patterns

This chapter is divided in two parts. The first part, section 4.1, describes the formalism used for expressing monitor rules. Then, in section 4.2, we proceed to describe, our *pattern-based* approach for the specification of monitor rules considering different user context types.

The chosen formalism is Event Calculus (EC). The patterns, which are templates for the specification of monitor rules, cover the direct user context types described in section 3.2.

4.1 Event Calculus

Event Calculus is a formalism for reasoning about actions and changes [238]. It considers a set of predicates, actions, and time-varying properties for describing different situations - including actions and states - in a given scenario, over a defined period of time. In our work we use EC for the specification of monitor rules.

The use of EC [210] to describe monitor rules is not new. In fact it has been advocated in several works, e.g. [69][71][160][220], and has shown to be appropriate to support the representation of several types of rules. EC allows *i)* rules to be represented as first order logic, which provides sufficient expressiveness for a large range of applications, *ii)* specification of quantitative temporal constraints and relationships that are necessary to be taken into consideration when monitoring SBSs, *iii)* distinction between events and states that are necessary to describe the behaviour of a system and interaction of users with the system, *iv)* definition of the influences between events and states despite the possibility of using multiple states and events.

4.1.1 Advantages of Event Calculus

According to [141][158] there are several advantages of EC over other formal languages, including:

i) Easy evaluation. This is because the axioms may easily be represented as logic programs with negation as failure.

ii) Distinction between *events* and *states* by introducing a limited set of predicates; whereas other temporal logic languages, e.g. Computation Tree Logic (CTL), Linear Time Logic (LTL), Propositional Temporal Logic (PTL), allow the introduction of predicates with arbitrary meanings.

iii) In addition to the clear distinction between events and fluents, EC has a specific set of predicates that signify the occurrence of events and their effect on the initiation or termination of states in a system.

iv) Unlike most of the existing temporal logic languages, EC has an explicit time structure that allows users to specify complex quantitative temporal relationships, such as temporal distances between events.

v) The time structure in EC enables the expression of both, future and past properties, which is not permitted in some temporal languages such as PTL.

vi) Unlike pure state-transition representation, EC has an explicit time structure that does not depend on any sequence of events under consideration. This characteristic allows EC to model a wide range of event-driven systems.

4.1.2 Events, Fluents and Predicates

EC is based on a first-order predicate calculus capable of representing a variety of phenomena. It makes use of *events* and *fluents*, over a period of time, for representing the behaviour of a system. More specifically, an event represents an action occurring at a specific instance of time and may change the state of a system. A fluent is a condition of a system state and may be affected by the occurrences of events. A fluent can be seen as anything whose value is subject to change over time. Both *events* and *fluents* are represented in EC using *predicates*.

Predicates are used to *(i)* specify what happens when, *(ii)* describe an initial situation, *(iii)* describe the effects of an *event* (action), and *(iv)* specify which *fluent* (state) holds at a given time. Table 4.1 introduces the predicates used in EC, a detailed explanation of the predicates is provided below. Further information can be found in [210].

The occurrence of an event $event_a$, at some time t, is represented by the predicate $Happens(event_a, t, R(t_1, t_2))$, which means the event $event_a$ occurs at a time t, where t is within an interval of time defined between t_1 and t_2. The time boundaries represented by t_1 and t_2, can be specified using time variables or arithmetic expressions over time variables, and represent the lower and upper time boundaries.

The predicate $Initiates(event_b, fluent_a, t)$ represents the initialisation of the fluent $fluent_a$ triggered by the occurrence of the event $event_b$ at a time t.

Predicate	Meaning
$Happens(event_a, t, R(t_1, t_2))$	Occurrence of an event $event_a$ within a time interval defined by t_1 and t_2
$Initiates(event_b, fluent_a, t)$	Initialisation of a $fluent_a$
$Initially_P(fluent_b)$	fluent $fluent_b$ holds from the beginning
$Initially_N(fluent_c)$	fluent $fluent_c$ does not hold from the beginning
$HoldsAt(fluent_d, t_3)$	fluent $fluent_d$ holds at a time t_3
$Clipped(t_4, fluent_e, t_5)$	fluent $fluent_e$ is terminated within a time interval defined by t_4 and t_5
$Declipped(t_6, fluent_f, t_7)$	fluent $fluent_f$ is initiated within a time interval defined by t_6 and t_7
$Terminates(event_c, fluent_f, t_8)$	fluent $fluent_f$ ceases to hold at t_8

Table 4.1: Event Calculus Predicates

The predicate $HoldsAt(fluent_d, t_3)$, means the fluent $fluent_d$ holds, i.e. the state $fluent_d$ represents is valid, at a time t_3.

The predicate $Terminates(event_c, fluent_f, t_8)$ represents the finalisation of the fluent $fluent_f$ as a consequence of the occurrence of the event $event_c$ at a time t_8.

The predicate $Initially_P(fluent_b)$ means fluent $fluent_b$ holds at a time-point 0. On the other hand, the predicate $Initially_N(fluent_c)$ means the fluent $fluent_c$ does not hold at a time-point 0.

The predicates *Declipped* and *Clipped* means, respectively, the initialisation and finalisation of a fluent in a given time gap.

The predicates before described, can be combined to create elaborated EC formulae. In order to do so, additional relational operators are used including:

- The relational symbols *less than* $<$, *greater than* $>$, *less than or equal to* $\leq$, *greater than or equal to* $\geq$, and *is equal to* $=$, to express time conditions. For example, the statement $t_1 < t_2$ is true if the time instance t_1 occurred before t_2.

- The implication symbol $\Rightarrow$; where the statement $A \Rightarrow B$ means A entails B.
- The logical conjunction $\wedge$; where the statement $A \wedge B$ is true if and only if A is true and B is true.
- The negation $\neg$; where the statement $\neg A$ produces true when A is false and false when A is true.

In order to relate the various predicates together a suitable set of axioms is required

4.1.3 Event Calculus Axioms

An axiom is defined as a set of relations called premises and a conclusion. Given the premises an axiom unequivocally yields a relation that holds as a conclusion [41]. In EC a suitable collection of axioms relate the various predicates together and are used to represent domains involving actions with indirect effects and actions with non-deterministic events [210]. The EC axioms are presented in Table 4.2.

4.1.4 Example

The following example is based on the Yale shooting problem, originally described in [112] and retaken in [210]. In this scenario a *turkey* is initially *alive* and a *gun*, which will be eventually used to *shoot* at the turkey, is initially *unloaded*. The turkey will remain alive as long as a shooter does not successfully shoot at it. In this scenario the state of the turkey, i.e. whether it is alive or not, can be represented by a fluent *alive*. In this scenario the initial state of the turkey (*alive*), can be expressed in EC by the formula shown in Table 4.3.

Axiom EC1: $Initially_P(fluent) \land \neg Clipped(0, fluent, t) \Rightarrow HoldsAt(flt, t)$ Meaning the fluent $fluent$ holds at a time t if it held from time 0 and it was not terminated between the interval specified between 0 and t
Axiom EC2: $Happens(ev_1, t, R(t_1, t_2)) \land Initiates(ev_1, flt, t) \land \neg Clipped(t_1, flt, t_3) \land t_2 < t_3 \Rightarrow HoldsAt(flt, t_3)$ Meaning the fluent flt holds at time t_3 if an event ev_1 initiated flt at a time t and the flt was not terminated between t_1 and t_3
Axiom EC3: $Happens(ev_1, t, R(t_2, t_3)) \land Terminates(ev_1, flt, t) \land t < t_3 \land t_2 < t_4 \Leftrightarrow Clipped(t, flt, t_4)$ Meaning a fluent flt ceases to hold between t and t_4 if an event ev_1 terminates it at a time t. Vice-versa a fluent flt that does not hold between a time t and t_4 has been previously terminated by an event ev_1 at a time t
Axiom EC4: $Initially_N(flt) \land \neg Declipped(0, flt, t) \Rightarrow \neg HoldsAt(flt, t)$ Meaning a fluent flt that does not hold from time 0 and has not been initiated between a time 0 and t, does not hold at a time t
Axiom EC5: $Happens(ev_1, t, R(t_2, t_3)) \land Terminates(ev_1, flt, t) \land \neg Declipped(t, flt, t_3) \land t_2 < t_3 \Rightarrow \neg HoldsAt(flt, t_3)$ Meaning a fluent flt does not hold at a time t_3 if an event ev_1 terminates it at a time t and flt has not been initiated between a time t and t_3
Axiom EC6: $Declipped(t_1, flt, t_4) \Leftrightarrow Happens(ev_1, t, R(t_2, t_3)) \land Initiates(ev_1, flt, t) \land t_1 < t_3 \land t_2 < t_4$ Meaning a fluent flt is initiated between a time t_1 and t_4 if an event ev_1 initiates it at a time t; vice-versa if an event ev_1 occurs at a time t, the fluent flt is initiated between a time t_1 and t_4
Axiom EC7: $Happens(ev_1, t, R(t_1, t_2)) \Rightarrow t_1 \leq t_2$ Meaning the time range specified for the occurrence of an event ev_1 ranges from a specific instant (if $t_1 = t_2$) to a defined range (when $t_1 \neq t_2$)

Table 4.2: Event Calculus Axioms

$Initially_P(alive)$

Table 4.3: Initiation of fluent *alive*

Turkey dies (successful shoot)	Turkey lives (unsuccessful shoot)
$Happens(shoot, t_2, R(t_1, t_3)) \wedge$	$Happens(shoot, t_2, R(t_1, t_3)) \wedge$
$HoldsAt(loaded, t_2) \wedge$	$HoldsAt(loaded, t_2) \wedge$
$Happens(succss\_ht, t_x, R(t_2, t_4)) \Rightarrow$	$\neg Happens(succss\_ht, t_x, R(t_2, t_4)) \Rightarrow$
$Terminates(succss\_ht, alive, t_x)$	$\neg Clipped(t_x, alive, t_3)$

Table 4.4: Termination (Left) and Continuity (Right) of the Fluent *alive*

$Happens(shoot, t_2, R(t_1, t_3)) \Rightarrow HoldsAt(loaded, t_2)$

Table 4.5: Conditioning for the Occurrence of the *shoot* Event

Assume a loaded gun is represented by the fluent *loaded*. Assume also the action representing the shooting at the turkey is represented by the event *shoot*, and the action representing the turkey being successfully hit is represented by the event *succss_ht*. The two possible states of the turkey after a shoot, can be represented by the two formulas in Table 4.4. In the Table, the formula on the left describes the death of the turkey, as a consequence of a sequence of *events* and *fluents* leading to the successful shooting of a loaded gun[1]. The formula on the right, on the other hand, describes the turkey alive, after the shooting, as a consequence of an unsuccessful shooting, which is represented by the predicate $\neg Happens(succss\_ht, t_x, R(t_2, t_4))$.

From the formulae in Table 4.4, it can be observed that by the time the *shoot* happens (predicate $Happens(shoot, t_2, R(t_1, t_3))$) the gun is loaded (predicate $HoldsAt$ $(loaded, t_2)$). In fact it does not make much sense to shot an unloaded gun, i.e. shooting while fluent *loaded* does not hold. The condition that a gun must be loaded in order to be shot, can be expressed in EC as shown in Table 4.5.

Furthermore, the formula expressed in Table 4.5 can be used in combination with the formulae described in Table 4.4. Even more, since the state of the fluent *loaded* is being verified to hold by the time of the shooting (by formula in Table 4.5), the

[1] In this example the turkey dies instantly after receiving a shoot

formulae in Table 4.4 can be modified by removing the predicate $HoldsAt(loaded, t_2)$ from each formula.

Considerations

From now on, we will refer to each EC formula as a monitor rule. In addition, and following the definition given in [158], we differentiate between two parts within a monitor rule: a *body* and a *head*.

- A *body* signifies the antecedent (condition) of the formula
- A *head* signifies the consequence (implication) of the formula

For example, in Table 4.4 the *body* of each formula consists of a set of EC predicates including $Happens$ and $HoldsAt$. The *head*, on the other hand, consists of the predicate $Terminates$ (for the case of a successful shoot) and $\neg Clipped$ (for the case of an unsuccessful shoot). Both, *body* and *head*, must have a predicate which may be followed by zero or any number of predicates or time constraints (e.g. $t_1 < t_2$), each separated by a logical operator.

In our work an event represents the *invocation* or *response* of an operation. Since each invocation is likely to be followed by its correspondent response we differentiate between two types of events in EC formulae, *i)* those events representing requests to operations, and *ii)* those events representing responses from operations. An event representing a request for an operation, can be identified by an "*ic_*" prefix. Events representing responses from an operation, can be identified by an "*ir_*" prefix. For example, the occurrence of an event $ic\_getStatus$ would represent the request of the operation $getStatus$. Respectively, the occurrence of an event $ir\_getStatus$ would represent the response of the operation $getStatus$.

4.2 Patterns

Several approaches can be found in the literature dealing with patterns and their use in software design, e.g. [100][101][208]. Most of them rely on the use patterns to specify properties, e.g. monitor rules, verifying the correct execution of a system. The majority of these approaches is related to distributed, concurrent, and real-time systems, e.g [83][84][209][215]. Within these approaches, it is also possible to find proposals focusing on SOA, more specifically, on the use of patterns in SBSs. In [38], for example, the results of a study, analysing different specification patterns for SBSs, are presented. The study proposes a classification for different groups of patterns in SBSs, ranging from systems of specification patterns to service provisioning patterns.

In the particular case of EC-based pattern approaches, there have been proposals dealing with security issues, e.g. [92][225], focusing on electronic and integrated control systems. Likewise, it is possible to find EC-based pattern approaches, addressing security issues in SBSs, e.g. [5][218]. The work in [124] proposes EC-patterns in HCI. In this work patterns are used to support verification of physical interaction, where *events* and *fluents* represent general actions and physical states.

In our work we rely on the use of patterns for the specification of monitor rules. More specifically, we propose for each one of the direct user context types (see Section 3.2), a set of monitor rules patterns.

A monitor rule pattern specifies the structure of the monitor rule in terms of predicates and time constraints, without defining specific *events*, *fluents* or time gaps. A monitor rule pattern focuses on the property the monitor rule should verify for a particular context type.

The main characteristics of our rule patterns are:

- They are based on the EC formalism.

- They address specific properties of the user context types.
- They are general enough to be applicable to almost any SBS.
- They consider user interaction in the execution of a SBS.
- The pattern representing a certain context type can be easily identified based on the pattern's underlying (unique) structure.
- They allow for the generation of *templates* (see Section 4.2.1) useful for identifying, modifying, creating or removing monitor rules.

4.2.1 Patterns Conventions & Considerations

In our framework, patterns and monitor rules are described in EC [210]. Patterns consist of two different parts, an *assumption* part and a *rule* part. The monitor rule part represents the property, of a SBS, that needs to be monitored, e.g. the occurrence of an event $Event_1$ is followed by the occurrence of an event $Event_2$ in a defined period of time. The assumption part represents additional information about the service composition in terms of the state of the system, e.g. the occurrence of an event $Event_3$ triggers the initialisation of a state $State_3$.

A pattern is applied to a specific part of a SBS. More specifically, a pattern is applied to the part of the SBS that is related to the context specified for the user (see Section 5.2).

The application of a pattern to a particular part of a SBS, results in the specification of the desired monitor rule for that part of the SBS. More specifically, the instantiation of a *pattern* with information retrieved from the specification of a SBS, i.e. pertinent *events*, *fluents* and time constraints, allows to specify all the information needed for a monitor rule.

Patterns are defined in a general way in order to be applicable to different types of service-based systems. Both rule and assumption parts in a pattern have *invariant* and *variant* parts. The *invariant* part in a pattern does not change with the possible instantiations of the pattern. Contrary, the *variant* part changes with the instantiations of the pattern and depends on the service-based system application. Table 4.6 presents a general template for the various types of patterns used in the framework. In the figure, the *invariant* part for the pattern is represented in bold, while the *variant* part is not. As shown in Table 4.6, a pattern is specified in terms of implications of conjunctions of the different EC predicates described in section 4.1.2. More specifically, an *Event_Predicate* represents the EC predicate *Happens*, an *Event_Fluent_Predicate* represents the EC predicates *Initiates* or *Terminates*, and a *Fluent_Predicate* represents the EC predicate *HoldsAt*.

Rule part
$\boldsymbol{Event\_Predicate}(event, t, R(t_1, t_2)) \mid \boldsymbol{Fluent\_Predicate}(fluent, t)$
$(\wedge ... \boldsymbol{Event\_Predicate}(event, t, R(t_1, t_2)) \mid \boldsymbol{Fluent\_Predicate}(fluent, t))^*$
$\Rightarrow$
$\boldsymbol{Event\_Predicate}(event, t, R(t_1, t_2)) \mid \boldsymbol{Fluent\_Predicate}(fluent, t)$
$(\wedge ... \boldsymbol{Event\_Predicate}(event, t, R(t_1, t_2)) \mid \boldsymbol{Fluent\_Predicate}(fluent, t))^*$

Assumption part
$\boldsymbol{Event\_Predicate}(event, t, R(t_1, t_2)) \mid \boldsymbol{Event\_Predicate}(event, t, R(t_1, t_2))$
$\Rightarrow$
$\boldsymbol{Fluent\_Predicate}(fluent, t)$

Table 4.6: General Pattern Template

We have created several rule patterns to represent each of the different context types. They have been presented in previous publications, [71][72][70] and consider time units in seconds.

Having different types of patterns for a context type allows us to consider monitor rules specified differently. For instance, for a context type some patterns can rely on the use of events and fluents, while for the same context type other patterns can rely solely on the use of events. Furthermore, having different patterns for a context type, also increases the possibility of identifying rules matching the patterns specified for the context type.

In addition to the above, it should be noted that while in some cases it is possible to specify several patterns for a context type (e.g. role), in other cases the amount of specified patterns is reduced (e.g. need). This can be explained by the complexity of the property that needs to be verified, the flexibility of the language (EC), and the fact that the patterns need to be general enough so they can be applied to different service-based systems in several scenarios. In any case, we believe the set of proposed patterns can be further expanded.

4.2.2 Categories and Design Criteria

As mentioned before, for each context type we provide a set of rule patterns. Each pattern corresponds to a template of a monitor rule, verifying a specific property for the particular context type. It is important to note that each set of patterns for a context type is not intended to be complete. Each set is in fact a representative sample of possible patterns for a context type and can be further expanded.

The design of the set of patters propose in this work was influenced by the following criteria:

- Global and individual rules. In order to provide flexibility in our work, some of the rule patterns have been designed as templates of monitor rules considering the occurrence of all events and participating states in a single specification (i.e. a single monitor rule). In addition, since there are different ways of expressing the properties of a monitor rule - e.g. by decomposing it - we have also designed rule patterns considering the occurrence of events and participating states separately.

- Treatment of flows. It has been observed - from several case studies - that for most specifications, the sequence and order of the involved operations in a service composition is known. However, for those cases in which the events and states might be triggered in a different order - as when executing services specified in a flow - we have created patterns capable of dealing with such behaviour.

- Monitoring based on the occurrence of events, states of the system, or a mix of states and evens. The consideration of events and states allows for the specification of patterns (and therefore rules) capable of verifying the behaviour of a service composition in different ways. More specifically, when monitoring states, it is possible (up to a certain extent) to rely on the occurrence of events for the specification of a monitor rule concerned with states. In a similar way, when monitoring the occurrence of events, it is also possible to rely on states for the specification of a monitor rule concerned with events. Furthermore, it is also possible to use a combination of events and states.

- Generalisation. It can be argued that the patterns can be further expanded, including complex templates for the specification of monitoring rules verifying specific behaviour for particular service compositions. However, we believe that the specification of complex patterns would not be completely suitable for most service compositions.

An important aspect concerned with our patterns, is their application in a service composition. The use of our pattern approach relies on a crucial component, the *Path Identifier* (see Chapter 5). This component allows to identify the part(s) of the service composition related to a specific user context. More specifically, given the user context (i.e. one or more defined context types), the *path identifier*, determines which branch of the service composition should be executed. The selection of the branch to be executed involves the resolution of conflicts in which two or more branches are related to the same context type(s).

Finally the patterns in this work do no consider loops. Although their identification in a service composition and their representation in terms of patterns can be specified (for example, in terms of a states), it is not possible to know *a priori* the exit condition for the loop. Similar to the *generality* criterion, we believe that it would be possible to specify rules associated to loop conditions, however it would also imply an analysis and observation of the behaviour of a service composition.

In what follows we describe and provide an example for each one of the patterns for the different user context types.

4.2.3 Role Patterns

Patterns for *role* context types have been created based on the fact that a specific *role* of a user may activate different parts of a system. More specifically, for a certain *role* of a user, different operations in the system representing the functionalities that are concerned with that specific *role* should be invoked. In the approach this is represented by the fact that an event in a SBS, should be followed by the invocation of operations related to a certain *role*.

Role Rule Pattern 1

Consider the extract of the part of the Wo-SBS application (see Section 3.5) associated with *role* "personal user", together with the operations to be invoked and their order, shown in Figure 4.1 in a diagrammatic way. In Figure 4.1, the nodes represent the operations and the respective amount of time that each operation should take to be executed, while the arrows represent the sequence in which the operations are executed. As shown in Figure 4.1, *oploginUserOperation*, *checkAccess*, *enableMessagingService* and *opselectFeatureUserOperation* represent a sequence of operations that are invoked every time the Wo-SBS is accessed; while operations *mailReview*, *mailComposer* and *mailManagement* represent operations that are invoked when a user in the *role* of a "personal user" accesses the Wo-SBS. A full specification of the Wo-SBS application in BPEL [120] can be found in Appendix A.

A pattern for a *role* context type is shown in Table 4.7. Table 4.8 presents the instantiation of the pattern in Table 4.7 for a set of operations related to *role* "personal user" in a web organizer SBS (Wo-SBS) application.

Rule part
$\boldsymbol{Happens}(\boldsymbol{ic}\_InitialEvent, t_1, R(t_1, t_1)) \Rightarrow$ $\boldsymbol{Happens}(\boldsymbol{ic}\_Event_1, t_2, R(t_1, t_1 + t_n{}^1))$ $\wedge ... \wedge$ $\boldsymbol{Happens}(\boldsymbol{ic}\_Event_i, t_{i+1}, R(t_i, t_1 + t_n{}^i))$

Assumption part
$\boldsymbol{Happens}(ic\_Event_1, t_1, R(t_1, t_1)) \Rightarrow \boldsymbol{Initiates}(\boldsymbol{ic}\_Event_1, Event_1, t_1)$ ... $\boldsymbol{Happens}(ic\_Event_i, t_1, R(t_1, t_1)) \Rightarrow \boldsymbol{Initiates}(\boldsymbol{ic}\_Event_i, Event_i, t_1)$

Table 4.7: Role Rule Pattern 1

The rule part of the *role* pattern in Table 4.7 states that the occurrence of the initial event in the system (*ic_InitialEvent*) is followed by the request of a sequence of operations (ic_Event_1, ..., ic_Event_i) within an interval of time ($t_i, t_1 + t_n{}^i$). An ic_Event_k ($1 \leq k \leq i$) corresponds to the invocation of an operation identified in the SBS specification that is associated with the specific *role* of the user.

The value of each time variable $t_n{}^l$ ($1 \leq l \leq i$) is computed as the sum of the maximum execution times of those service operations preceding the operation of interest in the service specification. The time variable $t_n{}^l$ also considers small time delays between consecutive events. More specifically, it considers small time gaps between the response from an invocation of an operation and a request for another operation. In our work, we considered time gaps of ten milliseconds (see Figure 4.1) based on our experiments and the work conducted in [158]. These time gaps can be easily readjusted if necessary.

The assumption part of the *role* pattern in Table 4.7 states that each request of an operation ic_Event_i triggers the initialisation of the state $Event_i$, which represents the active state of the operation. Assumptions are instantiated for each operation identified in the part of the SBS related to the *role* of the user. The invariant part for the *role* context type pattern in Table 4.7 is depicted in bold.

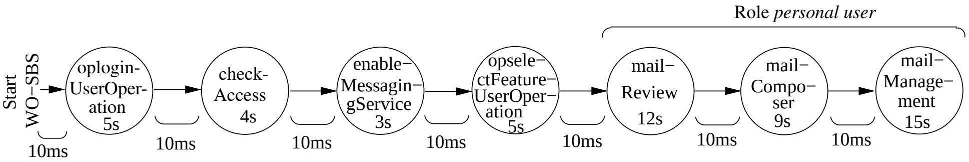


Figure 4.1: Service Specification Sequence for the Role of a *personal user*

The rule part in the example in Table 4.8 specifies that the request for operations *mailReview*, *mailComposer*, and *mailManagement* (events *ic_mailReview*, *ic_mailComposer*, *ic_mailManagement*) must happen at times t_2, t_3, and t_4 respectively after the occurrence of the initial event ($ic\_startWoSBS$) at time t_1, with $t_1 \leq t_2 \leq t_1+17050$ milliseconds ($0.01s+5s+0.01s+4s+0.01s+3s+0.01s+5s+0.01$s), $t_2 \leq t_3 \leq t_1+29060$ milliseconds ($0.01s+5s+0.01s+4s+0.01s+3s+0.01s+5s+0.01s+12s+0.01$s), and $t_3 \leq t_4 \leq t_1 + 38070$ milliseconds ($0.01s+5s+0.01s+4s+0.01s+3s+0.01s+5s+0.01s+12s+0.01s+9s+0.01$s). The upper bound time constraints are computed based on the times to execute the operations preceding the request of the operation of interest in the service-based system and the respective time gaps between operations (see the upper bound time limits for the execution of each operation in Figure 4.1).

Rule part
$Happens(ic\_startWoSBS, t_1, R(t_1, t_1)) \Rightarrow$ $Happens(ic\_mailReview, t_2, R(t_1, t_1 + 17.05))\wedge$ $Happens(ic\_mailComposer, t_3, R(t_2, t_1 + 29.06))\wedge$ $Happens(ic\_mailManagement, t_4, R(t_3, t_1 + 38.07))$

Assumption part
$Happens(ic\_mailReview, t_1, R(t_1, t_1)) \Rightarrow$ $Initiates(ic\_mailReview, mailReview, t_1)$
$Happens(ic\_mailComposer, t_1, R(t_1, t_1)) \Rightarrow$ $Initiates(ic\_mailComposer, mailComposer, t_1)$
$Happens(ic\_mailManagement, t_1, R(t_1, t_1)) \Rightarrow$ $Initiates(ic\_mailManagement, mailManagement, t_1)$

Table 4.8: Instantiation of Role Rule Pattern 1

Role Rule Pattern 2

Another role rule pattern is presented in Table 4.9. In the pattern, instead of considering all the invocations at once, a monitor rule and an assumption are generated for each operation related to the *role* of the user. The reason behind this is that, it is possible to find monitor rules specified in different ways where the semantic of a set of monitor rules is equivalent to the semantic of a single monitor rule, e.g. a monitor rule can be decomposed in several monitor rules. By providing variations of rule patterns verifying the same behaviour, we increase the possibility of finding suitable monitor rules for the monitoring activity.

Rule part
$\boldsymbol{Happens}(\boldsymbol{ic}\_InitialEvent, t_1, R(t_1, t_1)) \wedge$ $(\boldsymbol{Happens}(\boldsymbol{ic}\_Previous\_Event, t_2, R(t_1, t_2)))^* \Rightarrow$ $\boldsymbol{Happens}(\boldsymbol{ic}\_Event, t_{2+(1)^*}, R(t_{1+(1)^*}, t_{1+(1)^*} + t_n))$

Assumption part
$\boldsymbol{Happens}(\boldsymbol{ic}\_Event, t_1, R(t_1, t_1)) \Rightarrow$ $\boldsymbol{Initiates}(\boldsymbol{ic}\_Event, Event, t_1)$

Table 4.9: Role Rule Pattern 2

The rule part of the *role* pattern in Table 4.9 states that the occurrence of the *initial event* in the system (*ic_InitialEvent*) followed by the invocation of an operation related to the *role* of the user (*ic_Previous_Event*), must be followed by the invocation of another operation (*ic_Event*) related to the *role* of the user, in no more than t_n seconds.

The brackets ()* for the predicate $Happens(ic\_Previous\_Event, t_2, R(t_1, t_2))$ in the body of the rule part of the pattern, indicate the predicate instantiation is conditioned. More specifically, the predicate is instantiated in a monitor rule only if the invocation of the operation of interest related to the *role* of the user, *ic_Event*, is preceded by another operation invocation (*ic_Previous_Event*) also related to the *role* of the user. In the case no operation related to the *role* of the user is identified as occurring before the operation of interest (*ic_Event*), the predicate is omitted.

Note the above is also valid for the time variables specified in the predicate related to the occurrence of the *ic_Event* in the head of the rule part of the pattern.

The event (*ic_Event*) corresponds to the invocation of an operation belonging to the part of a SBS specification that is associated with the specific *role* of the user. The assumption part of the *role* pattern in Table 4.9 states that each request of an operation (*ic_Event*) triggers the initialisation of the state *Event*, which represents the initialisation of the operation.

The value of the time variable t_n is computed as the sum of the execution times of those service operations preceding the operation of interest in the service specification related to the *role* of the user. In the case there are no operations related to the *role* of the user occurring before the invocation *ic_Event* the time variable considers all previous operations. In the case there is a previous operation related to the *role* of the user, the time variable considers the time of the previous operation. The time variable t_n also considers small time delays between consecutive events (ten milliseconds).

Note that the rule part and the assumption part of the pattern are repeated for all the operations identified in the part of the SBS related to the *role* of the user. The invariant part for the *role* context type pattern in Table 4.9 is depicted in bold.

Rule part
$Happens(ic\_startWoSBS, t_1, R(t_1, t_1)) \Rightarrow$ $Happens(ic\_mailReview, t_2, R(t_1, t_1 + 17.05))$
$Happens(ic\_startWoSBS, t_1, R(t_1, t_1)) \wedge$ $Happens(ic\_mailReview, t_2, R(t_1, t_2)) \Rightarrow$ $Happens(ic\_mailComposer, t_3, R(t_2, t_2 + 12.01))$
$Happens(ic\_startWoSBS, t_1, R(t_1, t_1)) \wedge$ $Happens(ic\_mailComposer, t_2, R(t_1, t_2)) \Rightarrow$ $Happens(ic\_mailManagement, t_3, R(t_2, t_2 + 9.01))$

Assumption part
$Happens(ic\_mailReview, t_1, R(t_1, t_1)) \Rightarrow$ $Initiates(ic\_mailReview, mailReview, t_1)$
$Happens(ic\_mailComposer, t_1, R(t_1, t_1)) \Rightarrow$ $Initiates(ic\_mailComposer, mailComposer, t_1)$
$Happens(ic\_mailManagement, t_1, R(t_1, t_1)) \Rightarrow$ $Initiates(ic\_mailManagement, mailManagement, t_1)$

Table 4.10: Instantiation of Role Rule Pattern 2

As an example, consider the part of the SBS related to the *role* of a user depicted in Figure 4.1. Assume a user in the *role* of a "personal user". Applying the role pattern depicted in Table 4.9 results in the pattern instantiation depicted in Table 4.10. As shown in the table, a monitor rule is generated for each operation related to the *role* of the user. More specifically, the initial event, *ic_startWoSBS*, must be followed by the invocation of a single operation (*ic_mailReview*, *ic_mailComposer*, and *ic_mailManagement*) in a specified time constraint. In the case of *ic_mailReview* there is no operation related to the *role* of the user occurring before it, hence the predicate in the body of the rule concerned with the occurrence of a previous invocation is excluded (see first monitor rule in Table 4.10). In the case of *ic_mailComposer* and *ic_mailManagement* the invocations are preceded by the occurrence of previous operations, *mailReview* and *mailComposer* respectively (see second and third monitor rules in Table 4.10). All the time constraints consider small time delays. The time constraints consider all the previous operations in the case of *mailReview*, and only the previous operation in the case of *mailComposer* and *mailManagement*.

Role Rule Pattern 3

The *role* pattern in Table 4.11 concerns with the invocation of those operations related to the *role* of a user and that those operations related to different *roles* are not invoked.

In the pattern, invocations of operations related to the *role* of the user are represented by events of the type $ic\_Event_{R(i)}$. Invocations of operations related to *roles* differing from the one specified for the user are represented by events of the type $ic\_Event_{NR(j)}$. Note that in the case in which an operation appears in the part of the system related to the *role* of the user and in another part related to a different user *role*, the operation is considered as related to the *role* of the user.

Rule part
$\boldsymbol{Happens}(\boldsymbol{ic}\_InitialEvent, t_1, R(t_1, t_1)) \Rightarrow$
$\boldsymbol{Happens}(\boldsymbol{ic}\_Event_{R(1)}, t_2, R(t_1, t_1 + t_{var(1)})) \wedge ... \wedge$
$\boldsymbol{Happens}(\boldsymbol{ic}\_Event_{R(i)}, t_i, R(t_1, t_1 + t_{var(i)})$
$\wedge ... \wedge$
$\neg\boldsymbol{Happens}(\boldsymbol{ic}\_Event_{NR(1)}, t_{i+1}, R(t_1, t_1 + t_{var(i)})) \wedge ... \wedge$
$\neg\boldsymbol{Happens}(\boldsymbol{ic}\_Event_{NR(j)}, t_{i+j}, R(t_1, t_1 + t_{var(i)}))$

Assumption part
$\boldsymbol{Happens}(\boldsymbol{ic}\_Event_{R(1)}, t_1, R(t_1, t_1)) \Rightarrow \boldsymbol{Initiates}(\boldsymbol{ic}\_Event_{R(1)}, Event_{R(1)}, t_1)$
...
$\boldsymbol{Happens}(\boldsymbol{ic}\_Event_{R(i)}, t_1, R(t_1, t_1)) \Rightarrow \boldsymbol{Initiates}(\boldsymbol{ic}\_Event_{R(i)}, Event_{R(i)}, t_1)$

Table 4.11: Role Rule Pattern 3

The rule part of the pattern in Table 4.11 verifies the invocations of a set of operations, related to the *role* of a user ($\mathit{ic\_Event}_{R(i)}$). The pattern also verifies that during the expected time for the execution of all the operations related to the *role* of the user, no invocations concerning operations related to different user *roles* ($\mathit{ic\_Event}_{NR(i)}$) occur. The time constraints $t_{var(i)}$ are computed as the sum of the maximum execution times of those service operations preceding the operation of interest in the part of the SBS related to the *role* of the user. The computation of the time constraints considers small time delays (ten milliseconds) between consecutive events. The assumption part of the *role* pattern in Table 4.11 states that each request of an operation associated to the *role* of the user ($\mathit{ic\_Event}_{R(i)}$) triggers the initialisation of the corresponding $\mathit{Event}_{R(i)}$, which represents the active state of the operation. The invariant part for the *role* context type pattern in Table 4.11 is depicted in bold.

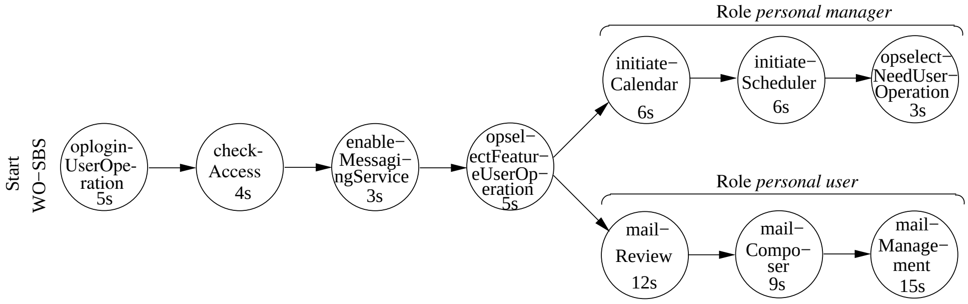


Figure 4.2: Service Specification for Role *personal user* and *personal manager*

As an example, consider the extract of the part of the Wo-SBS application depicted in Figure 4.2 where two different *roles*, “personal user” and “personal manager”, have been specified for two different branches. Assume a user in the *role* of a “personal user”. Applying the pattern depicted in Table 4.11 results in the pattern instantiation depicted in Table 4.12.

As shown in Table 4.12, after the occurrence of the *initial event* (*ic_startWoSBS*) a set of invocations related to the operations associated to the *role* of the user (*ic_mailReview*, *ic_mailComposer*, *ic_mailManagement*), should occur within the expected

Rule part
$Happens(ic\_startWoSBS, t_1, R(t_1, t_1)) \Rightarrow$ $Happens(ic\_mailReview, t_2, R(t_1, t_1 + 17.05)) \wedge$ $Happens(ic\_mailComposer, t_3, R(t_1, t_1 + 23.06)) \wedge$ $Happens(ic\_mailManagement, t_4, R(t_1, t_1 + 29.07)) \wedge$ $\neg Happens(ic\_initiateCalendar, t_5, R(t_1, t_1 + 29.07)) \wedge$ $\neg Happens(ic\_initiateScheduler, t_6, R(t_1, t_1 + 29.07)) \wedge$ $\neg Happens(ic\_opselectNeedUserOperation, t_7, R(t_1, t_1 + 29.07))$

Assumption part
$Happens(ic\_mailReview, t_1, R(t_1, t_1)) \Rightarrow$ $Initiates(ic\_mailReview, mailReview, t_1)$
$Happens(ic\_mailComposer, t_1, R(t_1, t_1)) \Rightarrow$ $Initiates(ic\_mailComposer, mailComposer, t_1)$
$Happens(ic\_mailManagement, t_1, R(t_1, t_1)) \Rightarrow$ $Initiates(ic\_mailManagement, mailManagement, t_1)$

Table 4.12: Instantiation of Role Rule Pattern 3

period of time. In addition, during the invocation of the former operations, no invocation should occur regarding the operations related to *role "personal manager"* (*ic_initiateCalendar*, *ic_initiateScheduler*, *ic_opselectNeedUserOperation*).

Role Rule Pattern 4

The *role* pattern in Table 4.13 verifies the invocation of those operations related to the *role* of a user by using even calculus predicates related to the states of the operations. More specifically, the pattern concerns with the states of those operations related to the *role* of the user (which initially do not hold) and verifies their states after periods of time corresponding to the time constraints in which the operations should have been invoked. Note that the state of an operation is active (i.e. it holds) only if it has been previously invoked.

In the pattern, the initial states of those operations related to the *role* of the user (represented by predicate $Initially_N(Event_i)$ in the body of the rule part of the pat-

Rule part
$\boldsymbol{Initially_N}(Event_1)$
$\wedge ... \wedge$
$\boldsymbol{Initially_N}(Event_i) \wedge$
$\boldsymbol{Happens}(\boldsymbol{ic}\_InitialEvent, t_1, R(t_1, t_1)) \Rightarrow$
$\boldsymbol{Declipped}(t_1, Event_1, t_1 + t_{var(1)}) \ \wedge \ \boldsymbol{Happens}(\boldsymbol{ic}\_Event_1, t_2, R(t_1, t_1 + t_{Max}))$
$\wedge ... \wedge$
$\boldsymbol{Declipped}(t_1, Event_i, t_1 + t_{var(i)}) \ \wedge \ \boldsymbol{Happens}(\boldsymbol{ic}\_Event_i, t_{i+1}, R(t_1, t_1 + t_{Max})) \wedge$
$t_2 < ... < t_{i+1}$

Assumption part
$\boldsymbol{Happens}(\boldsymbol{ic}\_Event_1, t_1, R(t_1, t_1)) \Rightarrow \boldsymbol{Initiates}(\boldsymbol{ic}\_Event_1, Event_1, t_1)$
...
$\boldsymbol{Happens}(\boldsymbol{ic}\_Event_i, t_1, R(t_1, t_1)) \Rightarrow \boldsymbol{Initiates}(\boldsymbol{ic}\_Event_i, Event_i, t_1)$

Table 4.13: Role Rule Pattern 4

tern) have not been triggered. The *initial event* ($ic\_InitialEvent$) represents the first event in a service composition at a time t_1. In the head of the pattern, the predicate $Declipped(t_1, Event_i, t_{var(i)})$ implies an operation is initiated (i.e. it has been invoked) within a time constraint defined by t_1 and $t_{var(i)}$. The predicate $Happens(ic\_Event_i, t_{i+1}, R(t_1, t_{Max}))$ represents the invocation of an operation related to the *role* of the user. Note that there is no specific time constraint for the invocation of an operation, in fact all the invocations related to *Happens* predicates are constrained by t_1 and t_{Max}. The specific time constraint for the occurrence of an invocation is specified by the predicate *Declipped* (within t_1 and $t_{var(i)}$). The time variable $t_{var(i)}$ corresponds to the maximum amount of expected time for the invocation of an operation (ic_Event_i). The time variable t_{Max} corresponds to the maximum amount of time for the invocation of the last operation related to the *role* of the user. All time variables consider small delays between consecutive events (ten milliseconds). The order for the occurrence of the different invocations is given by the last time constraint in the head of the rule part of the pattern ($t_2 < ... < t_{i+1}$).

The assumption part of the *role* pattern in Table 4.13 states that each request of an operation associated to the *role* of the user (ic_Event_i) triggers the initialisation of the corresponding $Event_i$, which represents the active state of the operation. The invariant part for the *role* context type pattern in Table 4.13 is depicted in bold.

As an example, consider the extract of the Wo-SBS depicted in Figure 4.2. Assume a user in the *role* of a “personal user”. Applying the pattern depicted in Table 4.13 results in the pattern instantiation depicted in Table 4.14.

Rule part
$Initially_N(mailReview) \land$ $Initially_N(mailComposer) \land$ $Initially_N(mailManagement) \land$ $Happens(ic\_startWoSBS, t_1, R(t_1, t_1)) \Rightarrow$ $Declipped(t_1, mailReview, 17.05) \land$ $Happens(ic\_mailReview, t_2, R(t_1, t_1 + 38.07)) \land$ $Declipped(t_1, mailComposer, 29.06) \land$ $Happens(ic\_mailComposer, t_3, R(t_1, t_1 + 38.07)) \land$ $Declipped(t_1, mailManagement, 38.07) \land$ $Happens(ic\_mailManagement, t_4, R(t_1, t_1 + 38.07)) \land$ $t_2 < t_3 < t_4$

Assumption part
$Happens(ic\_mailReview, t_1, R(t_1, t_1)) \Rightarrow$ $Initiates(ic\_mailReview, mailReview, t_1)$
$Happens(ic\_mailComposer, t_1, R(t_1, t_1)) \Rightarrow$ $Initiates(ic\_mailComposer, mailComposer, t_1)$
$Happens(ic\_mailManagement, t_1, R(t_1, t_1)) \Rightarrow$ $Initiates(ic\_mailManagement, mailManagement, t_1)$

Table 4.14: Instantiation of Role Rule Pattern 4

As shown in Table 4.14, the initial state of the fluents related to operations *mailReview*, *mailComposer*, and *mailManagement* is inactive, i.e. they do not hold. After the occurrence of the *initial event* (*ic_startWoSBS*) a set of invocations related to the operations associated to the *role* of the user (*ic_mailReview*, *ic_mailComposer*, *ic_mailManagement*), should occur within a time constraint of 38.07 seconds corresponding to the total maximum amount of time for the execution of the last operation related to the *role* of the user (*mailManagement*). The time constraints for the invocation of the different operations is given by the predicates *Declipped*. These predicates specify the states associated to the operations should hold (i.e. must have been invoked) within specific time constraints: 17.05 seconds for *mailReview*, 29.06 seconds for *mailComposer*, and 38.07 seconds for *mailManagement*. Finally the order in which the invocations should occur is given by the time constraint $t_2 < t_3 < t_4$.

Role Rule Pattern 5

The *role* pattern in Table 4.15 is a variation of the previous pattern. It verifies the invocation of those operations related to the *role* of a user by using even calculus predicates related to the states of the operations. Like in the previous pattern, it is concerned with the state of the operations related to the role of the user, which do not hold initially. It also verified the state of the operations after certain period of times represented by the time constraints in which the operations should have been invoked.

In the rule part of the pattern, the initial states of those operations related to the *role* of the user, represented by predicates of the type $Initially_N(Event_i)$ in the body of the pattern, have not been triggered. The *initial event* (*ic_InitialEvent*) represents the first invocation at a time t_1. In the head, the predicate $HoldsAt(Event_i, t_{i+1})$ implies an operation holds at a time t_{i+1}, this means the operation is invoked at a time t_{i+1} or was previously invoked, i.e. before t_{i+1}. The time variable $t_{var(i)}$ corresponds to the maximum amount of time expected for the execution of an operation ($Event_i$). The time variable t_{Prev} corresponds to sum of the operations occurring before the first operation related to the *role* of the user ($Event_1$). All time variables consider small delays between consecutive events (ten milliseconds). The assumption part of the *role* pattern in Table 4.15 states that each request of an operation associated to

Rule part
$\boldsymbol{Initially}_N(Event_1)$
$\wedge ... \wedge$
$\boldsymbol{Initially}_N(Event_i) \wedge$
$\boldsymbol{Happens}(\boldsymbol{ic}\_InitialEvent, t_1, R(t_1, t_1)) \Rightarrow$
$\boldsymbol{HoldsAt}(Event_1, t_2)$
$\wedge ... \wedge$
$\boldsymbol{HoldsAt}(Event_i, t_{i+1}) \wedge$
$t_1 < t_2 < t_1 + t_{Prev} \wedge$
$t_2 < t_3 < t_2 + t_{var(2)} \wedge$
$\wedge ... \wedge$
$t_i < t_{i+1} < t_i + t_{var(i)} \wedge$

Assumption part
$\boldsymbol{Happens}(\boldsymbol{ic}\_Event_1, t_1, R(t_1, t_1)) \Rightarrow \boldsymbol{Initiates}(\boldsymbol{ic}\_Event_1, Event_1, t_1)$
...
$\boldsymbol{Happens}(\boldsymbol{ic}\_Event_i, t_1, R(t_1, t_1)) \Rightarrow \boldsymbol{Initiates}(\boldsymbol{ic}\_Event_i, Event_i, t_1)$

Table 4.15: Role Rule Pattern 5

the *role* of the user (ic_Event_i) triggers the initialisation of the corresponding $Event_i$, which represents the active state of the operation. The invariant part for the *role* context type pattern in Table 4.15 is depicted in bold.

As an example, consider the extract of the Wo-SBS depicted in Figure 4.2. Assume a user in the *role* of a "personal user". Applying the pattern depicted in Table 4.15 results in the pattern instantiation depicted in Table 4.16.

As shown in Table 4.16, the initial state of the fluents related to operations *mailReview*, *mailComposer*, and *mailManagement* is inactive. After the occurrence of the *initial event ic_startWoSBS*, the set of states related to the previous operations, should hold at times t_2, t_3, and t_4. The time variables t_2, t_3, t_4 are constrained by the the sum of the previous operations, when invoking the first operation related to the *role* of the user (case for the time variable t_2), or by the preceding operation, when the invoked operation is not the first one (case for time variables t_3 and t_4).

Rule part
$Initially_N(mailReview) \wedge Initially_N(mailComposer) \wedge$ $Initially_N(mailManagement) \wedge Happens(ic\_startWoSBS, t_1, R(t_1, t_1)) \Rightarrow$ $HoldsAt(mailReview, t_2) \wedge HoldsAt(mailComposer, t_3) \wedge$ $HoldsAt(mailManagement, t_4) \wedge$ $(t_1 < t_2 < t_1 + 17.05) \wedge (t_2 < t_3 < t_2 + 12.01) \wedge (t_3 < t_4 < t_3 + 9.01)$

Assumption part
$Happens(ic\_mailReview, t_1, R(t_1, t_1)) \Rightarrow$ $Initiates(ic\_mailReview, mailReview, t_1)$
$Happens(ic\_mailComposer, t_1, R(t_1, t_1)) \Rightarrow$ $Initiates(ic\_mailComposer, mailComposer, t_1)$
$Happens(ic\_mailManagement, t_1, R(t_1, t_1)) \Rightarrow$ $Initiates(ic\_mailManagement, mailManagement, t_1)$

Table 4.16: Instantiation of Role Rule Pattern 5

Role Rule Pattern 6

The *role* pattern in Table 4.17 verifies the invocation of those operations related to the *role* of a user by using even calculus predicates related to the states of the operations. More specifically, the pattern concerns with the states of those operations related to the *role* of the user (which initially do not hold) and verifies their states after periods of time corresponding to the time constraints in which the operations should have been invoked. In addition the pattern concerns with the states of those operations related to *roles* differing from the one of the user, i.e. operations that should not be invoked. Note that the state of an operation is active (i.e. it holds) only if it has been previously invoked.

In the pattern, the initial states of those operations related to the different *roles* of the user are represented by predicates of the form $Initially_N(Event_i)$ in the body of the rule part of the pattern. The *initial event* (*ic_InitialEvent*) represents the first event in a service composition at a time t_1. Fluents and events containing the form $Event_{R(i)}$ imply the fluent or event is related to the *role* of the user. Similarly fluents and events

Rule part
$\boldsymbol{Initially_N}(Event_1)$
$\wedge ... \wedge$
$\boldsymbol{Initially_N}(Event_i) \wedge$
$\boldsymbol{Happens}(\boldsymbol{ic\_}InitialEvent, t_1, R(t_1, t_1)) \Rightarrow$
$\boldsymbol{Declipped}(t_1, Event_{R(1)}, t_1 + t_{var(1)}) \wedge \boldsymbol{Happens}(\boldsymbol{ic\_}Event_{R(1)}, t_2, R(t_1, t_1 + t_{Max}))$
$\wedge ... \wedge$
$\boldsymbol{Declipped}(t_1, Event_{R(i)}, t_1 + t_{var(i)}) \wedge \boldsymbol{Happens}(\boldsymbol{ic\_}Event_{R(i)}, t_{i+1}, R(t_1, t_1 + t_{Max}))$
$\wedge \neg\boldsymbol{Declipped}(t_1, Event_{NR(1)}, t_1 + t_{Max})$
$\wedge ... \wedge$
$\neg\boldsymbol{Declipped}(t_1, Event_{NR(j)}, t_1 + t_{Max}) \wedge$
$t_2 < ... < t_{i+1}$

Assumption part
$\boldsymbol{Happens}(\boldsymbol{ic\_}Event_1, t_1, R(t_1, t_1)) \Rightarrow \boldsymbol{Initiates}(\boldsymbol{ic\_}Event_1, Event_1, t_1)$
...
$\boldsymbol{Happens}(\boldsymbol{ic\_}Event_i, t_1, R(t_1, t_1)) \Rightarrow \boldsymbol{Initiates}(\boldsymbol{ic\_}Event_i, Event_i, t_1)$

Table 4.17: Role Rule Pattern 6

containing the form $Event_{NR(j)}$ imply the fluent or event is not related to the *role* of the user. In the head of the pattern the predicate $Declipped(t_1, Event_{R(i)}, t_{var(i)})$ represents an operation, related to the *role* of the user, is initiated (i.e. it has been invoked) within a time constraint defined by t_1 and $t_{var(i)}$. The predicate $Happens(ic\_Event_{R(i)}, t_{i+1}, R(t_1, t_{Max}))$ represents the invocation of an operation related to the *role* of the user at a time t_{i+1}.

Note that there is no specific time constraint for the invocation of an operation, in fact all the invocations related to *Happens* predicates are constrained by t_1 and t_{Max}. The specific time constraint for the occurrence of an invocation is specified by the predicate *Declipped* (within t_1 and $t_{var(i)}$). The time variable $t_{var(i)}$ corresponds to the maximum amount of expected time for the invocation of an operation ($ic\_Event_{R(i)}$). The time variable t_{Max} corresponds to the maximum amount of time for the invocation of the last operation related to the *role* of the user. All time variables consider small delays between consecutive events (ten milliseconds). The order for the occurrence of the different invocations is given by the last time constraint in the head of the rule part of the pattern ($t_2 < ... < t_{i+1}$).

The predicates $\neg Declipped(t_1, Event_{NR(j)}, t_1 + t_{Max})$, in the head of the rule part of the pattern, verify that operations related to *roles* differing from the one specified for the user, are not invoked.

The assumption part of the *role* pattern in Table 4.17 states that each request of an operation associated to the *role* of the user ($ic\_Event_{R(i)}$) triggers the initialisation of the corresponding $Event_{R(i)}$, which represents the active state of the operation. The invariant part for the *role* context type pattern in Table 4.17 is depicted in bold.

As an example, consider the extract of the Wo-SBS depicted in Figure 4.2. Assume a user in the *role* of a “personal user”. Applying the pattern depicted in Table 4.17 results in the pattern instantiation depicted in Table 4.18.

As shown in Table 4.18, the initial states of the fluents related to the operations associated to the two user *roles* (see Figure 4.2) do not hold. After the occurrence of the *initial event* (*ic_startWoSBS*), the set of states associated to the operations related to a “personal user” should hold at times t_2, t_3, and t_4. In the rule predicates of the form $\neg$*Declipped* verify operations related to the *role* of a “personal manager” should not be invoked during the expected time for the execution of operations related to the “personal user”.

Role Rule Pattern 7

Another pattern for a *role* context type is shown in Table 4.19. The pattern is a variation of the pattern previously shown in Table 4.7. The pattern in Table 4.19 considers, in the rule part, the occurrence of an *intermediate event* (ic_Int_{Event}) which represents an event occurring after the *initial event* in the system and before the invocation of those operations associated with the *role* of the user.

Rule part
$Initially_N(initiateCalendar) \wedge Initially_N(initiateScheduler) \wedge$ $Initially_N(opselectNeedUserOperation) \wedge Initially_N(mailReview) \wedge$ $Initially_N(mailComposer) \wedge Initially_N(mailManagement) \wedge$ $Happens(ic\_startWoSBS, t_1, R(t_1, t_1)) \Rightarrow$ $Declipped(t_1, mailReview, t_1 + 17.05) \wedge$ $Happens(ic\_mailReview, t_2, R(t_1, t_1 + 29.07)) \wedge$ $Declipped(t_1, mailComposer, t_1 + 23.06) \wedge$ $Happens(ic\_mailComposer, t_3, R(t_1, t_1 + 29.07)) \wedge$ $Declipped(t_1, mailManagement, t_1 + 29.07) \wedge$ $Happens(ic\_mailManagement, t_4, R(t_1, t_1 + 29.07)) \wedge$ $\neg Declipped(t_1, initiateCalendar, t_1 + 29.07) \wedge$ $\neg Declipped(t_1, initiateScheduler, t_1 + 29.07) \wedge$ $\neg Declipped(t_1, opselectNeedUserOperation, t_1 + 29.07) \wedge$ $t_2 < t_3 < t_4$

Assumption part
$Happens(ic\_mailReview, t_1, R(t_1, t_1)) \Rightarrow$ $Initiates(ic\_mailReview, mailReview, t_1)$
$Happens(ic\_mailComposer, t_1, R(t_1, t_1)) \Rightarrow$ $Initiates(ic\_mailComposer, mailComposer, t_1)$
$Happens(ic\_opselectNeedUserOperation, t_1, R(t_1, t_1)) \Rightarrow$ $Initiates(ic\_opselectNeedUserOperation, opselectNeedUserOperation, t_1)$

Table 4.18: Instantiation of Role Rule Pattern 6

Rule part
$\boldsymbol{Happens}(\boldsymbol{ic}\_Int_{Event}, t_1, R(t_1, t_1)) \Rightarrow$ $\boldsymbol{Happens}(\boldsymbol{ic}\_Event_1, t_2, R(t_1, t_1 + {t_n}^1))$ $\wedge ... \wedge$ $\boldsymbol{Happens}(\boldsymbol{ic}\_Event_i, t_{i+1}, R(t_i, t_1 + {t_n}^i))$

Assumption part
$\boldsymbol{Happens}(ic\_Event_1, t_1, R(t_1, t_1)) \Rightarrow \boldsymbol{Initiates}(\boldsymbol{ic}\_Event_1, Event_1, t_1)$ ... $\boldsymbol{Happens}(ic\_Event_i, t_1, R(t_1, t_1)) \Rightarrow \boldsymbol{Initiates}(\boldsymbol{ic}\_Event_i, Event_i, t_1)$

Table 4.19: Role Rule Pattern 7

The rule part of the *role* pattern in Table 4.19 states that the invocation of the *intermediate event* (ic_Int_{Event}) is followed by the request of a sequence of operations

(ic_Event_1, ..., ic_Event_i) at an interval of time $(t_i, t_1 + t_n{}^i)$. An event ic_Event_k ($1 \leq k \leq i$) corresponds to an invocation of an operation identified in the SBS specification that is associated with the specific *role* of the user.

The value of each time variable $t_n{}^l$ ($1 \leq l \leq i$) is computed as the sum of the maximum execution times of those service operations occurring after the *intermediate event* (ic_Int_{Event}) and before the events of interest in the service specification (ic_Event_i). The time variable $t_n{}^l$ also considers the operation related to the invocation of the *intermediate event* and small time delays between consecutive events (ten milliseconds).

The assumption part of the *role* pattern in Table 4.19 states that each request of an operation ic_Event_i triggers the initialisation of the state $Event_i$, which represents the active state of the operation. The invariant part for the *role* context type pattern in Table 4.19 is depicted in bold.

The rule in Table 4.20 presents the instantiation of the pattern in Table 4.19 for the Wo-SBS scenario considering a user in the *role* "personal user" (see Figure 4.1).

The rule parts in the example in Table 4.20 specify that the request for operations *oploginUserOperation*, *checkAccess*, *enableMessagingService*. and *opselectFeatureUserOperation* (events *ic_oploginUserOperation*, *ic_checkAccess*, *ic_enableMessagingService*, *ic_opselectFeatureUserOperation*) must be followed by the invocation of operations *mailReview*, *mailComposer*, *mailManagement*. The time constraints for each rule in Table 4.20 are computed considering the time between each intermediate event (*ic_oploginUserOperation*, *ic_checkAccess*, *ic_enableMessagingService*, *ic_opselectFeatureUserOperation*) and the invocation of operations *mailReview*, *mailComposer*, and *mailManagement*. See Figure 4.1 for operations sequence and expected time. In the assumption part of the instantiated pattern, the invocation of each operation initiates its corresponding state.

Rule part
$Happens(ic\_oploginUserOperation, t_1, R(t_1, t_1)) \Rightarrow$ $Happens(ic\_mailReview, t_2, R(t_1, t_1 + 17.04)) \wedge$ $Happens(ic\_mailComposer, t_3, R(t_2, t_1 + 29.05)) \wedge$ $Happens(ic\_mailManagement, t_4, R(t_3, t_1 + 38.06))$
$Happens(ic\_checkAccess, t_1, R(t_1, t_1)) \Rightarrow$ $Happens(ic\_mailReview, t_2, R(t_1, t_1 + 12.03)) \wedge$ $Happens(ic\_mailComposer, t_3, R(t_2, t_1 + 24.04)) \wedge$ $Happens(ic\_mailManagement, t_4, R(t_3, t_1 + 33.05))$
$Happens(ic\_enableMessagingService, t_1, R(t_1, t_1)) \Rightarrow$ $Happens(ic\_mailReview, t_2, R(t_1, t_1 + 8.02)) \wedge$ $Happens(ic\_mailComposer, t_3, R(t_2, t_1 + 20.03)) \wedge$ $Happens(ic\_mailManagement, t_4, R(t_3, t_1 + 29.04))$
$Happens(ic\_opselectFeatureUserOperation, t_1, R(t_1, t_1)) \Rightarrow$ $Happens(ic\_mailReview, t_2, R(t_1, t_1 + 5.01)) \wedge$ $Happens(ic\_mailComposer, t_3, R(t_2, t_1 + 17.02)) \wedge$ $Happens(ic\_mailManagement, t_4, R(t_3, t_1 + 26.03))$

Assumption part
$Happens(ic\_mailReview, t_1, R(t_1, t_1)) \Rightarrow$ $Initiates(ic\_mailReview, mailReview, t1_1)$
$Happens(ic\_mailComposer, t_1, R(t_1, t_1)) \Rightarrow$ $Initiates(ic\_mailComposer, mailComposer, t_1)$
$Happens(ic\_mailManagement, t_1, R(t_1, t_1)) \Rightarrow$ $Initiates(ic\_mailManagement, mailManagement, t_1)$

Table 4.20: Instantiation of Role Rule Pattern 7

4.2.4 Skills Patterns

Patterns for *skills* context types have been created based on the assumption that users interact with a system in different ways, depending on their *skills*. More specifically, an operation requiring an interaction with a user is executed if the operation is according to the level of *skills* of a user.

In our work we assume a *user operation* can be differentiated from an operation that does not involve the participation of a user, based on the syntax of the name of the operation. More specifically, we assume the name of a *user operation* possesses a prefix, corresponding to the characters “op”, followed by a variable number of characters and a suffix, corresponding to the string “UserOperation”. Although there are some proposals addressing user tasks and user interaction in BPEL compositions, e.g. [130][180], there is no agreed standard for the user participation in a BPEL process. Furthermore, our proposal is not only quite simple but, if necessary, can be easily modified.

Skills Rule Pattern 1

A pattern for the *skills* context type is depicted in Table 4.21. The rule part of the *skills* pattern in Table 4.21 states that between the invocation of the initial event (*ic_InitialEvent*) at a time t_1, and last response in the system (*ir_LastResponse*) at a time t_2, a set of *user operations*, i.e. operations with an “op” prefix and a “UserOperation” suffix should occur. The time variable t_n is computed as the sum of the maximum execution times of all the involved operations between the initial and last event and considers small time delays between consecutive events of ten milliseconds. The assumption part of the *skills* pattern in Table 4.21 states that each request of a *user operation*, $ic\_opNN_iUserOperation$, triggers the initialisation of the corresponding state, $opNN_iUserOperation$, which represents the active state of the operation.

In the case in which there are two or more *user operations* in the part of the SBS related to the level of *skills* of the user, the predicate in the head of the rule part of the pattern is repeated for each one of the operations. Assumptions are instantiated for each *user operation* identified in the part of the SBS related to the user level of *skills*.

The extract of the part of the Wo-SBS application associated with an “average” level of *skills*, together with the operations to be invoked and their order, is shown in Figure 4.3. As shown in Figure 4.3, the sequence of operations that should be

Rule part
$\boldsymbol{Happens}(\boldsymbol{ic}\_InitialEvent, t_1, R(t_1, t_1)) \wedge$ $\boldsymbol{Happens}(\boldsymbol{ir}\_LastResponse, t_2, R(t_1, t_1 + t_n)) \Rightarrow$ $\boldsymbol{Happens}(\boldsymbol{ic\_op}NN_1\boldsymbol{UserOperation}, t_3, R(t_1, t_2))$ $\wedge ... \wedge$ $\boldsymbol{Happens}(\boldsymbol{ic\_op}NN_i\boldsymbol{UserOperation}, t_{i+2}, R(t_1, t_2)) \wedge$ $t_{i+1} < t_{i+2}$

Assumption part
$\boldsymbol{Happens}(\boldsymbol{ic\_op}NN_i\boldsymbol{UserOperation}, t_1, R(t_1, t_1)) \Rightarrow$ $\boldsymbol{Initiates}(\boldsymbol{ic\_op}NN_i\boldsymbol{UserOperation}, \boldsymbol{op}NN_i\boldsymbol{UserOperation}, t_1)$ ... $\boldsymbol{Happens}(\boldsymbol{ic\_op}NN_i\boldsymbol{UserOperation}, t_1, R(t_1, t_1)) \Rightarrow$ $\boldsymbol{Initiates}(\boldsymbol{ic\_op}NN_i\boldsymbol{UserOperation}, \boldsymbol{op}NN_i\boldsymbol{UserOperation}, t_1)$

Table 4.21: Skills Rule Pattern 1

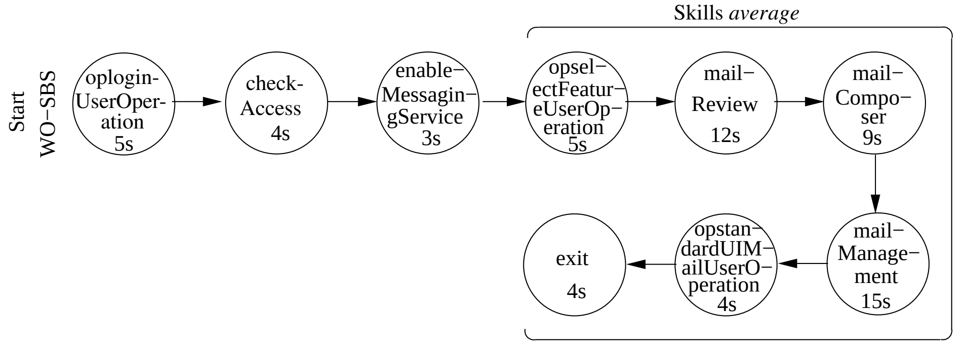

Skills *average*
Start
WO−SBS
oplogin-UserOper-ation 5s
check-Access 4s
enable−Messagin−gService 3s
opsel−ectFeatur−eUserOp−eration 5s
mail−Review 12s
mail−Compo−ser 9s
mail−Manage−ment 15s
opstan−dardUIM−ailUserO−peration 4s
exit 4s

Figure 4.3: Service Specification Sequence for a user with an *average* level of Skills

executed for a user with an "average" level of *skills* are: *opselectFeatureUserOperation*, *mailReview*, *mailComposer*, *mailManagement*, *opstandardUIMailUserOperation*, and *exit*. In this case operations *opselectFeatureUserOperation* and *opstandardUIMailUserOperation* correspond to the *user operations* related to an "average" level of *skills* of a user (note the "op" prefix and "UserOperation" suffix in the name of the operations).

Table 4.22 presents the instantiation of the pattern depicted in Table 4.21 for a user with an "average" level of *skills* for the previous specification in Figure 4.3.

Rule part
$Happens(ic\_startWoSBS, t_1, R(t_1, t_1)) \wedge$ $Happens(ir\_exit, t_2, R(t_1, t_1 + 61.09)) \Rightarrow$ $Happens(ic\_opselectFeatureUserOperation, t_3, R(t_1, t_2)) \wedge$ $Happens(ic\_opstandardUIMailUserOperation, t_4, R(t_1, t_2)) \wedge$ $t_3 < t_4$

Assumption part
$Happens(ic\_opselectFeatureUserOperation, t_1, R(t_1, t_1)) \Rightarrow$ $Initiates(ic\_opselectFeatureUserOperation,$ $opselectFeatureUserOperation, t_1)$
$Happens(ic\_opstandardUIMailUserOperation, t_1, R(t_1, t_1)) \Rightarrow$ $Initiates(ic\_opstandardUIMailUserOperation,$ $opstandardUIMailUserOperation, t_1)$

Table 4.22: Instantiation of Skills Rule Pattern 1

The rule part specifies that the invocation of operations *opselectFeatureUserOperation* and *opstandardUIMailUserOperation* (events *ic_opselectFeatureUserOperation* and *ic_opstandardUIMailUserOperation*) must occur at times t_3 and t_4, after the occurrence of the initial event in the system, *ic_startWoSBS*, at time t_1, and before the response of the last operation (*ir_exit*) at a time t_2. The response of the last operation should occur, at most, 61.09 seconds after the initial event (*ic_startWoSBS*).

Skill Rule Pattern 2

Another pattern for *skills* is shown in Table 4.23. In rule part of the pattern, a *user operation*, i.e. an operation with an "op" prefix and a "UserOperation" suffix, should occur, at a time t_2. The invocation of the user operation should occur between the initial invocation in the system (*ic_InitialEvent*), at a time t_1, and the last response (*ir_LastResponse*) at a time t_3. In the pattern, the time variable t_n, representing the time constraint for the occurrence of the last response, is computed as the sum of the maximum execution times of all the involved operations between the initial and last event, and considers the small time delays between consecutive events of ten milliseconds. The assumption part of the *skills* pattern states that each request of a *user operation*, $ic\_opNN_iUserOperation$, triggers the initialisation of the state, $opNN_iUserOperation$, which represents the active state of the *user operation*.

In the case there are two or more *user operations* in the part of the SBS related to the user *skills*, the pattern is repeated for each one of the *user operations*. The invariant part for the *skills* context type pattern in Table 4.23 is depicted in bold.

Rule part
$\boldsymbol{Happens}(\boldsymbol{ic\_}InitialEvent, t_1, R(t_1, t_1)) \wedge$ $\boldsymbol{Happens}(\boldsymbol{ir\_}LastResponse, t_3, R(t_1, t_1 + t_n)) \Rightarrow$ $\boldsymbol{Happens}(\boldsymbol{ic\_opNNUserOperation}, t_2, R(t_1, t_3))$

Assumption part
$\boldsymbol{Happens}(\boldsymbol{ic\_opNNUserOperation}, t_1, R(t_1, t_1)) \Rightarrow$ $\boldsymbol{Initiates}(\boldsymbol{ic\_opNNUserOperation}, \boldsymbol{opNNUserOperation}, t_1)$

Table 4.23: Skills Rule Pattern 2

Rule part
$Happens(ic\_startWoSBS, t_1, R(t_1, t_1)) \wedge$ $Happens(ir\_exit, t_3, R(t_1, t_1 + 61.09)) \Rightarrow$ $Happens(ic\_opselectFeatureUserOperation, t_2, R(t_1, t_3))$
$Happens(ic\_startWoSBS, t_1, R(t_1, t_1)) \wedge$ $Happens(ir\_exit, t_3, R(t_1, t_1 + 61.09)) \Rightarrow$ $Happens(ic\_opstandardUIMailUserOperation, t_2, R(t_1, t_3))$

Assumption part
$Happens(ic\_opselectFeatureUserOperation, t_1, R(t_1, t_1)) \Rightarrow$ $Initiates(ic\_opselectFeatureUserOperation,$ $opselectFeatureUserOperation, t_1)$
$Happens(ic\_opstandardUIMailUserOperation, t_1, R(t_1, t_1)) \Rightarrow$ $Initiates(ic\_opstandardUIMailUserOperation,$ $opstandardUIMailUserOperation, t_1)$

Table 4.24: Instantiation of Skills Rule Pattern 2

As an example, consider the part of the SBS related to a user associated to an “average” level of *skills*, as depicted in Figure 4.3. Applying the *skills* pattern depicted in Table 4.23 results in the pattern instantiations depicted in Table 4.24. The pattern instantiations depicted in Table 4.24 consists of two rules and two assumption parts. Each instantiation considers the invocation of a *user operation*; more specifically invocations *ic_opselectFeatureUserOperation* and *ic_opstandardUIMailUserOperation*, which correspond to the *user operations* involved in the part of the Wo-SBS associated to an "average" level of *skills* (Figure 4.3).

In the rule part of the instantiated pattern the invocation of *user operations opselectFeatureUserOperation* and *opstandardUIMailUserOperation* should occur between the first invocation (*ic_startWoSBS*) and last response (*ir_exit*) in the system. In the assumption part of the instantiated pattern, the occurrence of *user operations opselectFeatureUserOperation* and *opstandardUIMailUserOperation* trigger the initialisation of the corresponding states associated to each operation.

Skill Rule Pattern 3

The rule part of the *skills* pattern in Table 4.25 states that after the invocation of the initial event (*ic_InitialEvent*) at a time t_1, a set of *user operations* invocations ($ic\_opNN_iUserOperation$) should occur, followed by the last response in the system (*ir_LastResponse*) at a time t_2. The time variable t_n is computed as the sum of the maximum execution times of all the involved operations between the initial and last event, and considers the small time delays between consecutive events of ten milliseconds. In the assumption part of the pattern, the occurrence of an invocation of a user operation ($ic\_opNN_iUserOperation$) implies the corresponding fluent ($opNN_iUserOperation$) has been initiated, i.e. it holds.

In the case there are two or more *user operations* in the part of the SBS related to the level of *skills*, the predicates in the pattern associated to *user operations* are instantiated for each one of the operations. The invariant part for the *skills* pattern in

Table 4.25 is depicted in bold.

Rule part
$\boldsymbol{Happens}(\boldsymbol{ic\_}InitialEvent, t_1, R(t_1, t_1)) \Rightarrow$ $\boldsymbol{Happens}(\boldsymbol{ic\_op}NN_1\boldsymbol{UserOperation}, t_3, R(t_1, t_2))$ $\wedge ... \wedge$ $\boldsymbol{Happens}(\boldsymbol{ic\_op}NN_i\boldsymbol{UserOperation}, t_{i+2}, R(t_1, t_2)) \wedge$ $\boldsymbol{Happens}(\boldsymbol{ir\_}LastResponse, t_2, R(t_1, t_1 + t_n)) \wedge$ $t_1 < t_3 \ \wedge ... \wedge \ t_i < t_{i+1} \wedge ... \wedge \ t_{i+2} < t_2$

Assumption part
$\boldsymbol{Happens}(\boldsymbol{ic\_op}NN_1\boldsymbol{UserOperation}, t_1, R(t_1, t_1)) \Rightarrow$ $\boldsymbol{Initiates}(\boldsymbol{ic\_op}NN_1\boldsymbol{UserOperation}, \boldsymbol{op}NN_1\boldsymbol{UserOperation}, t_1)$ ... $\boldsymbol{Happens}(\boldsymbol{ic\_op}NN_i\boldsymbol{UserOperation}, t_1, R(t_1, t_1)) \Rightarrow$ $\boldsymbol{Initiates}(\boldsymbol{ic\_op}NN_i\boldsymbol{UserOperation}, \boldsymbol{op}NN_i\boldsymbol{UserOperation}, t_1)$

Table 4.25: Skills Rule Pattern 3

Rule part
$Happens(ic\_startWoSBS, t_1, R(t_1, t_1)) \Rightarrow$ $Happens(ic\_opselectFeatureUserOperation, t_3, R(t_1, t_2)) \wedge$ $Happens(ic\_opstandardUIMailUserOperation, t_4, R(t_1, t_2)) \wedge$ $Happens(ir\_exit, t_2, R(t_1, t_1 + 61.09)) \wedge$ $t_1 < t_3 \wedge t_3 < t_4 \wedge t_4 < t_2$

Assumption part
$Happens(ic\_opselectFeatureUserOperation, t_1, R(t_1, t_1)) \Rightarrow$ $Initiates(ic\_opselectFeatureUserOperation,$ $opselectFeatureUserOperation, t_1)$
$Happens(ic\_opstandardUIMailUserOperation, t_1, R(t_1, t_1)) \Rightarrow$ $Initiates(ic\_opstandardUIMailUserOperation,$ $opstandardUIMailUserOperation, t_1)$

Table 4.26: Instantiation of Skills Rule Pattern 3

As an example, consider a user with an “average” level of *skills* and the extract of the Wo-SBS application depicted in Figure 4.3. Applying the *skills* pattern depicted in Table 4.25 results in the pattern instantiation depicted in Table 4.26. According to the instantiation depicted in Table 4.26 the occurrence of the first event, $ic\_startWoSBS$, should be followed by the invocation of operations *opselectFea-*

tureUserOperation and *opstandardUIMailUserOperation*. The two *user operations* should occur within the first and last event (*ir_exit*) of the Wo-SBS. Furthermore, the occurrence of the last event (*ir_exit*) should occur in no more than 61.09 seconds after the first event (*ic_startWoSBS*). In the assumption part, the invocations of the two *user operations* trigger the initialisation of the corresponding fluents.

Skill Rule Pattern 4

The rule part of the *skills* pattern in Table 4.27 states that after the invocation of the initial event (*ic_InitialEvent*) at a time t_1, the invocation of a *user operation* (*ic_opNNUserOperation*) should occur at a time t_2, followed by the last response in the system (*ir_LastResponse*) at a time t_3. The time variable t_n is computed as the sum of the maximum execution times of all the involved operations between the initial and last response, and considers the small time delays between consecutive events of ten milliseconds. In the assumption part of the pattern, the occurrence of the invocation of the user operation (*ic_opNNUserOperation*) implies the corresponding fluent (*opNNUserOperation*) has been initiated, i.e. it holds.

In the case there are two or more *user operations* in the part of the SBS related to the level of *skills*, the pattern is applied for each one of the *user operations*. The invariant part for the *skills* pattern in Table 4.27 is depicted in bold.

Rule part
$\boldsymbol{Happens}(\boldsymbol{ic\_}InitialEvent, t_1, R(t_1, t_1)) \Rightarrow$ $\boldsymbol{Happens}(\boldsymbol{ic\_opNNUserOperation}, t_2, R(t_1, t_3)) \wedge$ $\boldsymbol{Happens}(\boldsymbol{ir}\_LastResponse, t_3, R(t_1, t_1 + t_n))$

Assumption part
$\boldsymbol{Happens}(\boldsymbol{ic\_op}NN\boldsymbol{UserOperation}, t_1, R(t_1, t_1)) \Rightarrow$ $\boldsymbol{Initiates}(\boldsymbol{ic\_opNNUserOperation}, \boldsymbol{op}NN\boldsymbol{UserOperation}, t_1)$

Table 4.27: Skills Rule Pattern 4

As an example, consider a user with an "average" level of *skills* and the extract of

Rule part
$Happens(ic\_startWoSBS, t_1, R(t_1, t_1)) \Rightarrow$ $Happens(ic\_opselectFeatureUserOperation, t_2, R(t_1, t_3)) \wedge$ $Happens(ir\_exit, t_3, R(t_1, t_1 + 61.09))$
$Happens(ic\_startWoSBS, t_1, R(t_1, t_1)) \Rightarrow$ $Happens(ic\_opstandardUIMailUserOperation, t_2, R(t_1, t_3)) \wedge$ $Happens(ir\_exit, t_3, R(t_1, t_1 + 61.09))$

Assumption part
$Happens(ic\_opselectFeatureUserOperation, t_1, R(t_1, t_1)) \Rightarrow$ $Initiates(ic\_opselectFeatureUserOperation,$ $opselectFeatureUserOperation, t_1)$
$Happens(ic\_opstandardUIMailUserOperation, t_1, R(t_1, t_1)) \Rightarrow$ $Initiates(ic\_opstandardUIMailUserOperation,$ $opstandardUIMailUserOperation, t_1)$

Table 4.28: Instantiation of Skills Rule Pattern 4

the Wo-SBS application depicted in Figure 4.3. Applying the *skills* pattern depicted in Table 4.27 results in the pattern instantiation depicted in Table 4.28. In the table two monitor rules are depicted, one for each *user operation*. The first monitor rule verifies that the occurrence of the first event, *ic_startWoSBS*, should be followed by the invocation of operation *opselectFeatureUserOperation*; this operation should occur before the last event (*ir_exit*) of the Wo-SBS. Similarly the second monitor rule verifies that the occurrence of the first event, *ic_startWoSBS*, should be followed by the invocation of operation *opstandardUIMailUserOperation* and that this operation occurs before the last event (*ir_exit*). For both monitor rules, the occurrence of the last event must be, at most, 61.09 seconds after the first event (*ic_startWoSBS*). In the assumption part, the invocations of the two *user operations* trigger the initialisation of the corresponding fluents.

Skill Rule Pattern 5

The pattern concerns with the states of those *user operations* related to the *skills* of the user, initially not holding, and verifies their states after periods of time corresponding to the time constraints in which the operations should have been invoked. Note that the state of an operation is active, i.e. it holds, only if it has been previously invoked. The rule part of the *skills* pattern in Table 4.29 states that at some point during the invocation of the initial event (*ic_InitialEvent*) at a time t_1, and the last response in the system (*ir_LastResponse*) at a time t_2, a set of states corresponding to the *user operations* should hold. The time variable t_n is computed as the sum of the maximum execution times of all the involved operations between the initial and last response, and considers the small time delays between consecutive events of ten milliseconds. In the assumption part of the pattern, the occurrence of the invocation of the user operation ($ic\_opNN_iUserOperation$) implies the corresponding fluent is initiated, i.e. it holds.

In the case there are two or more *user operations* in the part of the SBS related to the level of *skills*, the predicates related to the *user operations* in the pattern is repeated for each operation. The invariant part for the *skills* pattern in Table 4.29 is depicted in bold.

As an example, consider a user with an "average" level of *skills* and the extract of the Wo-SBS application depicted in Figure 4.3. Applying the *skills* pattern depicted in Table 4.29 results in the pattern instantiation depicted in Table 4.30. In the table the monitor rule verifies that the fluents related to the *user operations opselectFeatureUserOperation* and *opstandardUIMailUserOperation* hold after the occurrence of the first event ($ic\_startWoSBS$) and before the occurrence of the last event (*ir_exit*) of the Wo-SBS. Note that in the body of the rule, the fluents related to the *user operations*, do not hold from the beginning. The occurrence of the last event must be, at

Rule part
$\boldsymbol{Initially_N}(\boldsymbol{opNN_1UserOperation})$ $\wedge ... \wedge$ $\boldsymbol{Initially_N}(\mathbf{op}NN_i\mathbf{UserOperation}) \wedge$ $\boldsymbol{Happens}(\boldsymbol{ic}\_InitialEvent, t_1, R(t_1, t_1)) \Rightarrow$ $\boldsymbol{Happens}(\boldsymbol{ic\_opNN_1UserOperation}, t_3, R(t_1, t_2))$ $\wedge ... \wedge$ $\boldsymbol{Happens}(\boldsymbol{ic\_opNN_iUserOperation}, t_{i+2}, R(t_1, t_2)) \wedge$ $\boldsymbol{Happens}(\boldsymbol{ir}\_LastResponse, t_2, R(t_1, t_1 + t_n))$

Assumption part
$\boldsymbol{Happens}(\boldsymbol{ic\_opNN_1UserOperation}, t_1, R(t_1, t_1)) \Rightarrow$ $\boldsymbol{Initiates}(\boldsymbol{ic\_opNN_1UserOperation}, \boldsymbol{opNN_1UserOperation}, t_1)$ ... $\boldsymbol{Happens}(\boldsymbol{ic\_opNN_iUserOperation}, t_1, R(t_1, t_1)) \Rightarrow$ $\boldsymbol{Initiates}(\boldsymbol{ic\_opNN_iUserOperation}, \boldsymbol{opNN_iUserOperation}, t_1)$

Table 4.29: Skills Rule Pattern 5

Rule part
$Initially_N(opselectFeatureUserOperation) \wedge$ $Initially_N(opstandardUIMailUserOperation) \wedge$ $Happens(ic\_startWoSBS, t_1, R(t_1, t_1)) \Rightarrow$ $Happens(ic\_opselectFeatureUserOperation, t_3, R(t_1, t_2)) \wedge$ $Happens(ic\_opstandardUIMailUserOperation, t_4, R(t_1, t_2)) \wedge$ $Happens(ir\_exit, t_2, R(t_1, t_1 + 61.09))$

Assumption part
$Happens(ic\_opselectFeatureUserOperation, t_1, R(t_1, t_1)) \Rightarrow$ $Initiates(ic\_opselectFeatureUserOperation, opselectFeatureUserOperation, t_1)$
$Happens(ic\_opstandardUIMailUserOperation, t_1, R(t_1, t_1)) \Rightarrow$ $Initiates(ic\_opstandardUIMailUserOperation,$ $opstandardUIMailUserOperation, t_1)$

Table 4.30: Instantiation of Skills Rule Pattern 5

most, 61.09 seconds after the occurrence of the first event ($ic\_startWoSBS$). In the assumption part, the invocations of the two *user operations* trigger the initialisation of the corresponding fluents.

Skill Rule Pattern 6

The pattern in Table 4.31 also concerns with the states of those *user operations* related to the *skills* of the user. It verifies *user operation* states, initially not holding, are active after periods of time corresponding to the time constraints in which the operations should have been invoked. The pattern relies on the use of predicates of the type $Declipped(t_1, Event_i, t_2)$, where $Event_i$ represents a *user operation* related to the level of *skills* of the user. A *user operation* is initiated (i.e. it is invoked) within a time constraint defined by the initial invocation and last response in the system. More specifically, the pattern assumes the occurrence of an initial invocation (*ic_InitialEvent*) at a time t_1 and the last response in the system (*ir_LastResponse*) at a time t_2. All *user operations* related to the level of *skills* of the user, should occur within the initial invocation and last response. The occurrence of the last response is computed as the sum of the maximum execution times of all the involved operations between the initial invocation and the last response, and considers the small time delays between consecutive events of ten milliseconds.

In the case there are two or more *user operations* in the part of the SBS related to the level of *skills*, the predicates related to the *user operations* in the pattern are repeated for each operation. The invariant part for the *skills* pattern in Table 4.31 is depicted in bold. In the assumption part of the pattern, the occurrence of the invocation of a *user operation* ($ic\_opNN_iUserOperation$) implies the corresponding fluent is initiated, i.e. it holds.

As an example, consider a user with an "average" level of *skills* and the extract of the Wo-SBS application depicted in Figure 4.3. Applying the *skills* pattern depicted in Table 4.31 results in the pattern instantiation depicted in Table 4.32. In the table the monitor rule verifies that the fluents related to the *user operations opselectFeatureUserOperation* and *opstandardUIMailUserOperation* are initiated after

Rule part
$\boldsymbol{Initially_N(opNN_1UserOperation)}$ $\wedge ... \wedge$ $\boldsymbol{Initially_N}(\mathbf{op}NN_i\mathbf{UserOperation}) \wedge$ $\boldsymbol{Happens(ic}\_InitialEvent, t_1, R(t_1, t_1)) \Rightarrow$ $\boldsymbol{Declipped}(t_1, \boldsymbol{ic\_opNN_1UserOperation}, t_2)$ $\wedge ... \wedge$ $\boldsymbol{Declipped}(t_1, \boldsymbol{ic\_opNN_iUserOperation}, t_2) \wedge$ $\boldsymbol{Happens(ir}\_LastResponse, t_2, R(t_1, t_1 + t_n))$

Assumption part
$\boldsymbol{Happens(ic\_opNN_1UserOperation}, t_1, R(t_1, t_1)) \Rightarrow$ $\boldsymbol{Initiates(ic\_opNN_1UserOperation, opNN_1UserOperation}, t_1)$ ... $\boldsymbol{Happens(ic\_opNN_iUserOperation}, t_1, R(t_1, t_1)) \Rightarrow$ $\boldsymbol{Initiates(ic\_opNN_iUserOperation, opNN_iUserOperation}, t_1)$

Table 4.31: Skills Rule Pattern 6

Rule part
$Initially_N(opselectFeatureUserOperation) \wedge$ $Initially_N(opstandardUIMailUserOperation) \wedge$ $Happens(ic\_startWoSBS, t_1, R(t_1, t_1)) \Rightarrow$ $Declipped(t_1, opselectFeatureUserOperation, t_2) \wedge$ $Declipped(t_1, opstandardUIMailUserOperation, t_2) \wedge$ $Happens(ir\_exit, t_2, R(t_1, t_1 + 61.09))$

Assumption part
$Happens(ic\_opselectFeatureUserOperation, t_1, R(t_1, t_1)) \Rightarrow$ $Initiates(ic\_opselectFeatureUserOperation, opselectFeatureUserOperation, t_1)$
$Happens(ic\_opstandardUIMailUserOperation, t_1, R(t_1, t_1)) \Rightarrow$ $Initiates(ic\_opstandardUIMailUserOperation,$ $opstandardUIMailUserOperation, t_1)$

Table 4.32: Instantiation of Skills Rule Pattern 6

the occurrence of the first event ($ic\_startWoSBS$) and before the occurrence of the last event (ir_exit) of the Wo-SBS. Note that in the body of the rule, the fluents related to the *user operations*, do not hold from the beginning. The occurrence of the last event must be, at most, 61.09 seconds after the occurrence of the first event

($ic\_startWoSBS$). In the assumption part, the invocations of the two *user operations* trigger the initialisation of the corresponding fluents.

Skills Rule Pattern 7

Another pattern for *skills* context type is shown in Table 4.33. The pattern is a variation of the pattern previously shown in Table 4.27. The pattern in Table 4.33 considers, in the rule part, the occurrence of an *intermediate event* (ic_Int_{Event}) which represents an event occurring after the *initial event* in the system and before the invocation of those *user operations* associated to the level of *skills* of the user.

Rule part
$\boldsymbol{Happens}(\boldsymbol{ic\_}Int_{Event}, t_1, R(t_1, t_1)) \Rightarrow$ $\boldsymbol{Happens}(\boldsymbol{ic\_opNNUserOperation}, t_2, R(t_1, t_3)) \wedge$ $\boldsymbol{Happens}(\boldsymbol{ir\_}LastResponse, t_3, R(t_1, t_1 + t_n))$

Assumption part
$\boldsymbol{Happens}(\boldsymbol{ic\_opNNUserOperation}, t_1, R(t_1, t_1)) \Rightarrow$ $\boldsymbol{Initiates}(\boldsymbol{ic\_opNNUserOperation}, \boldsymbol{opNNUserOperation}, t_1)$

Table 4.33: Skills Rule Pattern 7

The rule part of the *skills* pattern in Table 4.33 states that the invocation of the *intermediate event* (ic_Int_{Event}) at a time t_1 must be followed by the request of a *user operation* at a time t_2 and the occurrence of the last response at a time t_3. The *user operation* is identified in the SBS specification as related to the specific level of *skills* of the user.

The value of the time variable t_n is computed as the sum of the maximum execution times of those service operations occurring after the *intermediate event* (ic_Int_{Event}) and the last response of the system. The time variable t_n also considers the operation related to the invocation of the *intermediate event* and small time delays between consecutive events (ten milliseconds).

The assumption part of the *skills* pattern in Table 4.33 states that a request of an operation triggers the initialisation of a state, that represents the active state of the operation.

If there are several *user operations* related to the level of *skills* of the user, the pattern is applied to each one of the identified operations. The invariant part for the *skills* context type pattern in Table 4.33 is depicted in bold.

The rule in Table 4.34 presents the instantiation of the pattern in Table 4.33 for the extract of the Wo-SBS scenario considering a user with an "average" level of *skills* (see Figure 4.3).

The rule parts in Table 4.34 specify that the invocations of *user operations opselectFeatureUserOperation* and *opstandardUIMailUserOperation* must occur after an intermediate event and before the last response in the system. The time constraint for each monitor rule is computed considering the operation related to the intermediate event and the remaining operations in the service composition (see Figure 4.3) plus small time delays between consecutive events.

4.2.5 Cognition Patterns

There have been several studies dealing with human *cognition* and its importance when a user interacts with a system, e.g. [7][87][128][167][191][237]. The focus of the studies dealing with user *cognition* can range, for example, from the analysis of the type of tasks performed during user interaction, to the composition of the user interface. In our work we focus on a specific aspect concerning *cognition* and *user interaction*: the expected time required by a user to perform a *user operation* in the system. More specifically, we propose *cognition* patterns that focus on the time required for a user to interact with a system.

Rule part
$Happens(ic\_oploginUserOperation, t_1, R(t_1, t_1)) \Rightarrow$ $Happens(ic\_opselectFeatureUserOperation, t_2, R(t_1, t_3)) \wedge$ $Happens(ir\_exit, t_3, R(t_1, t_1 + 12.03)$
$Happens(ic\_checkAccess, t_1, R(t_1, t_1)) \Rightarrow$ $Happens(ic\_opselectFeatureUserOperation, t_2, R(t_1, t_3)) \wedge$ $Happens(ir\_exit, t_3, R(t_1, t_1 + 7.02)$
$Happens(ic\_enableMessagingService, t_1, R(t_1, t_1)) \Rightarrow$ $Happens(ic\_opselectFeatureUserOperation, t_2, R(t_1, t_3)) \wedge$ $Happens(ir\_exit, t_3, R(t_1, t_1 + 3.01)$
$Happens(ic\_oploginUserOperation, t_1, R(t_1, t_1)) \Rightarrow$ $Happens(ic\_opstandardUIMailUserOperation, t_2, R(t_1, t_3)) \wedge$ $Happens(ir\_exit, t_3, R(t_1, t_1 + 53.07)$
$Happens(ic\_checkAccess, t_1, R(t_1, t_1)) \Rightarrow$ $Happens(ic\_opstandardUIMailUserOperation, t_2, R(t_1, t_3)) \wedge$ $Happens(ir\_exit, t_3, R(t_1, t_1 + 48.06)$
$Happens(ic\_enableMessagingService, t_1, R(t_1, t_1)) \Rightarrow$ $Happens(ic\_opstandardUIMailUserOperation, t_2, R(t_1, t_3)) \wedge$ $Happens(ir\_exit, t_3, R(t_1, t_1 + 44.05)$
$Happens(ic\_opselectFeatureUserOperation, t_1, R(t_1, t_1)) \Rightarrow$ $Happens(ic\_opstandardUIMailUserOperation, t_2, R(t_1, t_3)) \wedge$ $Happens(ir\_exit, t_3, R(t_1, t_1 + 41.04)$
$Happens(ic\_mailReview, t_1, R(t_1, t_1)) \Rightarrow$ $Happens(ic\_opstandardUIMailUserOperation, t_2, R(t_1, t_3)) \wedge$ $Happens(ir\_exit, t_3, R(t_1, t_1 + 36.03)$
$Happens(ic\_mailComposer, t_1, R(t_1, t_1)) \Rightarrow$ $Happens(ic\_opstandardUIMailUserOperation, t_2, R(t_1, t_3)) \wedge$ $Happens(ir\_exit, t_3, R(t_1, t_1 + 24.02)$
$Happens(ic\_mailManagement, t_1, R(t_1, t_1)) \Rightarrow$ $Happens(ic\_opstandardUIMailUserOperation, t_2, R(t_1, t_3)) \wedge$ $Happens(ir\_exit, t_3, R(t_1, t_1 + 15.01)$

Assumption part
$Happens(ic\_opselectFeatureUserOperation, t_1, R(t_1, t_1)) \Rightarrow\ Initiates$ $(ic\_opselectFeatureUserOperation, opselectFeatureUserOperation, t_1)$
$Happens(ic\_opstandardUIMailUserOperation, t_1, R(t_1, t_1)) \Rightarrow\ Initiates$ $(ic\_opstandardUIMailUserOperation, opstandardUIMailUserOperation, t_1)$

Table 4.34: Instantiation of Skills Rule Pattern 7

Different studies have addressed the issues regarding *cognition* and user response times, e.g. [117][118][154][187], their focus however, is on particular actions/tasks (e.g. keystroke-level models), rather than user *cognition* in a broad sense. In our work we propose a general classification, consisting of three general levels, for the user *cognition*. Our classification is based on what has been proposed in previous studies regarding user *cognition*, see [57][138][197]. The classification considers three categories *slow*, *average*, and *fast*, where each category represents the expected amount of time a user requires to interact with a system, according to his/her level of *cognition*. More specifically, for each category a percentage represents the fraction of the expected amount of time required to accomplish a *user operation*. Table 4.35 specifies the percentages according to the level of *cognition* of the user.

Cognition Level	Expected utilisation of the specified time
Low	100%
Average	75%
High	50%

Table 4.35: Expected Time Percentages According to the Cognition Level of the User

In the case of a "low" level of *cognition*, the time to execute a *user operation* should be up to the total amount of time specified for the operation. In the case of an "average" level of *cognition*, the time to execute a *user operation* should be up to 75% of the time expected for the execution of the operation. In the case of "high" level of *cognition*, the time to execute a *user operation* should be up to 50% of the time expected for the execution of the operation. The above percentage ranges are fixed but, if necessary, can be easily modified.

The patterns for *cognition* are concerned with the time it takes for a user to interact with the system. Therefore the patterns for *cognition* focus on the invocations and responses of *user operations*, and the time it takes to perform such actions. In our work it is assumed that the time it takes for a user to perform a *user operation* is proportional to the user's level of *cognition*. This assumption is a simplified interpretation of studies conducted in the field of psychology, dealing with user cognitive levels and processing capabilities, see [4][165][3], where imponderable factors, e.g. user beliefs, may influence the user cognitive process.

Cognition Rule Pattern 1

A pattern for *cognition* is shown in Table 4.36. As shown in the rule part of the pattern, the invocation of a *user operation* (*ic_opNNUserOperation*) must be followed by the response of the same operation, (*ir_opNNUserOperation*) in no more than $t_{Current\_cog}$ units of time. The time variable $t_{Current\_cog}$ represents the maximum amount of time for the execution of a *user operation*. Its computation considers the time specified for the *user operation* multiplied by the percentage associated to the level of *cognition* of the user. In the assumption part of the pattern the fluent $opNNUserOperation$ represents the operation "opNNUserOperation" is active, i.e. has been triggered. The initialisation of the fluent $opNNUserOperation$ is triggered by the invocation of the user operation ($ic\_opNNUserOperation$). The invariant part in the pattern is depicted in bold.

Rule part
$\boldsymbol{Happens}(\boldsymbol{ic\_opNNUserOperation}, t_1, R(t_1, t_1))$ $\Rightarrow$ $\boldsymbol{Happens}(\boldsymbol{ir\_opNNUserOperation}, t_2, R(t_1, t_1 + t_{Current\_cog}))$

Assumption part
Happens$(\boldsymbol{ic\_opNNUserOperation}, t_1, R(t_1, t_1))$ $\Rightarrow$ **Initiates**$(\boldsymbol{ic\_opNNUserOperation}, \boldsymbol{opNNUserOperation}, t_1)$

Table 4.36: Cognition Rule Pattern 1

In the case two or more *user operations* have been identified in the part of the SBS to be executed, the *cognition* pattern in Table 4.36 is instantiated for each one of them.

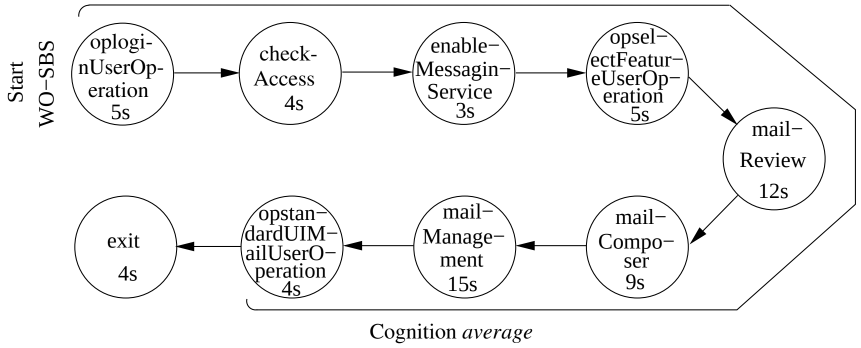

Start
WO−SBS
oplogi-nUserOp-eration 5s
check-Access 4s
enable−Messagin−Service 3s
opsel−ectFeatur−eUserOp−eration 5s
mail−Review 12s
mail−Compo−ser 9s
mail−Manage−ment 15s
opstan−dardUIM−ailUserO−peration 4s
exit 4s
Cognition *average*

Figure 4.4: Service Specification Sequence for an *average* Cognition

As an example, consider the extract of the Wo-SBS scenario depicted in Figure 4.4, assume a user with an "average" level of *cognition* interacting with the system. Applying the *cognition* pattern depicted in Table 4.36 results in the pattern instantiations depicted in Table 4.37.

Rule part
$Happens(ic\_oploginUserOperation, t_1, R(t_1, t_1)) \Rightarrow$ $Happens(ir\_oploginUserOperation, t_2, R(t_1, t_1 + 3.75))$
$Happens(ic\_opselectFeatureUserOperation, t_1, R(t_1, t_1)) \Rightarrow$ $Happens(ir\_opselectFeatureUserOperation, t_2, R(t_1, t_1 + 3.75))$
$Happens(ic\_opstandardUIMailUserOperation, t_1, R(t_1, t_1)) \Rightarrow$ $Happens(ir\_opstandardUIMailUserOperation, t_2, R(t_1, t_1 + 3))$

Assumption part
$Happens(ic\_oploginUserOperation, t_1, R(t_1, t_1)) \Rightarrow$ $Initiates(ic\_oploginUserOperation, oploginUserOperation, t_1)$
$Happens(ic\_opselectFeatureUserOperation, t_1, R(t_1, t_1)) \Rightarrow$ $Initiates(ic\_opselectFeatureUserOperation,$ $opselectFeatureUserOperation, t_1)$
$Happens(ic\_opstandardUIMailUserOperation, t_1, R(t_1, t_1)) \Rightarrow$ $Initiates(ic\_opstandardUIMailUserOperation,$ $opstandardUIMailUserOperation, t_1)$

Table 4.37: Instantiation of Cognition Rule Pattern 1

The pattern instantiations depicted in Table 4.37 consider *user operations oploginUserOpera-tion*, *opselectFeatureUserOperation*, and *opstandardUIMailUserOperation*, which correspond to operations involved in the part of the Wo-SBS associated to an "average" level of *cognition*. In the rule part of the pattern instantiations, the invocation of operation *oploginUserOperation* (event $ic\_oploginUserOperation$) should be followed by the response of the operation (event $ir\_oploginUserOperation$) in no more than 3 seconds. This time constraint represents 75% of the time specified for the execution of this operation ($4*0.75$ seconds) given the "average" level of *cognition* of the user. Similarly, the invocation of operations *opselectFeatureUserOperation* and *opstandardUIMailUserOperation* (events $ic\_opselectFeatureUserOperation$ and $ic\_opstandardUIMailUserOperation$) should be followed by the operation responses (i.e. $ir\_opselectFeatureUserOperation$ and $ir\_opstandardUIMailUserOperation$) in no more than 3.75 seconds ($5*0.75$ seconds for each operation). In the assumption part of the instantiated pattern, the *user operations* are initiated.

Cognition Rule Pattern 2

Another *cognition* pattern is depicted in Table 4.38. As shown in the rule part of the pattern, the invocation of a *user operation* ($ic\_opOperationNN_iUserOperation$) must be followed by its response ($ir\_opOperationNN_iUserOperation$). Each response should occur in no more than $t_{Current\_cog_i}$ units of time. The time variable $t_{Current\_cog_i}$ corresponds to the maximum time specified for the execution of a *user operation*, multiplied by the percentage associated to the respective level of *cognition* of the user.

In the assumption part of the pattern the fluent $opNN_iUserOperation$ represents the operation "opNNiUserOperation" is active. The initialisation of the fluent $opNN_iUserOperation$ is triggered by the invocation of the *user operation* (ic_opNN_i

Rule part
$\boldsymbol{Happens}(\boldsymbol{ic\_opNN_1UserOperation}, t_1, R(t_1, t_1))$
$\wedge ... \wedge$
$\boldsymbol{Happens}(\boldsymbol{ic\_opNN_iUserOperation}, t_i, R(t_i, t_i)) \Rightarrow$
$\boldsymbol{Happens}(\boldsymbol{ir\_opNN_1UserOperation}, t_{i+1}, R(t_1, t_1 + t_{Current\_cog_1}))$
$\wedge ... \wedge$
$\boldsymbol{Happens}(\boldsymbol{ir\_opNN_iUserOperation}, t_j, R(t_i, t_i + t_{Current\_cog_i}))$

Assumption part
Happens$(\boldsymbol{ic\_opNN_1UserOperation}, t_1, R(t_1, t_1)) \Rightarrow$
Initiates$(\boldsymbol{ic\_opNN_1UserOperation}, \boldsymbol{opNN_1UserOperation}, t_1)$
...
Happens$(\boldsymbol{ic\_opNN_iUserOperation}, t_1, R(t_1, t_1)) \Rightarrow$
Initiates$(\boldsymbol{ic\_opNN_iUserOperation}, \boldsymbol{opNN_iUserOperation}, t_1)$

Table 4.38: Cognition Rule Pattern 2

$UserOperation$). The invariant part in the pattern is depicted in bold.

The predicates for invocations and responses in the rule part of the pattern and the initialisation of the operations in the assumption part of the pattern, are instantiated for each*user operation* identified in the part of the SBS to be executed.

As an example, consider the extract of the Wo-SBS depicted in Figure 4.4, where part of the extract is associated to an "average" level of *cognition*. Applying the *cognition* pattern depicted in Table 4.38 results in the pattern instantiation depicted in Table 4.39.

The pattern shown in Table 4.39 depicts operations *oploginUserOperation*, *opselectFeatureUserOperation*, and *opstandardUIMailUserOperat-ion*, which correspond to the *user operations* involved in the part of the Wo-SBS associated to an "average" level of *cognition*. In the instantiation, the invocation of operation *oploginUserOperation* (event $ic\_oploginUserOperation$) should be followed by the response of the operation (event $ir\_oploginUserOperation$) in no more than 3.75 seconds (i.e. $5 * 0.75$ seconds) given the "average" *cognition* of the user. Similarly, the invocation of operation *opselectFeatureUserOperation* (event $ic\_opselectFeature$

Rule part
$Happens(ic\_oploginUserOperation, t_1, R(t_1, t_1)) \wedge$ $Happens(ic\_opselectFeatureUserOperation, t_2, R(t_2, t_2)) \wedge$ $Happens(ic\_opstandardUIMailUserOperation, t_3, R(t_3, t_3)) \Rightarrow$ $Happens(ir\_oploginUserOperation, t_4, R(t_1, t_1 + 3.75)) \wedge$ $Happens(ir\_opselectFeatureUserOperation, t_5, R(t_2, t_2 + 3.75)) \wedge$ $Happens(ir\_opstandardUIMailUserOperation, t_6, R(t_3, t_3 + 3))$

Assumption part
$Happens(ic\_oploginUserOperation, t_1, R(t_1, t_1)) \Rightarrow$ $Initiates(ic\_oploginUserOperation, oploginUserOperation, t_1)$
$Happens(ic\_opselectFeatureUserOperation, t_1, R(t_1, t_1)) \Rightarrow$ $Initiates(ic\_opselectFeatureUserOperation,$ $opselectFeatureUserOperation, t_1)$
$Happens(ic\_opstandardUIMailUserOperation, t_1, R(t_1, t_1)) \Rightarrow$ $Initiates(ic\_opstandardUIMailUserOperation,$ $opstandardUIMailUserOperation, t_1)$

Table 4.39: Instantiation of Cognition Rule Pattern 2

$UserOperation$) should be followed by the response of the operation (event $ir\_opselectFeatureUserOperation$) in no more than 3.75 seconds. Finally, the invocation of operation *opstandardUIMailUserOperation* (event $ic\_opstandardUIMailUserOperation$) must be followed by the response of the operation (event $ir\_opstandardUIMailUserOperation$) in no more than 3 seconds ($4 * 0.75$ seconds).

Cognition Rule Pattern 3

Another *cognition* pattern is depicted in Table 4.40. In the pattern, the response of a *user operation* ($ir\_opNNUserOperation$), must be preceded by the invocation of the *user operation* ($ic\_opNNUserOperation$), in a period of time that does not exceed the execution time specified for the *user operation*. The time constraint is specified by the time variable $t_{Current\_cog}$, which corresponds to the maximum time specified for the execution of the *user operation*, multiplied by the percentage associated to the *cognition* level of the user.

Rule part
$\boldsymbol{Happens}(\boldsymbol{ir\_opNNUserOperation}, t_1, R(t_1, t_1))$ $\Rightarrow$ $\boldsymbol{Happens}(\boldsymbol{ic\_opNNUserOperation}, t_0, R((t_1 - t_{Current\_cog}), t_1))$

Assumption part
$\mathbf{Happens}(\boldsymbol{ic\_opNNUserOperation}, t_1, R(t_1, t_1))$ $\Rightarrow$ $\mathbf{Initiates}(\boldsymbol{ic\_opNNUserOperation}, \boldsymbol{opNNUserOperation}, t_1)$

Table 4.40: Cognition Rule Pattern 3

In the case two or more *user operations* have been identified in the part of the SBS to be executed, the *cognition* pattern in Table 4.40 is instantiated for each one of them.

As an example, consider the part of the SBS related to a user, associated to an "average" level of *cognition*, depicted in Figure 4.4. Applying the *cognition* pattern depicted in Table 4.40 results in the pattern instantiations depicted in Table 4.41.

The pattern instantiations depicted in Table 4.41 consider operations *oploginUserOperation*, *opselectFeatureUserOperation*, and *opstandardUIMailUserOperation*, which correspond to the *user operations* involved in the part of the Wo-SBS associated to an "average" level of *cognition*. In the pattern instantiations, the response of operation *oploginUserOperation* (event *ir_oploginUserOperation*) should have been preceded by the invocation of the operation (event *ic_oploginUserOperation*) within a time range that does not exceed 3.75 seconds. This time range represents 75% of the time specified for the execution of this operation ($5 * 0.75$ seconds) given the "average" *cognition* of the user. Similarly, the responses of operations *opselectFeatureUserOperation* and *opstandardUIMailUserOperation* (events *ir_opselectFeatureUserOperation* and *ir_opstandardUIMailUserOperation*) should be preceded by the invocations of the operations (events *ic_opselectFeatureUserOperation* and *ic_opstan-*

Rule part
$Happens(ir\_oploginUserOperation, t_1, R(t_1, t_1)) \Rightarrow$ $Happens(ic\_oploginUserOperation, t_0, R(t_1 - 3.75, t_1))$
$Happens(ir\_opselectFeatureUserOperation, t_1, R(t_1, t_1)) \Rightarrow$ $Happens(ic\_opselectFeatureUserOperation, t_0, R(t_1 - 3.75, t_1))$
$Happens(ir\_opstandardUIMailUserOperation, t_1, R(t_1, t_1)) \Rightarrow$ $Happens(ic\_opstandardUIMailUserOperation, t_0, R(t_1 - 3, t_1))$

Assumption part
$Happens(ic\_oploginUserOperation, t_1, R(t_1, t_1)) \Rightarrow$ $Initiates(ic\_oploginUserOperation, oploginUserOperation, t_1)$
$Happens(ic\_opselectFeatureUserOperation, t_1, R(t_1, t_1)) \Rightarrow$ $Initiates(ic\_opselectFeatureUserOperation,$ $opselectFeatureUserOperation, t_1)$
$Happens(ic\_opstandardUIMailUserOperation, t_1, R(t_1, t_1)) \Rightarrow$ $Initiates(ic\_opstandardUIMailUserOperation,$ $opstandardUIMailUserOperation, t_1)$

Table 4.41: Instantiation of Cognition Rule Pattern 3

dardUIMailUserOperation) within a time range of 3.75 and 3 seconds ($4 * 0.75$ seconds).

Cognition Rule Pattern 4

Another *cognition* pattern is depicted in Table 4.42. The pattern is similar to the one shown in Table 4.40, however in this pattern all *user operations* are considered in a single monitor rule. More specifically, all operations responses, i.e. $ir\_opNN_iUser-Operation$, must be preceded by the corresponding invocations, i.e. $ic\_opNN_iUser-Operation$, in periods of time that do not exceed the execution time specified for the *user operations*. The time constraints are specified by the time variables $t_{Current\_cog_{(i)}}$, which correspond to the maximum time specified for the execution of *user operations*, multiplied by the percentage associated to the level of *cognition* of the user.

As an example, consider the part of the SBS related to a user, associated to an

Rule part
$\boldsymbol{Happens}(\boldsymbol{ir\_opNN_1UserOperation}, t_1, R(t_1, t_1))$ $\wedge ... \wedge$ $\boldsymbol{Happens}(\boldsymbol{ir\_opNN_iUserOperation}, t_i, R(t_i, t_i))$ $\Rightarrow$ $\boldsymbol{Happens}(\boldsymbol{ic\_opNN_1UserOperation}, t_{i+1}, R((t_1 - t_{Current\_cog_{(1)}}), t_1))$ $\wedge ... \wedge$ $\boldsymbol{Happens}(\boldsymbol{ic\_opNN_iUserOperation}, t_{2i}, R((t_i - t_{Current\_cog_{(i)}}), t_i))$

Assumption part
Happens$(\boldsymbol{ic\_opNN_1UserOperation}, t_1, R(t_1, t_1)) \Rightarrow$ **Initiates**$(\boldsymbol{ic\_opNN_1UserOperation}, \boldsymbol{opNN_1UserOperation}, t_1)$ ... **Happens**$(\boldsymbol{ic\_opNN_iUserOperation}, t_1, R(t_1, t_1)) \Rightarrow$ **Initiates**$(\boldsymbol{ic\_opNN_iUserOperation}, \boldsymbol{opNN_iUserOperation}, t_1)$

Table 4.42: Cognition Rule Pattern 4

"average" level of *cognition*, depicted in Figure 4.4. Applying the *cognition* pattern depicted in Table 4.42 results in the pattern instantiations depicted in Table 4.43.

Rule part
$Happens(ir\_oploginUserOperation, t_1, R(t_1, t_1)) \wedge$ $Happens(ir\_opselectFeatureUserOperation, t_2, R(t_2, t_2)) \wedge$ $Happens(ir\_opstandardUIMailUserOperation, t_3, R(t_3, t_3)) \Rightarrow$ $Happens(ic\_oploginUserOperation, t_4, R(t_1 - 3.75, t_1)) \wedge$ $Happens(ic\_opselectFeatureUserOperation, t_5, R(t_2 - 3.75, t_2)) \wedge$ $Happens(ic\_opstandardUIMailUserOperation, t_6, R(t_3 - 3, t_3))$

Assumption part
$Happens(ic\_oploginUserOperation, t_1, R(t_1, t_1)) \Rightarrow$ $Initiates(ic\_oploginUserOperation, oploginUserOperation, t_1)$
$Happens(ic\_opselectFeatureUserOperation, t_1, R(t_1, t_1)) \Rightarrow$ $Initiates(ic\_opselectFeatureUserOperation,$ $opselectFeatureUserOperation, t_1)$
$Happens(ic\_opstandardUIMailUserOperation, t_1, R(t_1, t_1)) \Rightarrow$ $Initiates(ic\_opstandardUIMailUserOperation,$ $opstandardUIMailUserOperation, t_1)$

Table 4.43: Instantiation of Cognition Rule Pattern 4

The pattern instantiation depicted in Table 4.43 considers the *user operations* *oploginUserOperation*, *opselectFeatureUserOperation*, and *opstandardUIMailUserOperation* which correspond to operations participating in the part of the Wo-SBS associated to an "average" level of *cognition*. In the rule, the response of operation *oploginUserOperation* (event *ir_oploginUserOperation*) should have been preceded by the invocation of the operation (event *ic_oploginUserOperation*) within a time range that does not exceed 3.75 seconds. This time range represents 75% of the time specified for the execution of this operation ($5 * 0.75$ seconds) given the "average" *cognition* of the user. Similarly, the responses of operations *opselectFeatureUserOperation* and *opstandardUIMailUserOperation* (events *ir_opselectFeatureUserOperation* and *ir_opstandardUIMailUserOperation*) should be preceded by the invocations of the operations (events *ic_opselectFeatureUserOperation* and *ic_opstandardUIMailUserOperation*) within a time range of 3.75 and 3 seconds ($4 * 0.75$ seconds).

Cognition Rule Pattern 5

Another pattern for *cognition* is shown in Table 4.44. As shown in the rule part of the pattern, the invocation of an initial event (*ic_InitialEvent*) must be followed by the invocation and response of a *user operation* (events $ic\_opNNUserOperation$ and $ir\_opNNUserOperation$) at times t_2 and t_3. The response must occur in no more than $t_{Current\_cog}$ units of time after the invocation at a time t_2. The time variable $t_{Current\_cog}$ represents the maximum amount of time for the execution of a *user operation*. Its computation considers the time specified for the *user operation* multiplied by the percentage associated to the level of *cognition* of the user. In the assumption part of the pattern the fluent $opNNUserOperation$ represents the operation "opNNUserOperation" is active, i.e. has been triggered. The initialisation of the fluent $opNNUserOperation$ is triggered by the invocation of the user operation ($ic\_opNNUserOperation$). The invariant part in the pattern is depicted in bold.

Rule part
$\boldsymbol{Happens}(\boldsymbol{ic\_}InitialEvent, t_1, R(t_1, t_1)) \Rightarrow$ $\boldsymbol{Happens}(\boldsymbol{ic\_opNNUserOperation}, t_2, R(t_1, t_2)) \wedge$ $\boldsymbol{Happens}(\boldsymbol{ir\_opNNUserOperation}, t_3, R(t_2, t_2 + t_{Current\_cog}))$

Assumption part
$\mathbf{Happens}(\boldsymbol{ic\_opNNUserOperation}, t_1, R(t_1, t_1)) \Rightarrow$ $\mathbf{Initiates}(\boldsymbol{ic\_opNNUserOperation}, \boldsymbol{opNNUserOperation}, t_1)$

Table 4.44: Cognition Rule Pattern 5

In the case two or more *user operations* have been identified in the part of the SBS to be executed, the *cognition* pattern in Table 4.44 is instantiated for each one of them.

As an example, consider the extract of the Wo-SBS scenario depicted in Figure 4.4, assume a user with an "average" level of *cognition* interacting with the system. Applying the *cognition* pattern depicted in Table 4.44 results in the pattern instantiations depicted in Table 4.45.

The pattern instantiations depicted in Table 4.45 consider *user operations oploginUserOperation*, *opselectFeatureUserOperation*, and *opstandardUIMailUserOperation*, which correspond to operations involved in the part of the Wo-SBS associated to an "average" level of *cognition*. In each monitor rule the occurrence of the initial event (*ic_startWoSBS*) must be followed by the invocation of an operation and its response. Thus after the occurrence of the initial event *ic_startWoSBS*, the invocation of operation *oploginUserOperation* (event *ic_oploginUserOperation*) must be followed by the response of the operation (event *ir_oploginUserOperation*) in no more than 3 seconds. This time constraint represents 75% of the time specified for the execution of this operation (4∗0.75 seconds) given the "average" level of *cognition* of the user. Similarly, monitor rules state that after the occurrence of the initial event, invocations *ic_opselectFeatureUserOperation* and *ic_opstandardUIMailUserOperation* should be followed by the operation responses (*ir_opselectFeatureUserOperation*

Rule part
$Happens(ic\_startWoSBS, t_1, R(t_1, t_1)) \Rightarrow$ $Happens(ic\_oploginUserOperation, t_2, R(t_1, t_2)) \wedge$ $Happens(ir\_oploginUserOperation, t_3, R(t_2, t_2 + 3.75))$
$Happens(ic\_startWoSBS, t_1, R(t_1, t_1)) \Rightarrow$ $Happens(ic\_opselectFeatureUserOperation, t_2, R(t_1, t_2)) \wedge$ $Happens(ir\_opselectFeatureUserOperation, t_3, R(t_2, t_2 + 3.75))$
$Happens(ic\_startWoSBS, t_1, R(t_1, t_1)) \Rightarrow$ $Happens(ic\_opstandardUIMailUserOperation, t_2, R(t_1, t_2)) \wedge$ $Happens(ir\_opstandardUIMailUserOperation, t_3, R(t_2, t_2 + 3))$

Assumption part
$Happens(ic\_oploginUserOperation, t_1, R(t_1, t_1)) \Rightarrow$ $Initiates(ic\_oploginUserOperation, oploginUserOperation, t_1)$
$Happens(ic\_opselectFeatureUserOperation, t_1, R(t_1, t_1)) \Rightarrow$ $Initiates(ic\_opselectFeatureUserOperation,$ $opselectFeatureUserOperation, t_1)$
$Happens(ic\_opstandardUIMailUserOperation, t_1, R(t_1, t_1)) \Rightarrow$ $Initiates(ic\_opstandardUIMailUserOperation,$ $opstandardUIMailUserOperation, t_1)$

Table 4.45: Instantiation of Cognition Rule Pattern 5

and $ir\_opstandardUIMailUserOperation$) in no more than 3.75 seconds ($5 * 0.75$ seconds for each operation). In the assumption part of the pattern instantiations the *user operations* are initiated.

Cognition Rule Pattern 6

Table 4.46 shows another pattern for *cognition*. As shown in the rule part of the pattern, the invocation of an initial event (*ic_InitialEvent*) must be followed by the invocations and responses of a set of *user operations* (events $ic\_opNN_iUserOperation$ and $ir\_opNN_iUserOperation$ respectively). Each operation response must occur in no more than $t_{Current\_cog_{(i)}}$ units of time after the invocation of the operation. The time variable $t_{Current\_cog_{(i)}}$ represents the maximum amount of time for the execution of a *user operation*. The computation of the time variable $t_{Current\_cog_{(i)}}$ considers the time specified for a *user operation* multiplied by the percentage associated to the level of *cognition* of the user. In the assumption part of the pattern a fluent represents the associated operation is active, i.e. has been triggered. for example, the initialisation of the fluent $opNN_iUserOperation$ is triggered by the invocation of the user operation ($ic\_opNN_iUserOperation$). The invariant part in the pattern is depicted in bold.

Rule part
$\boldsymbol{Happens}(\boldsymbol{ic}\_InitialEvent, t_1, R(t_1, t_1)) \Rightarrow$ $\boldsymbol{Happens}(\boldsymbol{ic\_op}NN_1\boldsymbol{UserOperation}, t_2, R(t_1, t_2))$ $\wedge ... \wedge$ $\boldsymbol{Happens}(\boldsymbol{ic\_op}NN_i\boldsymbol{UserOperation}, t_{i+1}, R(t_1, t_{i+1})) \wedge$ $\boldsymbol{Happens}(\boldsymbol{ir\_op}NN_1\boldsymbol{UserOperation}, t_{i+2}, R(t_2, t_2 + t_{Current\_cog_{(1)}}))$ $\wedge ... \wedge$ $\boldsymbol{Happens}(\boldsymbol{ir\_op}NN_i\boldsymbol{UserOperation}, t_{2i+1}, R(t_{i+1}, t_{i+1} + t_{Current\_cog_{(i)}}))$

Assumption part
$\mathbf{Happens}(\boldsymbol{ic\_op}NN\boldsymbol{UserOperation}, t_1, R(t_1, t_1)) \Rightarrow$ $\mathbf{Initiates}(\boldsymbol{ic\_op}NN\boldsymbol{UserOperation}, \boldsymbol{op}NN\boldsymbol{UserOperation}, t_1)$ ... $\mathbf{Happens}(\boldsymbol{ic\_op}NN\boldsymbol{UserOperation}, t_1, R(t_1, t_1)) \Rightarrow$ $\mathbf{Initiates}(\boldsymbol{ic\_op}NN\boldsymbol{UserOperation}, \boldsymbol{op}NN\boldsymbol{UserOperation}, t_1)$

Table 4.46: Cognition Rule Pattern 6

As an example, consider the extract of the Wo-SBS scenario depicted in Figure 4.4, assume a user with an “average” level of *cognition* interacting with the system. Applying the *cognition* pattern depicted in Table 4.46 results in the pattern instantiations depicted in Table 4.47.

The pattern instantiations depicted in Table 4.47 consider *user operations oploginUserOperation*, *opselectFeatureUserOperation*, and *opstandardUIMailUserOperation*, which correspond to operations involved in the part of the Wo-SBS associated to an “average” level of *cognition*. In the monitor rule the occurrence of the initial event ($ic\_startWoSBS$) must be followed by the invocation of the *user operations* and

Rule part
$Happens(ic\_startWoSBS, t_1, R(t_1, t_1)) \Rightarrow$ $Happens(ic\_oploginUserOperation, t_2, R(t_1, t_2)) \wedge$ $Happens(ic\_opselectFeatureUserOperation, t_3, R(t_1, t_3)) \wedge$ $Happens(ic\_opstandardUIMailUserOperation, t_4, R(t_1, t_4)) \Rightarrow$ $Happens(ir\_oploginUserOperation, t_5, R(t_2, t_2 + 3.75))$ $Happens(ir\_opselectFeatureUserOperation, t_6, R(t_3, t_3 + 3.75))$ $Happens(ir\_opstandardUIMailUserOperation, t_7, R(t_4, t_4 + 3))$

Assumption part
$Happens(ic\_oploginUserOperation, t_1, R(t_1, t_1)) \Rightarrow$ $Initiates(ic\_oploginUserOperation, oploginUserOperation, t_1)$
$Happens(ic\_opselectFeatureUserOperation, t_1, R(t_1, t_1)) \Rightarrow$ $Initiates(ic\_opselectFeatureUserOperation,$ $opselectFeatureUserOperation, t_1)$
$Happens(ic\_opstandardUIMailUserOperation, t_1, R(t_1, t_1)) \Rightarrow$ $Initiates(ic\_opstandardUIMailUserOperation,$ $opstandardUIMailUserOperation, t_1)$

Table 4.47: Instantiation of Cognition Rule Pattern 6

their responses. Thus after the occurrence of the initial event $ic\_startWoSBS$, the invocation of operation *oploginUserOperation* (event $ic\_oploginUserOperation$) must be followed by the response of the operation (event $ir\_oploginUserOperation$) in no more than 3 seconds. This time constraint represents 75% of the time specified for the execution of this operation ($4*0.75$ seconds) given the "average" level of *cognition* of the user. Similarly, monitor rules state that after the occurrence of the initial event, invocations $ic\_opselectFeatureUserOperation$ and $ic\_opstandardUIMailUserOperation$ should be followed by the operation responses ($ir\_opselectFeatureUserOperation$ and $ir\_opstandardUIMailUserOperation$) in no more than 3.75 seconds ($5 * 0.75$ seconds for each operation). The fluents related to the *user operations* are initiated in the assumption part of the instantiated pattern.

Cognition Rule Pattern 7

The pattern for *cognition* shown in Table 4.48 verifies the response time of a *user operation* after its state starts holding. More specifically, the pattern relies on the change of states of a *user operation* as an indicator the operation has been invoked. Note that the state of an operation, i.e. a fluent, starts holding when it is invoked. The pattern verifies that the response of the *user operation* occurs in the time constraint defined by the time of the operation and the level of *cognition* of the user.

As shown in the rule part of the pattern in Table 4.48, the changes of state of a fluent associated to a *user operation*, must be followed by the response of the operation within a defined period of time. The period of time is represented by the time variable $t_{Current\_cog}$ and its computation considers the time specified for the *user operation* multiplied by the percentage associated to the level of *cognition* of the user. In the assumption part of the pattern the fluent $opNNUserOperation$ represents the operation "opNNUserOperation" is active, i.e. has been triggered. The initialisation of the fluent $opNNUserOperation$ is triggered by the invocation of the user operation ($ic\_opNNUserOperation$). The invariant part in the pattern is depicted in bold.

Rule part
$\neg \boldsymbol{HoldsAt}(\boldsymbol{opNNUserOperation}, t_1) \wedge$
$\boldsymbol{HoldsAt}(\boldsymbol{opNNUserOperation}, t_2) \Rightarrow$
$\boldsymbol{Happens}(\boldsymbol{ir\_opNNUserOperation}, t_3, R(t_2, t_2 + t_{Current\_cog}))$
$t_1 \ < \ t_2$

Assumption part
$\mathbf{Happens}(\boldsymbol{ic\_opNNUserOperation}, t_1, R(t_1, t_1)) \Rightarrow$
$\mathbf{Initiates}(\boldsymbol{ic\_opNNUserOperation}, \boldsymbol{opNNUserOperation}, t_1)$

Table 4.48: Cognition Rule Pattern 7

In the case two or more *user operations* have been identified in the part of the SBS to be executed, the *cognition* pattern in Table 4.44 is instantiated for each one of them.

As an example, consider the extract of the Wo-SBS scenario depicted in Figure 4.4, assume a user with an "average" level of *cognition* interacting with the system. Applying the *cognition* pattern depicted in Table 4.48 results in the pattern instantiations depicted in Table 4.49.

Rule part
$\neg HoldsAt(oploginUserOperation, t_1) \wedge$ $HoldsAt(oploginUserOperation, t_2) \Rightarrow$ $Happens(ir\_oploginUserOperation, t_3, R(t_2, t_2 + 3.75))$
$\neg HoldsAt(opselectFeatureUserOperation, t_1) \wedge$ $HoldsAt(opselectFeatureUserOperation, t_2) \Rightarrow$ $Happens(ir\_opselectFeatureUserOperation, t_3, R(t_2, t_2 + 3.75))$
$\neg HoldsAt(opstandardUIMailUserOperation, t_1) \wedge$ $HoldsAt(opstandardUIMailUserOperation, t_2) \Rightarrow$ $Happens(ir\_opstandardUIMailUserOperation, t_3, R(t_2, t_2 + 3))$

Assumption part
$Happens(ic\_oploginUserOperation, t_1, R(t_1, t_1)) \Rightarrow$ $Initiates(ic\_oploginUserOperation, oploginUserOperation, t_1)$
$Happens(ic\_opselectFeatureUserOperation, t_1, R(t_1, t_1)) \Rightarrow$ $Initiates(ic\_opselectFeatureUserOperation,$ $opselectFeatureUserOperation, t_1)$
$Happens(ic\_opstandardUIMailUserOperation, t_1, R(t_1, t_1)) \Rightarrow$ $Initiates(ic\_opstandardUIMailUserOperation,$ $opstandardUIMailUserOperation, t_1)$

Table 4.49: Instantiation of Cognition Rule Pattern 7

The pattern instantiations depicted in Table 4.49 consider *user operations oploginUserOperation*, *opselectFeatureUserOperation*, and *opstandardUIMailUserOperation*, which correspond to operations involved in the part of the Wo-SBS associated to an "average" level of *cognition*. In each monitor rule the change of state of the fluent associated to the *user operation* signifies the invocation of the operations. The change of state of the fluent $oploginUserOperation$, from not holding to holding, must be followed by the response of the operation ($ir\_oploginUserOperation$) in no more than 3 seconds. This time constraint represents 75% of the time specified for the execution of this operation ($4*0.75$ seconds) given the "average" level of *cognition* of the user. Similarly, the other two monitor rules state that after the changes of states (for fluents $opselectFeatureUserOperation$ and $opstandardUIMailUserOperation$) responses from each operation must occur in no more than 3.75 seconds ($5 * 0.75$ seconds for each operation). In the assumption part of the instantiation, the *user operations* are initiated as they are invoked.

4.2.6 Preferences Patterns

In the literature it is possible to find several studies dealing with user *preferences*. These studies cover different topics, ranging from preferences dealing with information retrieval, e.g. [63][150][170] to users' intuitive or emotional preferences, e.g. [97][123][139].

In our work we have created a set of *preferences* patterns to support the representation of alternative computational resources. More specifically, we focus on the *time* and *security preferences* of a user, for the complete or partial execution of a SBS, and on the *reliability* preferences of a user, when a system is invoked. These *preferences* have been addressed in several works including [34][56][74][95][127].

Time Preferences Pattern

A time preference pattern concerns with the execution of part of a SBS within a specified time constraint. More specifically the SBS, or part of it, must be executed within a time constraint that is specified by the user. A *time preferences* pattern is depicted in Table 4.50. In the rule part of the pattern, the invocation of an operation (ic_Event_i), must be followed by the response of an operation (ir_Event_j), within the time constraint specified by the user. The time constraint specified by the user is represented by the time variable $t_{preferences}$. In the assumption part of the pattern the invocation of the involved operations imply the associated fluents have been initiated. The invariant part in the pattern is depicted in bold.

Rule part
$\boldsymbol{Happens}(\boldsymbol{ic}\_Event_i, t_1, R(t_1, t_1)) \Rightarrow$ $\boldsymbol{Happens}(\boldsymbol{ir}\_Event_j, t_2, R(t_1, t_1 + t_{preferences}))$

Assumption part
$\boldsymbol{Happens}(\boldsymbol{ic}\_Event_i, t_1, R(t_1, t_1)) \Rightarrow$ $\boldsymbol{Initiates}(\boldsymbol{ic}\_Event_i, Event_i, t_1)$
$\boldsymbol{Happens}(\boldsymbol{ic}\_Event_j, t_1, R(t_1, t_1)) \Rightarrow$ $\boldsymbol{Initiates}(\boldsymbol{ic}\_Event_j, Event_j, t_1)$

Table 4.50: Time Preference Rule Pattern

As an example, consider the extract of the Wo-SBS scenario depicted in Figure 4.4 and assume the user has a response time *preference* of 5 seconds for the execution of operations *checkAccess* and *enableMessagingService*. Applying the *time preferences* pattern depicted in Table 4.50 results in the pattern instantiation depicted in Table 4.51.

Rule part
$Happens(ic\_checkAccess, t_1, R(t_1, t_1)) \Rightarrow$ $Happens(ir\_enableMessagingService, t_2, R(t_1, t_1 + 5))$

Assumption part
$Happens(ic\_checkAccess, t_1, R(t_1, t_1)) \Rightarrow$ $Initiates(ic\_checkAccess, checkAccess, t_1)$
$Happens(ic\_enableMessagingService, t_1, R(t_1, t_1)) \Rightarrow$ $Initiates(ic\_enableMessagingService, enableMessagingService, t_1)$

Table 4.51: Instantiation of Time Preferences Rule Pattern

The instantiation of the pattern shown in Table 4.51 specifies that the invocation of operation *checkAccess* ($ic\_checkAccess$) should be followed by the response of operation *enableMessagingService* ($ir\_enableMessagingService$) in an interval of time that does not exceed 5 seconds. The assumption part represents the initialisation of the fluents associated to the operations, as a result of the corresponding invocations.

Security Preferences Pattern

Security preferences patterns are concerned with the total or partial execution of a SBS while properties regarding security requirements are enabled.

In the patterns, we assume enabled security properties are represented by a generic fluent, *security*, and that its initialisation has been previously specified. The enabled state of the fluent *security* denotes security properties hold, while the disabled state denotes security properties do not hold.

The above assumption is necessary since states guaranteeing *security* properties can be triggered by specific events, states, or combination of both. Even more, triggers enabling *security* properties can change from one system to another.

In addition to the above, security requirements may focus, for example, on the integrity or confidentiality of a system, e.g. [36][218][223]. In our work we consider the *security* concept form a general perspective; more specifically we consider security as the *absence of threats in a system*, as specified in [27][129][247].

A *security preference* pattern is depicted in Table 4.52. In the pattern, the enabled security property is represented by the fluent *security*. In the rule part of the pattern, invocations (ic_Event_i) and responses (ir_Event_j) of a set of operations must occur while the security property is enabled. The set of invocations and responses correspond to operations to which *security* is required. In the assumption part the invocations of the different operations trigger the states corresponding to the associated operations. The invariant part of the pattern is depicted in bold.

As an example, consider the extract of the Wo-SBS scenario depicted in Figure 4.5 and assume a user requires a secure access when authenticating into the Wo-SBS.

Rule part
$\boldsymbol{Happens}(\boldsymbol{ic}\_Event_1, t_1, R(t_1, t_1)) \wedge$
$\boldsymbol{Happens}(\boldsymbol{ir}\_Event_1, t_2, R(t_1, t_2))$
$\wedge ... \wedge$
$\boldsymbol{Happens}(\boldsymbol{ic}\_Event_i, t_{(2i-1)}, R(t_{(2i-1)}, t_{(2i-1)})) \wedge$
$\boldsymbol{Happens}(\boldsymbol{ir}\_Event_i, t_{2i}, R(t_{(2i-1)}, t_{2i})) \Rightarrow$
$\boldsymbol{HoldsAt}(\mathbf{security}, t_1)$
$\wedge ... \wedge$
$\boldsymbol{HoldsAt}(\mathbf{security}, t_{2i})$

Assumption part
$\boldsymbol{Happens}(\boldsymbol{ic}\_Event_1, t_1, R(t_1, t_1)) \Rightarrow$
$\boldsymbol{Initiates}(\boldsymbol{ic}\_Event_1, Event_1, t_1)$
...
$\boldsymbol{Happens}(\boldsymbol{ic}\_Event_i, t_1, R(t_1, t_1)) \Rightarrow$
$\boldsymbol{Initiates}(\boldsymbol{ic}\_Event_i, Event_i, t_1)$

Table 4.52: Security Preference Rule Pattern

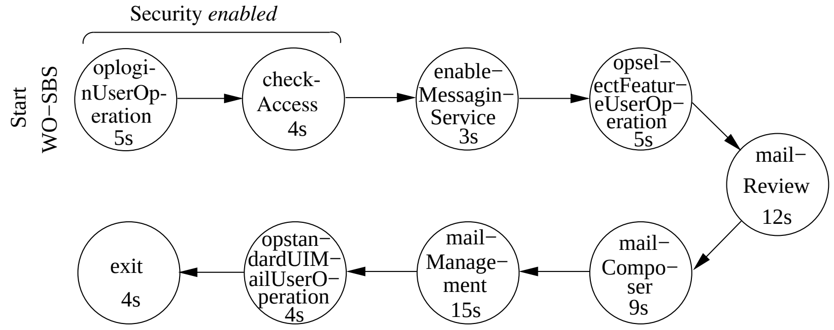

Security *enabled*
Start
WO−SBS
oplogi-nUserOp-eration 5s
check-Access 4s
enable−Messagin−Service 3s
opsel−ectFeatur−eUserOp−eration 5s
mail−Review 12s
mail−Compo−ser 9s
mail−Manage−ment 15s
opstan−dardUIM−ailUserO−peration 4s
exit 4s

Figure 4.5: Service Specification Sequence for Security

In this case the *security* fluent must hold during the authentication process (operations *oploginUserOperation* and *checkAccess*).

Applying the *security preference* pattern depicted in Table 4.52, results in the pattern instantiations depicted in Table 4.53.

The instantiation of the pattern shown in Table 4.53 specifies that the invocations and responses of operations *oploginUserOperation* (i.e. events *ic_oploginUserOperation* and *ir_oploginUserOperation*) and *checkAccess* (i.e. events *ic_checkAccess* and *ir_checkAccess*)

Rule part
$Happens(ic\_oploginUserOperation, t_1, R(t_1, t_1)) \wedge$ $Happens(ir\_oploginUserOperation, t_2, R(t_1, t_2)) \wedge$ $Happens(ic\_checkAccess, t_3, R(t_3, t_3)) \wedge$ $Happens(ir\_checkAccess, t_4, R(t_3, t_4)) \Rightarrow$ $HoldsAt(security, t_1) \wedge$ $HoldsAt(security, t_2) \wedge$ $HoldsAt(security, t_3) \wedge$ $HoldsAt(security, t_4)$

Assumption part
$Happens(ic\_oploginUserOperation, t_1, R(t_1, t_1)) \Rightarrow$ $Initiates(ic\_oploginUserOperation, oploginUserOperation, t_1)$
$Happens(ic\_checkAccess, t_1, R(t_1, t_1)) \Rightarrow$ $Initiates(ic\_checkAccess, checkAccess, t_1)$

Table 4.53: Instantiation of Security Preference Rule Pattern

should occur at times t_1, t_2, t_3, and t_4 and that at each occurrence the *security* fluent must hold (be enabled). The assumption part represents the states associated to the operations, have been triggered by the corresponding invocations.

Note that, as previously pointed out, it is not possible to specify which combination of states or actions trigger the *security* fluent. As a result, the instantiated pattern in Table 4.53 is concerned with the state of the *security* fluent but not its initialisation.

Reliability Preferences Pattern

The patterns concerned with the *reliability* of a system focus on the expected behaviour of a system with respect to the desired level of performance, specified by the user, of the system under normal circumstances. More specifically, the expected behaviour of a service composition, expressed in terms of *reliability*, should be equal or greater than the level of *reliability* specified by the user.

Similar to the security pattern, we consider *reliability* in a broad sense. More specifically, we consider *reliability* as the probability a system performs correctly its intended functionality over a specified period of time [73][86][235].

We assume the *reliability* for a SBS has been computed for the whole composition and that it is constantly updated after each execution of the system. This assumption is based on the previous work presented in [233]. The *reliability* computed for a system is represented by the holding fluent *systemReliability*, while the *reliability* specified by the user is represented by the holding fluent *userReliability*.

A *reliability preference* pattern is depicted in Table 4.54. The rule part of the pattern states that if the user level of *reliability* is specified while invoking the SBS (*ic_InitialEvent*), i.e. that the fluent *userReliability* holds at the time of the first invocation, the level of *reliability* of the SBS (fluent *systemReliability*) should be equal or greater than the one specified by the user. In the assumption part the invocation of the initial event implies the *reliability* for the SBS has been defined. The invariant part of the pattern is depicted in bold.

Rule part
$\boldsymbol{HoldsAt}(userReliability, t_1) \wedge$ $\boldsymbol{Happens}(\boldsymbol{ic}\_InitialEvent, t_1, R(t_1, t_1)) \Rightarrow$ $\boldsymbol{HoldsAt}(userReliability \leq systemReliability, t_1)$

Assumption part
$\boldsymbol{Happens}(\boldsymbol{ic}\_InitialEvent, t_1, R(t_1, t_1)) \Rightarrow$ $\boldsymbol{HoldsAt}(systemReliability, t_1)$

Table 4.54: Reliability Preference Rule Pattern

As an example consider the extract of the SBS depicted in Figure 4.5 and assume a user requires the system to be reliable 90% of the time; assume also the computed *systemReliability* is 0.95. Table 4.55 shows the instantiation of the pattern previously depicted in Table 4.54. In the instantiation, the reliability of the user (0.9) is less than the reliability computed for the system (0.95) when invoking the SBS (*ic_startWoSBS*), the rule will not be violated. In the case in which the level of reli-

Rule part
$\mathit{HoldsAt}(0.9, t_1) \wedge$ $\mathit{Happens}(\mathit{ic\_startWoSBS}, t_1, R(t_1, t_1)) \Rightarrow$ $\mathit{HoldsAt}(0.9 \leq 0.95, t_1)$

Assumption part
$\mathit{Happens}(\mathit{ic\_startWoSBS}, t_1, R(t_1, t_1)) \Rightarrow$ $\mathit{HoldsAt}(\mathit{systemReliability}, t_1)$

Table 4.55: Instantiation of Reliability Preference Rule Pattern

ability computed for the system is equal or less than the level of reliability specified for the user, the rule based on the reliability pattern is violated.

4.2.7 Need Pattern

The user need can be defined as a task, objective or goal a user wants to perform and accomplish with the help of a system [125][145][240].

The need of a user goes beyond the specification of a set of tasks to be performed, it also considers (in order to successfully accomplish the goal) the correct execution of a set of tasks. For example, while in the *role* context type the concern was on the invocation of those operations related to the *role* of the user, whether the involved operations were executed correctly or not was not an issue. More specifically, when dealing with the *need* context type the concern is on the correct invocation, operation, and termination of the involved operations.

The patterns for *need* context type focus on the correct execution of the part of the SBS related to the *need* of the user. A way of verifying whether a system is executing as expected, is by analysing all invocations and responses occur within expected time constraints.

Need Rule Pattern 1

A pattern for the *need* context type is depicted in Table 4.56. The rule part of the *need* pattern states that after the invocation of the initial event (*ic_InitialEvent*) at a time t_1, a set of operations invocations and responses should occur. The operations correspond to those operations identified as related to the *need* of the user. The invocation and response of each operation should occur at defined times. More specifically, the time variable t_{First} represents the maximum amount of time between the occurrence of the initial event, and the invocation of the first operation associated to the part of the SBS related to the *need* of the user. The operations related to the *need* of the user should be executed in a defined order and the time constraint defined for the response of each operation must not be exceeded. The time constraint for each operation is represented by the time variable T_{ExpOp_i}, which states the maximum amount of time allowed between the invocation and response of an operation. The time constraints consider small time delays between consecutive events of ten milliseconds. The assumption part of the pattern states that each request of an operation *ic_Event*$_i$ triggers the initialisation of the state *Event*$_i$, which represents the active state of the operation. The invariant of the pattern is depicted in bold.

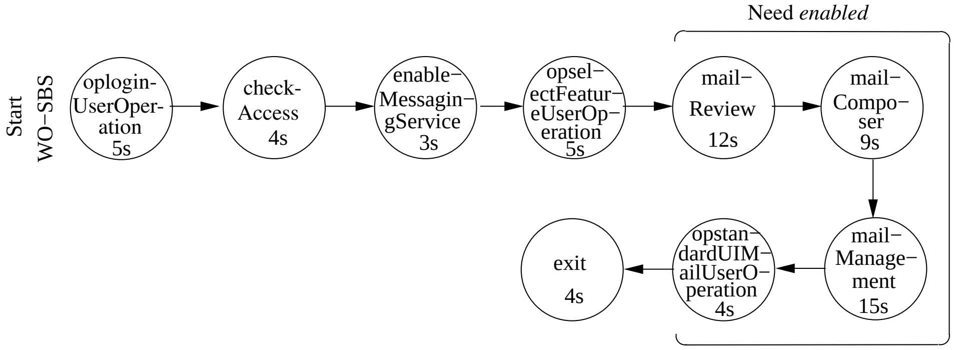

Need *enabled*
Start
WO−SBS
oplogin-UserOper-ation 5s
check-Access 4s
enable−Messagin−gService 3s
opsel−ectFeatur−eUserOp−eration 5s
mail−Review 12s
mail−Compo−ser 9s
mail−Manage−ment 15s
opstan−dardUIM−ailUserO−peration 4s
exit 4s

Figure 4.6: Service Specification Sequence and Need Context Type

As an example consider the sequence depicted in Figure 4.6 where operations

Rule part
$\boldsymbol{Happens}(\boldsymbol{ic}\_InitialEvent, t_1, R(t_1, t_1))$
$\Rightarrow$
$\boldsymbol{Happens}(\boldsymbol{ic}\_Event_1, t_2, R(t_1, t_1 + t_{First})) \wedge$
$\boldsymbol{Happens}(\boldsymbol{ir}\_Event_1, t_3, R(t_2, t_2 + T_{ExpOp_1})) \wedge$
$\boldsymbol{Happens}(\boldsymbol{ic}\_Event_2, t_4, R(t_3, t_3 + 0.01) \wedge$
$\boldsymbol{Happens}(\boldsymbol{ir}\_Event_2, t_5, R(t_4, t_4 + T_{ExpOp_2}))$
$\wedge \ldots \wedge$
$\boldsymbol{Happens}(\boldsymbol{ic}\_Event_{\lceil (n-1)/2 \rceil}, t_{n-1}, R(t_{n-2}, t_{n-2} + 0.01)) \wedge$
$\boldsymbol{Happens}(\boldsymbol{ir}\_Event_{\lceil (n-1)/2 \rceil}, t_n, R(t_{n-1}, t_{n-1} + T_{ExpOp_{\lceil (n-1)/2 \rceil}}))$

Assumption part
$\boldsymbol{Happens}(\boldsymbol{ic}\_Event_n, t_1, R(t_1, t_1))$
$\Rightarrow$
$\boldsymbol{Initiates}(\boldsymbol{ic}\_Event_n, Event_n, t_1)$

Table 4.56: Need Rule Pattern 1

mailReview, *mailComposer*, *mailManagement*, and *opstandardUIMailUserOperation* are related to the *need* of a user. Applying the pattern depicted in Table 4.56 results in the pattern instantiations depicted in Table 4.57.

Need Rule Pattern 2

Another *need* pattern is specified in Table 4.58. In this pattern each operation, identified as related to the *need* of the user, is verified according to its invocation followed by its response. The time variable $t_{execution}$ represents the time constraint between the initial event (*ic_InitialEvent*) and the invocation of an operation (*ic_Event$_i$*) related to the *need* of the user. The time constraint between the invocation of an operation and the response of the operation (*ir_Event$_i$*) is represented by the time variable t_{Event}. The computed time constraints consider small delays of ten milliseconds between consecutive events.

The assumption part of the pattern states that an invocation of an operation triggers the initialisation of the associated state, which represents the active state of the

Rule part
$Happens(ic\_startWoSBS, t_1, R(t_1, t_1))$ $\Rightarrow$ $Happens(ic\_mailReview, t_2, R(t_1, t_1 + 17.05)) \wedge$ $Happens(ir\_mailReview, t_3, R(t_2, t_2 + 12)) \wedge$ $Happens(ic\_mailComposer, t_4, R(t_3, t_3 + 0.01) \wedge$ $Happens(ir\_mailComposer, t_5, R(t_4, t_4 + 9)) \wedge$ $Happens(ic\_mailManagement, t_6, R(t_5, t_5 + 0.01)) \wedge$ $Happens(ir\_mailManagement, t_7, R(t_6, t_6 + 15)) \wedge$ $Happens(ic\_opstandardUIMailUserOperation, t_8, R(t_7, t_7 + 0.01)) \wedge$ $Happens(ir\_opstandardUIMailUserOperation, t_9, R(t_8, t_8 + 4))$

Assumption part
$Happens(ic\_mailReview, t_1, R(t_1, t_1)) \Rightarrow$ $Initiates(ic\_mailReview, mailReview, t_1)$
$Happens(ic\_mailComposer, t_1, R(t_1, t_1)) \Rightarrow$ $Initiates(ic\_mailComposer, mailComposer, t_1)$
$Happens(ic\_mailManagement, t_1, R(t_1, t_1)) \Rightarrow$ $Initiates(ic\_mailManagement, mailManagement, t_1)$
$Happens(ic\_opstandardUIMailUserOperation, t_1, R(t_1, t_1)) \Rightarrow$ $Initiates(ic\_opstandardUIMailUserOperation,$ $opstandardUIMailUserOperation, t_1)$

Table 4.57: Instantiation of Need Rule Pattern 1

Rule part
$\boldsymbol{Happens}(\boldsymbol{ic}\_InitialEvent, t_1, R(t_1, t_1)) \Rightarrow$ $\boldsymbol{Happens}(\boldsymbol{ic}\_Event_i, t_2, R(t_1, t_1 + t_{execution})) \wedge$ $\boldsymbol{Happens}(\boldsymbol{ir}\_Event_i, t_3, R(t_2, t_2 + T_{Event}))$

Assumption part
$\boldsymbol{Happens}(\boldsymbol{ic}\_Event_i, t_1, R(t_1, t_1)) \Rightarrow$ $\boldsymbol{Initiates}(\boldsymbol{ic}\_Event_i, Event_i, t_1)$

Table 4.58: Need Rule Pattern 2

operation. The invariant of the pattern is depicted in bold. The pattern is applied to each operation identified as related to the *need* of the user.

As an example consider the sequence depicted in Figure 4.6 where operations *mailReview*, *mailComposer*, *mailManagement*, and *opstandardUIMailUserOpera-*

Rule part
$Happens(ic\_startWoSBS, t_1, R(t_1, t_1)) \Rightarrow$ $Happens(ic\_mailReview, t_2, R(t_1, t_1 + 17.05))\wedge$ $Happens(ir\_mailReview, t_3, R(t_2, t_2 + 12))$
$Happens(ic\_startWoSBS, t_1, R(t_1, t_1)) \Rightarrow$ $Happens(ic\_mailComposer, t_2, R(t_1, t_1 + 29.06) \wedge$ $Happens(ir\_mailComposer, t_3, R(t_2, t_2 + 9))$
$Happens(ic\_startWoSBS, t_1, R(t_1, t_1)) \Rightarrow$ $Happens(ic\_mailManagement, t_2, R(t_1, t_1 + 38.07)) \wedge$ $Happens(ic\_mailManagement, t_3, R(t_2, t_2 + 15))$
$Happens(ic\_startWoSBS, t_1, R(t_1, t_1)) \Rightarrow$ $Happens(ic\_opStandardMailUserOperation, t_2, R(t_1, t_1 + 53.08)) \wedge$ $Happens(ic\_opStandardMailUserOperation, t_3, R(t_2, t_2 + 4))$

Assumption part
$Happens(ic\_mailReview, t_1, R(t_1, t_1)) \Rightarrow$ $Initiates(ic\_mailReview, mailReview, t_1)$
$Happens(ic\_mailComposer, t_1, R(t_1, t_1)) \Rightarrow$ $Initiates(ic\_mailComposer, mailComposer, t_1)$
$Happens(ic\_mailManagement, t_1, R(t_1, t_1)) \Rightarrow$ $Initiates(ic\_mailManagement, mailManagement, t_1)$
$Happens(ic\_opstandardUIMailUserOperation, t_1, R(t_1, t_1)) \Rightarrow$ $Initiates(ic\_opstandardUIMailUserOperation,$ $opStandardMailUserOperation, t_1)$

Table 4.59: Instantiation of Need Rule Pattern 2

tion are related to the *need* of a user. Applying the pattern depicted in Table 4.58 results in the pattern instantiations depicted in Table 4.59. The invocation of each operation (i.e. events *ic_mailReview*, *ic_mailComposer*, *ic_mailManagement*, and *ic_opstandard-UIMailUserOperation*) should occur after the occurrence of the initial event (*ic_start-WoSBS*) and within the computed time constraint consisting of the previous operations plus the time delays. For each operation invocation, there should be a response (represented in each case by events *ir_mailReview*, *ir_mailComposer*, *ir_mailManage-ment*, and *ir_opstandardUIMailUserOperation*) occurring after the invocation of the operation and within the time constraint specified for the operation.

4.2.8 Cognition and Role Pattern

We have also created patterns to represent more than one context type at the same time in order to denote combinations of context types. The Cognition and Role Pattern is concerned with the times that a user takes to interact with the system for the various user's operations (involving the *cognition* context type) and the order in which the operations, requiring user participation, need to be executed (involving the *role* context type). The semantic of such situation cannot be represented by patterns concerned with the isolated context types. Moreover, if both context types are specified in an event, the framework will identify and instantiate the patterns for each of the context types and execute the monitor rules resulting from these instantiations. However, the execution of these rules do not provide the semantic meaning of verifying if the user operations are specified in a certain order and each of them are executed in a certain amount of time depending on the *cognition* level of the user. As an example, consider a pattern for *cognition* and *role* context types shown in Table 4.60. Table 4.61 presents the instantiation of this pattern for the extract if the Wo-SBS example with a user with "average" level of cognition and the role "personal user". The extract of the part of the Wo-SBS application associated to the above example is shown in Figure 4.7.

The rule part in the pattern in Table 4.60 considers all the user operations involved in the execution when a user, in a certain *role*, accesses the system. The pattern states that between the occurrence of the initial event in the system (*ic_InitialEvent*) and the occurrence of the last response (*ir_FinalResponse*) in the system, an event or set of events, concerned with user interactions should occur in a certain order. More specifically, each invocation of a user operation (*ic_opOperationNUserOperation*) must be

Rule part
$\boldsymbol{Happens}(\boldsymbol{ic}\_InitialEvent, t_1, R(t_1, t_1)) \wedge$ $\boldsymbol{Happens}(\boldsymbol{ir}\_FinalResponse, t_2, R(t_2, t_2)) \Rightarrow$ $\boldsymbol{Happens}(\boldsymbol{ic\_op}NN_1\boldsymbol{UserOperation}, t_3, R(t_1, t_2)) \wedge$ $\boldsymbol{Happens}(\boldsymbol{ir\_op}NN_1\boldsymbol{UserOperation}, t_4, R(t_3, t_3 + t_{current\_cog}))$ $\wedge \ldots \wedge$ $\boldsymbol{Happens}(\boldsymbol{ic\_op}NN_j\boldsymbol{UserOperation}, t_{2j+1}, R(t_{2j}, t_2)) \wedge$ $\boldsymbol{Happens}(\boldsymbol{ir\_op}NN_j\boldsymbol{UserOperation}, t_{2j+2}, R(t_{2j+1}, t_{2j+1} + t_{current\_cog}))$

Assumption part
$\boldsymbol{Happens}(\boldsymbol{ic\_op}NN_1\boldsymbol{UserOperation}, t_1, R(t_1, t_1)) \Rightarrow$ $\boldsymbol{Initiates}(\boldsymbol{ic\_op}NN_1\boldsymbol{UserOperation}, \boldsymbol{op}NN_1\boldsymbol{UserOperation}, t_1)$ ... $\boldsymbol{Happens}(\boldsymbol{ic\_op}NN_j\boldsymbol{UserOperation}, t_1, R(t_1, t_1)) \Rightarrow$ $\boldsymbol{Initiates}(\boldsymbol{ic\_op}NN_j\boldsymbol{UserOperation}, \boldsymbol{op}NN_j\boldsymbol{UserOperation}, t_1)$

Table 4.60: Cognition-Role Pattern

Rule part
$Happens(ic\_startWoSBS, t_1, R(t_1, t_1)) \wedge$ $Happens(ir\_exit, t_2, R(t_2, t_2)) \Rightarrow$ $Happens(ic\_oploginUserOperation, t_3, R(t_1, t_2)) \wedge$ $Happens(ir\_oploginUserOperation, t_4, R(t_3, t_3 + 5 * 0.75)) \wedge$ $Happens(ic\_opselectFeatureUserOperation, t_5, R(t_4, t_2)) \wedge$ $Happens(ir\_opselectFeatureUserOperation, t_6, R(t_5, t_5 + 5 * 0.75)) \wedge$ $Happens(ic\_opstandardUIMailUserOperation, t_7, R(t_6, t_2)) \wedge$ $Happens(ir\_opstandardUIMailUserOperation, t_8, R(t_7, t_7 + 4 * 0.75))$

Assumption part
$Happens(ic\_oploginUserOperation, t_1, R(t_1, t_1)) \Rightarrow$ $Initiates(ic\_oploginUserOperation, oploginUserOperation, t_1)$
$Happens(ic\_opselectFeatureUserOperation, t_1, R(t_1, t_1)) \Rightarrow$ $Initiates(ic\_opselectFeatureUserOperation,$ $opselectFeatureUserOperation, t_1)$
$Happens(ic\_opstandardUIMailUserOperation, t_1, R(t_1, t_1)) \Rightarrow$ $Initiates(ic\_opstandardUIMailUserOperation,$ $opstandardUIMailUserOperation, t_1)$

Table 4.61: Instantiation of Cognition-Role Pattern

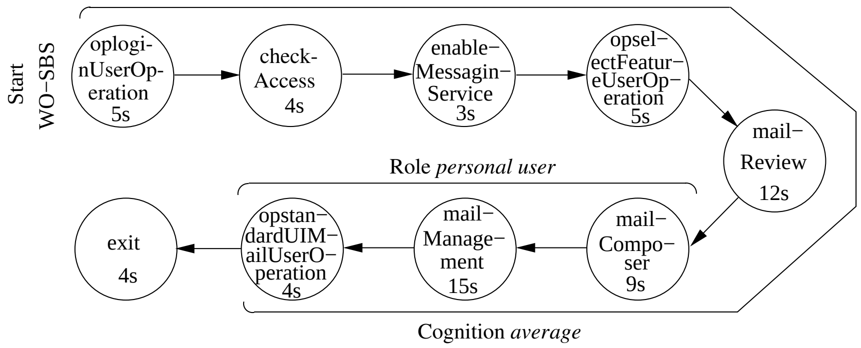


Figure 4.7: Service Specification Sequence, for a user with an “average” level of *cognition* and *role* “personal user”

followed by the response of the same operation (*ir_opOperationNUserOperation*) in no more than $t_{current\_cog}$ units of time. Time variable $t_{current\_cog}$ corresponds to the maximum amount of time for the execution of the user operation according to the *cognition* level of the user. The occurrence of two or more user operations should follow the order specified in the identified path. The assumption part of the pattern in Table 4.60 states that the invocation of a user operation ($ic\_opNN_iUserOperation$) triggers the initialisation of the associated state ($opNN_iUserOperation$) which represents the active state of the user operation.

The instantiation of the pattern shown in Table 4.61 specifies that the invocation and response of operation *oploginUserOperation* (i.e. events *ic_oploginUserOperation* and *ir_oploginUserOperation*) should be followed by the invocations and responses of operations *opselectFeatureUserOperation* (i.e. events *ic_opselectFeatureUserOperation* and *ir_opselectFeatureUserOperation*) and *opstandardUIMailUserOperation* (i.e. events *ic_opstandardUIMailUserOperation* and *ir_opstandardUIMailUserOperation*). The time constraints for the invocations and responses of each operation, consider the execution time for the operation and the “average” level of *cognition* of the user.

4.2.9 Cognition and Time Preferences Pattern

Another combined pattern is the *cognition* and *time preferences* pattern depicted in Table 4.62. The rule part in the pattern considers the invocation and response of a set of operations within a time constraint specified by the user. In the rule part the pattern also verifies (for the participating user operations) the user responses according to the level of *cognition*, and that the time constraint specified by the user (time preferences) is greater than the sum of the time constraints of the participating user operations. The latter ensures the *time preference* specified by the user can never go below the expected time required to perform the involved user operations.

Whether the time preferences specified by the user are below the sum of time constraints of the participating user operations is represented by the fluent *TimeConstraintOk*). In the case the fluent holds, the *time preferences* from the user are below the sum of the time constraints of the participating user operations. In the case the fluent does not hold, the time preferences are above the sum of the time constraints of the participating user operations.

The semantic of the above situation cannot be represented by patterns concerned with the isolated *cognition* and *time preferences* context types. Moreover, if both context types are specified in an event, the framework will identify and instantiate the patterns for each of the context types and execute the monitor rules resulting from these instantiations. However, the execution of these rules do not provide the semantic meaning of verifying if the time constraint specified by the user is in conflict with the expected time required for the execution of the involved user operations.

In the pattern in Table 4.62 the time to invoke the first operation (ic_Event_1) and to receive the response of the last operation (ir_Event_2) of the part of the SBS related to the *time preference*, should occur within the time constraint specified by the user (time variable $t_{preference}$). For each user operation, participating in the part of the

Rule part
$\boldsymbol{Happens}(\boldsymbol{ic\_}Event_1, t_1, R(t_1, t_1)) \Rightarrow$ $\boldsymbol{Happens}(\boldsymbol{ic\_op}NN_1\boldsymbol{UserOperation}, t_2, R(t_2, t_2)) \wedge$ $\boldsymbol{Happens}(\boldsymbol{ir\_op}NN_1\boldsymbol{UserOperation}, t_3, R(t_2, t_2 + t_{current\_cog_1})$ $\wedge \ldots \wedge$ $\boldsymbol{Happens}(\boldsymbol{ic\_op}NN_i\boldsymbol{UserOperation}, t_n, R(t_n, t_n)) \wedge$ $\boldsymbol{Happens}(\boldsymbol{ir\_op}NN_i\boldsymbol{UserOperation}, t_{n+1}, R(t_n, t_n + t_{current\_cog_i})) \wedge$ $\boldsymbol{Happens}(\boldsymbol{ir\_}Event_2, t_{n+2}, R(t_1, t_1 + t_{preference})) \wedge$ $\boldsymbol{HoldsAt}(TimeConstraintOk)$

Assumption part
$\boldsymbol{Happens}(\boldsymbol{ic\_}Event_1, t_1, R(t_1, t_1)) \Rightarrow$ $\boldsymbol{Initiates}(\boldsymbol{ic\_}Event_1, Event_1, t_1)$
$\boldsymbol{Happens}(\boldsymbol{ic\_}Event_2, t_1, R(t_1, t_1)) \Rightarrow$ $\boldsymbol{Initiates}(\boldsymbol{ic\_}Event_2, Event_2, t_1)$
$\boldsymbol{Happens}(\boldsymbol{ic\_op}NN_i\boldsymbol{UserOperation}, t_1, R(t_1, t_1)) \Rightarrow$ $\boldsymbol{Initiates}(\boldsymbol{ic\_op}NN_i\boldsymbol{UserOperation}, \boldsymbol{op}NN_i\boldsymbol{UserOperation}, t_1)$

Table 4.62: Cognition Time Preference Pattern

SBS concerned with the *time preference*, the invocation of the operation should be followed by the response of the user operation. The response should occur within the time constraint ($t_{current\_cog_i}$) which is computed according to the time to execute the user operation and the level of *cognition* of the user. The fluent *TimeConstraintOk* is initiated every time the sum of the time limits of the user operations (considering the *cognition* level of the user) is below the *time preference* specified by the user. More specifically $\sum_{j=1}^{i} t_{current\_cog_i} \leq t_{preference}$.

As an example, consider the extract of the SBS depicted in Figure 4.7 and assume the user specifies time preferences for the execution of operations *oploginUserOperation*, *checkAccess*, *enableMessagingService*, and *opselectFeatureUserOperation* of nine seconds. Assume also the user has an "average" level of *cognition*. The instantiation of pattern in Table 4.62 is shown in Table 4.63.

Note that in the instantiated pattern in Table 4.63 the first and last events corre-

Rule part
$Happens(ic\_oploginUserOperation, t_1, R(t_1, t_1)) \Rightarrow$ $Happens(ic\_oploginUserOperation, t_2, R(t_2, t_2)) \wedge$ $Happens(ir\_oploginUserOperation, t_3, R(t_2, t_2 + 5 * 0.75) \wedge$ $Happens(ic\_opselectFeatureUserOperation, t_4, R(t_4, t_4)) \wedge$ $Happens(ir\_opselectFeatureUserOperation, t_5, R(t_4, t_4 + 5 * 0.75)) \wedge$ $Happens(ir\_opselectFeatureUserOperation, t_6, R(t_1, t_1 + 9)) \wedge$ $HoldsAt(TimeConstraintOk)$

Assumption part
$Happens(ic\_oploginUserOperation, t_1, R(t_1, t_1)) \Rightarrow$ $Initiates(ic\_oploginUserOperation, oploginUserOperation, t_1)$
$Happens(ic\_opselectFeatureUserOperation, t_1, R(t_1, t_1)) \Rightarrow$ $Initiates(ic\_opselectFeatureUserOperation, opselectFeatureUserOperation, t_1)$

Table 4.63: Instantiation of Cognition Time Preference Pattern

spond to user operations, more specifically invocation *ic_oploginUserOperation* and response *ir_opselectFeatureUserOperation*. This is just a coincidence, in fact the first and last event in the instantiated pattern depend on the sequence of operations related to the *time preferences* specified by the user.

Note also that the instantiation of fluent *TimeConstraintOk* is not shown in the pattern. The instantiation of fluent *TimeConstraintOk* is the consequence of sum of time constraints related to the user operations and its comparison with the *time preference*. More specifically, the sum of the time constraints for the execution of operations *oploginUserOperation* and *opselectFeatureUserOperation* considering an "average" level of *cognition* (in both cases $5 * 0.75$ seconds), is less than the nine seconds specified in the *time preferences* of the user. In this case the fluent *TimeConstraintOk* is initiated. As an example, where the fluent *TimeConstraintOk* does not hold, consider the same scenario described above and a user with a "low" level of *cognition*. In this case the *time preference* specified by the user (nine seconds) is lower than the sum of the time constraints of the user operations, computed considering a “low” level of *cognition*. Consequently the fluent will not be initiated.

4.2.10 Considerations

This section summarises the characteristics and considerations of our context patterns.

The patterns represent templates for the generation of monitor rules concerned with the operations defined in a BPEL specification. The patterns can be applied whenever there is a defined sequence of operations. It is important to note that - as mentioned before - the proposed set of patterns do not cover loops in the service specification.

The identification of the part of a system related to a set of user context types, depends on a specific component in our framework (see *Path Identifier* in Section 3.8). Patterns are applied to the part of the system in order to generate monitor rules, only if there is a correspondence between the defined context types for the user, i.e. the value of the contexts, and the value of context types defined for a system.

It is perfectly possible for a user to be characterised by more than one context type. In such cases our approach is capable of dealing with them by applying the involved context type patterns accordingly.

It is also possible to have the same context type associated to different parts of the SBS, e.g. two instantiations of the *role* context type associated to two different sets of operations in the service specification (see Figure 4.2). Furthermore, as shown in Figure 4.7, the context types associated to the different parts of the system can overlap each other.

Our approach depends on the differentiation between those operations involving the participation of the user and those operations not involving the direct participation of the user. Our strategy is based on the syntax of the name of the operation and, if necessary, can be easily modified.

The exposed patterns consider time delays of ten milliseconds between the occurrence of consecutive events. These time delays are attributed to imponderable factors, e.g. network traffic, processing latency and their value has been established based on the experiments we have conducted (see Chapter 6). Although the time delay for the occurrence of consecutive events is fixed, it can be easily modified. Furthermore, the time delay, although realistic, is not really significant when compared to the time required for the execution of the operations, usually in the magnitude of seconds.

All patterns have an *invariant*, which corresponds to a predicate structure that, despite the instantiation of the context type and the service composition (i.e. the identified part of the SBS related to a specific context type), remains unmodified. This *invariant* is uniquely identifiable in each pattern and in those monitor rules based on the patterns. This *invariant* is useful in the identification, creation, modification, and removal of monitor rules. More specifically, *invariants* allows for the identification of context patterns that will be useful in the creation of monitor rules templates, called *semi-instantiated rules* (see Chapter 5). Even more, invariants are also used in the comparisons process performed to detect potentially suitable monitor rules from a repository.

4.3 Summary

In this chapter we have introduced and explained the formalism, *event calculus* (EC), used in our work for the specification of monitor rules. Based on the formalism, and on the user context types introduced in Chapter 3, we presented and described a set of context patterns that serve as templates for the specification of monitor rules in EC. The resulting monitor rules depend on the operations specified in the service composition and the instantiated user context types. Furthermore, we showed that for the same service specification and user context type, different monitor rules may be generated depending on the instance of the user context. We concluded the chapter by providing a general characterisation of the patterns.

In the following chapter we focus on the adaptation process. We present our strategy, based on the use of annotations, for the identification of the part of the SBS related to a user context type. We also explain the monitor rules instantiation process, which is based on the use of patterns. Finally we present the process for the identification, modification, creation, and removal of monitor rules and give a general example.

Chapter 5

Monitor Rules Adaptation Process

In this chapter we focus on the adaptation process of monitor rules for a given set of user context types and SBS. We also tackle the problem regarding user interaction with a service-based system. The contexts of concern include *cognition*, *role*, *skills*, *needs*, and *preferences* of a user. The instantiated context types are represented in user models based on the ontology we have created (see Chapter 3).

The framework relies on the use of monitor rule patterns (see Chapter 4) for the specification of monitor rules. Patterns are concerned with parts of a SBS specification. More specifically, a pattern is applied to a specific part of a SBS (see Section 5.2). The monitor rules, resulting from the application of the patterns, are used by a *monitor component* to verify the correct execution of the SBS. Overall, patterns are used *a)* to support the identification of monitor rules that need to be used for monitoring a SBS, *b)* to modify monitor rules in those cases in which a monitor rule is not completely suitable for monitoring a SBS, *c)* to create new monitor rules, in those cases where neither the identification or modification is possible, and *d)* to remove obsolete monitor rules. In order to perform the above mentioned activities our approach considers the existence of a monitor rules repository.

The chapter is structured as follows. In section 5.1 the adaptation process is explained; in this section we present the annotation strategy for the identification of the part of the SBS to be executed and present the monitor rule adaptation algorithm. We finish the chapter providing examples that cover the different activities related to the identification, creation, modification, and removal of monitor rules.

5.1 Adaptation Process

In the framework, the monitor adaptation process is triggered by an event representing context types of a user. For example, in the case in which Mary (Section 3.5) is using the Wo-SBS application in the first day of her holidays to organise the conference, the event represents the *role* of a "personal manager" with a "beginner" level of *skills*.

In the framework, we assume context types represented in the events identified by different types of sensors. Many existing approaches focus on the acquisition of context types based on the use of sensors for specific context types, e.g. temperature, location [47][230]. Research has also been conducted on how to identify context types related to the user [32][68][189]. For example, Pathan et al. [189] proposed an architecture that utilises software sensors to capture the user context in which the model is enabled. The proposal is not only able to deal with an entity (a person or an object) performing an action, but also considers relations among entities (e.g. a student behaviour in a classroom). The approach presented by Cole et al. [68] suggests that measures of physiological data (e.g., eye movement) are sources of information of (at least) three types of cognitive information: semantics, attention, and decision making. These claims have been supported by recent applications in portable devices involving eye-tracking recognition. Another proposal for recognising user context information is given by Belázquez et al. [32]. In their proposal the authors rely on the acquisition of user context from two classes of sensors: *hard sensors* corresponding to sensors built in mobile devices, and *soft sensors* providing information from social network sites to which a user is subscribed.

It is possible to have information in an event context type that is different from the information in the user profile for that context type. For example, suppose that a user has a “beginner” level of *skills* represented in his/her user model, but after using the system for a certain time, an event representing a change of context for this user occurs. Suppose the new context type to be an “expert” level of *skills*. The framework assumes the information of an event context type as the most up to date context type, even when it is different from the information in the user model for that context type. In the above example, the level of *skills* of the user will be considered as “expert” because of the event. The information in the event will be used to update the user model. In the case in which the information for certain context types is not represented in an event, the approach uses the user model to complement the context types of a user. This is explained in what follows.

For a certain user U_i interacting with the system, the *Rule Adaptor* identifies the relevant context patterns based on the event context type and the information in the model of user U_i. For each user U_i, the framework considers all the different context types used in our approach for the user that are defined in an event or in the user model. When a context type is not defined for user U_i, the context type is not considered in the process. In some cases, it is necessary to complement the characteristics of a user with information in the user model to support the identification of patterns that combine different context types and to provide more information about the context of a user. For example, suppose the case in which the user, Mary, is accessing the system with the *role* of “personal manager” (see Section 3.5). Assume the event context type specifying only the *role* of Mary (i.e. “personal manager”). In this case, the framework will use other information specified in Mary’s user model for other context types of Mary (e.g. *skills* and *cognition* as specified in the model), if such information exists. The context patterns are identified from the rule pattern repository (see Figure 3.5). The approach also assumes the existence of a monitor rule repository $MR\_SBS_j(U_i)$ for every user U_i of a service-based system SBS_j.

After identifying the relevant context patterns, the adaptation process *i)* creates semi-instantiated patterns, *ii)* updates the repository of monitor rules, or *iii)* selects the monitor rules that are relevant for a user in a service-based system. The step concerned with the creation of semi-instantiated patterns involves the use of the BPEL service-base system specification and *annotation* files (see Section 5.2). The step concerned with updating the monitor rule repository and selecting the relevant rules for a user involves the activities of identifying, creating, modifying, and removing monitor rules. The identification, modification, and removal of monitor rules for a user U_i are executed by analysing already existing monitor rules in the repository in order to identify if a relevant rule *(a)* already exists in the repository and *(a.1)* can be used as it stands, *(a.2)* needs to be modified, or *(a.3)* needs to be removed. In the case in which a relevant rule does not exist in the repository, or the repository is empty, new rules are created. In the next subsections we explain the *annotations*, the process dealing with the creation of semi-instantiated patterns, the update of the monitor rules repository and the selection of the relevant monitor rules (steps *i*, *ii*, and *iii* above).

5.2 Annotations

The use of annotation to support different activities in service-based systems has been advocated by several authors [51][82][85][148]. For example, in the approach proposed by Bucchiarone et al. [51] service operations are annotated with context effects indicating the impact of the execution of a service operation due to context changes to support adaptation of service compositions. In order to support process level selection of service, the work of Pietro et al. [82] uses semantic annotations of BPEL and WSDL specifications. Similarly, the work by Eberle et al. [85] annotates groups of activities in a business process with constraints and constraints handling capabilities. The approach developed by Le et al. [148] annotates BPEL specifications to support request of services. In our framework, we also use annotations to support the description of the parts of a service-based system specification that are concerned with the various context types. As in the case of the above approaches, in our framework, service-based system designers specify the annotations based on the requirements of the system. Moreover, annotations can be changed due to changes in the requirements of a system (e.g., a new actor with a different role type can use the system, new functionalities are available to the system, or certain functionalities are removed from the system), or changes in the service-based system due to services that become unavailable or new better services that are created.

Field	Meaning
Name	Specifies the context type
Value	Specifies the value for the context type
startPointTag	Specifies the type of tag used as a starting point in the BPEL specification
nameStartPoint	Specifies the name of the startPointTag in the specification
endPointTag	Specifies the type of tag used as a ending point in the BPEL specification
nameEndPoint	Specifies the name of the endPointTag in the specification

Table 5.1: General structure of an annotation

In the framework an *annotation* represents the parts of the specifications that are related to context types. *Annotations* are specified in separate files in order to preserve the original BPEL specification. They are linked to the BPEL specification through XPath expressions [67]. The framework assumes annotations specified by the designer of the service-based system based on the requirements of the system. Table 5.1 shows a general structure for the annotations used in the framework. An annotation represents *i)* a specific context type and its instance and *ii)* the part in the BPEL specification related to the instance of the context type. As shown in Table 5.1, the context type and its instance are represented in the fields *Name* and *Value*, respectively. The fields *startPointTag* and *endPointTag* specify the element types (tags) in the BPEL specification composing the part of the specification related to the context type, while the fields *nameStartPoint* and *nameEndPoint* specify the content of the elements represented in the *startPointTag* and *endPointTag*, respectively.

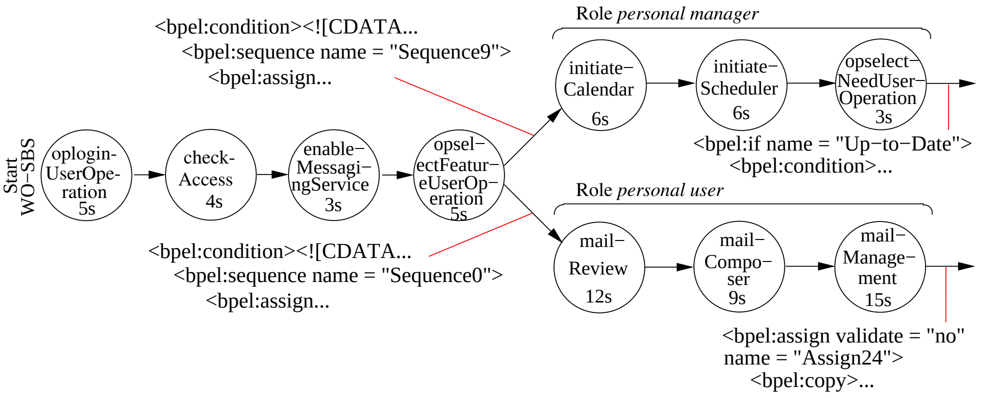

Start WO−SBS
oplogin-UserOpe-ration 5s
check-Access 4s
enable−Messagi−ngService 3s
opsel−ectFeatur−eUserOp−eration 5s
Role *personal manager*
initiate−Calendar 6s
initiate−Scheduler 6s
opselect−NeedUser−Operation 3s
Role *personal user*
mail−Review 12s
mail−Compo−ser 9s
mail−Manage−ment 15s
<bpel:condition><![CDATA...
<bpel:sequence name = "Sequence9">
<bpel:assign...
<bpel:if name = "Up−to−Date">
<bpel:condition>...
<bpel:condition><![CDATA...
<bpel:sequence name = "Sequence0">
<bpel:assign...
<bpel:assign validate = "no"
name = "Assign24">
<bpel:copy>...

Figure 5.1: Extract of the BPEL process for “personal user” and “personal manager”

```
Annotation 1 (role ”personal user”)
<ContextType name  =”Role”  value =”Personal User” startPointTag  =”//bpel:sequence”
        nameStartPoint  =”//*[contains(name(),’Sequence0’)]”
        endPointTag  =”//bpel:assign”
        nameEndPoint  =”//*[contains(name(),’Assign24’)]” />

Annotation 2 (role ”personal manager”)
<ContextType name  =”Role”  value  =”Personal Manager”  startPointTag  =”//bpel:sequence”
        nameStartPoint  =”//*[contains(name(),’Sequence9’)]”
        endPointTag  =”//bpel:if”
        nameEndPoint  =”//*[contains(name(),’Up-to-Date’)]” />
```

Figure 5.2: Example annotations

As an example, consider the extract from the BPEL process depicted in Figure 5.1. The figure depicts the parts of the code of the BPEL specification, corresponding to different actions and control activities (e.g. “bpel:sequence”, “bpel:assign”). The annotations relating the two *roles* (“personal user” and “personal manager”) shown in Figure 5.1 are depicted in Figure 5.2. As shown in Figure 5.2, the *Name* field is *role*, and the *Value* fields are “personal user” for annotation 1 and “personal manager” for annotation 2. For annotation 1, fields *startPointTag* and *nameStartPoint* are “bpel:sequence” and “Sequence0”, respectively; and *endPointTag* and *nameEndPoint* are “bpel:assign” and “Assign24”, respectively. For annotation 2 the *startPointTag* and *nameStartPoint* are “bpel:sequence” and “Sequence9”, respectively, while *endPointTag* and *nameEndPoint* are “bpel:if” and “Up-to-Date. For simplicity we refer to the node in the BPEL specification related to the *startPointTag* and the *nameStartPoint* as *starting point*. Similarly we refer to the node related to the *endPointTag* and the *nameEndPoint* as the *ending point*.

The *Path Identifier* (see Section 3.8) relies on the use of the annotations for the identification of the operations comprised in the part of the SBS specification relevant to a specific instance of a user context type, i.e. operations comprised within the *starting* and the *ending points*. In the case there is no match between the instance of a user context type and the context values specified in the annotations, no operations are identified.

The *Path Identifier* has been implemented as a variation of the deep-first search (DFS) algorithm [224]. It starts from the root of the BPEL specification, and traverses each branch searching for a *starting* and *ending points* that match the instantiated user contexts. The search is optimised; more specifically, in the case a context defined for a user matches the context type of a *starting point* but there is a mismatch in the instance of the user context and the value defined in the annotations, the *Path Identifier* quits the search in that branch and traverses the next one.

The separation between the BPEL specification and annotations allows our framework to deal with modifications related to the system, seamlessly. This is, monitor rules are able to be identified, modified, created, and removed even when services are substituted or changes occur in the workflow of the process. The above holds as long as *i)* the *starting* and *ending points* are not affected by the modifications and *ii)* the semantic of the annotated part in the BPEL specification remains related to the same context and its value after modification.

5.3 Creation of Semi-instantiated Patterns

The patterns used in the framework are generic context patterns and are not concerned with a specific application. However, the monitor rules need to be defined for specific operations, or sets of operations, in a service-based system application. It is not possible to know in advance, which are the relevant operations in an application that need to be used in the monitor rules. Therefore, the framework needs to use information from the SBS specification to *instantiate* the relevant context patterns and obtain the monitor rules for an application.

In the framework, we use the concept of *semi-instantiated* patterns representing context patterns in which the events and fluents are instantiated with the respective information from the SBS specification. As explained in Section 3.8, the instantiation of the pattern with the respective information is performed by the *Rule Adaptor* with the support of the *Path Identifier*. The approach assumes that the annotations are correct for a certain context type instance. When no annotation is specified for a certain context type, the patterns cannot be instantiated.

As an example, consider the extract of the BPEL process depicted in Figure 5.1 and assume a user, *Mary*, accessing the Wo-SBS in the role of a personal user. In this case, the context type (role) and its instance "personal user" match the context type and value in annotation1 in Figure 5.2. The associated annotation includes operations *mailReview*, *mailComposer*, and *mailManagement* corresponding to the path between startPointTag and endPointTag. Table 5.2 shows an example of a semi-instantiated pattern for this situation based on the pattern previously described in Table 4.7.

Rule part
$Happens(ic\_startWoSBS, t_1, R(t_1, t_1)) \Rightarrow$ $Happens(ic\_mailReview, t_2, R(t_1, t_1 + t_{n1})) \wedge$ $Happens(ic\_mailComposer, t_3, R(t_2, t_1 + t_{n2})) \wedge$ $Happens(ic\_mailManagement, t_4, R(t_3, t_1 + t_{n3}))$

Assumption part
$Happens(ic\_mailReview, t_n, R(t_n, t_n)) \Rightarrow$ $Initiates(ic\_mailReview, mailReview, t_n)$
$Happens(ic\_mailComposer, t_n, R(t_n, t_n)) \Rightarrow$ $Initiates(ic\_mailComposer, mailComposer, t_n)$
$Happens(ic\_mailManagement, t_n, R(t_n, t_n)) \Rightarrow$ $Initiates(ic\_mailManagement, mailManagement, t_n)$

Table 5.2: Example of Semi-instantiated Pattern for Role Context Type

The proposed context patterns previously described in Chapter 4 are quite general. This creates difficulties to identify the corresponding monitor rules in a repository relying solely on the patterns. The use of the *semi-instantiated patterns* solves this problem; they reduce the generalisation by creating patterns concerned with a certain set of service operations related to a SBS application. For example, consider an extract of the Wo-SBS with the respective services and operations for roles "personal user" and "personal manager" shown in Figure 5.1. Suppose a user in the role of "personal user" accessing the system and the pattern for role context type shown in Figure 4.7. Assume a rule repository with monitor rules R_1, R_2, and R_3 depicted in Figure 5.3 (without the assumptions for simplicity). If the approach relies solely on the role pattern for the identification of the correct monitor rules, all the three rules in Figure 5.3 will be identified since they all match the *invariant* part of the pattern. However, only monitor rule R_3 is concerned with operations related to role "personal user". Instead, if the semi-instantiated pattern shown in Figure 5.2 is used,only rule R_3 will be identified since its events match the events in the semi-instantiated pattern. Overall, the semi-instantiated patterns are important to support the activities of identifying, modifying, and creating monitor rules.

R_1	$Happens(ic\_startWoSBS, t_1, R(t_1, t_1)) \Rightarrow$ $Happens(ic\_initiateCalendar, t_2, R(t_1, t_1 + 17.05)) \wedge$ $Happens(ic\_initiateScheduler, t_3, R(t_2, t_1 + 23.06)) \wedge$
R_2	$Happens(ic\_startWoSBS, t_1, R(t_1, t_1)) \Rightarrow$ $Happens(ic\_dummyOperation, t_2, R(t_1, t_1 + 100))$
R_3	$Happens(ic\_startWoSBS, t_1, R(t_1, t_1)) \Rightarrow$ $Happens(ic\_mailReview, t_2, R(t_1, t_1 + 17.05)) \wedge$ $Happens(ic\_mailComposer, t_3, R(t_2, t_1 + 29.06)) \wedge$ $Happens(ic\_mailManagement, t_4, R(t_3, t_1 + 38.07))$

Table 5.3: Monitor Rules Repository Consisting of Rules R_1, R_2, and R_3

An important aspect regarding the creation of semi-instantiated patterns is that their generation is not restricted to a single pattern for a context type. As mentioned in Chapter 4, sets of patterns (which are not intended to be complete) have been created for each one of the user context type. Every time a context is defined for a user, and the part of the service composition related to that context is identified, all the patterns defined for the that context are semi-instantiated. The main benefits of having a set of semi-instantiated pattern instead of a single one include:

- A broad covering. Having different semi-instantiated patters increases the chances of finding existing rule(s) in the repository, e.g. defined manually by a designer, matching the semi-instantiated patterns.

- Flexibility when verifying a property. Monitor rules can verify the behaviour of the system according to different events, states, from a global (i.e. all events/states in a single rule), or individual (i.e. events/states in a single rule) perspective, or even considering different checkpoints (i.e. focusing on different part of the system). Having different ways of specifying monitor rules (syntaxes and semantics) for a context property, allows for a broad coverage of the property.

- Expansion. Given the set of patterns is not complete, it is plausible to assume that more context patterns can be defined for the different context types. Allowing several instantiations of context patterns for a context type leads to a seamless integration of the potential new patterns in our approach, without modification of the existing ones.

5.4 Rules Identification, Creation, Modification and Removal

After creating the *semi-instantiated patterns*, they are compared with existing monitor rules in the repository in order to identify if a relevant rule *(a)* exists in the repository and can be used as it stands, *(b)* exists in the repository and needs to be modified, *(c)* needs to be created and added to the repository, or *(d)* exists in the repository and needs to be removed.

Figure 1 presents an algorithm in pseudo-code for the process of selecting and updating monitor rules in repositories. As shown in Figure 1, the process consists of searching in the repository for monitor rules that match *semi-instantiated patterns*. In the case in which there are monitor rules in the repository that match the *semi-instantiated patterns* (i.e. the monitor rules matching *predicates*, *fluents*, and *events* of the *semi-instantiated pattern*), the process verifies whether the time values in the rules are consistent with the times specified in the SLAs or historical data for the respective operations and services. In positive case, the rules are maintained in the repository. Otherwise, the rules are modified with new time values according to the information in the SLAs (or historical data).

Algorithm 1: Adaptation Process

```
Data: $SI_{Rule}$ semi-instantiated pattern
Data: $SBS_{Spec}$ service-based system specification
Data: $SLAs$ SLAs for the services and operations
Data: $R_{Rep}$ rules repository
begin
    Search $SI_{Rule}$ in $R_{Rep}$;
    /* Match invariant, events and fluents */
    if $R_{Rep}$ has rules that fully match $SI_{Rule}$ then
        for $R \in R_{Rep}$ do
            if time in $SLAs$ is within time values in $R$ then
                do nothing;
            else
                Adjust time in $R$ based on $SLAs$;
    else
        /* Match invariant */
        if $R_{Rep}$ has Rules that only match invariant parts of $SI_{Rule}$ then
            for $R \in R_{Rep}$ do
                if There is a path in $SBS_{Spec}$ that uses $R$ then
                    /* Rule R is not obsolete */
                else
                    Remove $R$ from $R_{Rep}$;
    else
        /* No match in invariant */
        if $R_{Rep}$ has no rule matching the invariant parts of $SI_{Rule}$ then
            Create $I_{Rule}$ by instantiating $SI_{Rule}$ time with times in $SLAs$ or
            historical data;
            Add $I_{Rule}$ to $R_{Rep}$;
end
```

In the situation in which there are rules in the repository that match only the invariant parts of the *semi-instantiated patterns* (i.e. the monitor rules that match predicates in the semi-instantiated patterns, but that do not match *fluents* and *events* in the *semi-instantiated pattern*) the process verifies if they are still valid monitor rules for the service-based system. The verification of the validity of the matched monitor rules is executed by the *Rule Verifier*. It analyses if operations specified in the rules correspond to valid operations in the current specification of the service-based system. This is executed by traversing the service-based system specification and identifying if the operations are used in the specifications. In positive case, the monitor rules are kept in the repository. In negative case, the monitor rules are removed from the repository.

In the case in which none of the rules in the repository match the variant or invariant parts of the semi-instantiated patterns, new rules are created after instantiating the semi-instantiated patterns with the time values computed from the times specified in the SLAs (or historical data). The computation of the times values depend on the context type of the pattern. For example, in the case of a role context type, the times are computed considering the necessary times to execute previous operations in the workflow. In the case of cognition context type, the times are computed considering the times of the users operations. In the case of skills context type, the times are computed based on the time to execute the part of the workflow relevant to the pattern.

In order to illustrate the identification, modification, creation, and removal of monitor rules consider the extract of Wo-SBS scenario depicted in Figure 5.1. Assume a user, Mary, accessing the system. For simplicity, assume that there is no information about the Mary's context in the user model. Finally, consider a Rule Repository containing the monitor rules depicted in Table 5.4. Note that the set of monitor rules in the repository consists of monitor rules (indexed by a R_i) and assumptions (indexed by an A_i).

Case 1 - Rule Identification: Assume Mary accessing the system in the role of a “personal user”. Our framework receives the role context from Mary and, since

R_1	$Happens(ic\_startWoSBS, t_1, R(t_1, t_1)) \Rightarrow$ $Happens(ic\_initiateCalendar, t_2, R(t_1, t_1 + 17.05))$
R_2	$Happens(ic\_startWoSBS, t_1, R(t_1, t_1)) \Rightarrow$ $Happens(ic\_initiateScheduler, t_2, R(t_1, t_1 + 23.06))$
R_3	$Happens(ic\_startWoSBS, t_1, R(t_1, t_1)) \Rightarrow$ $Happens(ic\_initiateCrossReferencing, t_2, R(t_1, t_1 + 71.12))$
R_4	$Happens(ic\_startWoSBS, t_1, R(t_1, t_1)) \Rightarrow$ $Happens(ic\_mailReview, t_2, R(t_1, t_1 + 17.05)) \wedge$ $Happens(ic\_mailComposer, t_3, R(t_2, t_1 + 29.06)) \wedge$ $Happens(ic\_mailManagement, t_4, R(t_3, t_1 + 38.07))$
A_1	$Happens(ic\_initiateCalendar, t_1, R(t_1, t_1)) \Rightarrow$ $Initiates(ic\_initiateCalendar, initiateCalendar, t_1)$
A_2	$Happens(ic\_initiateScheduler, t_1, R(t_1, t_1)) \Rightarrow$ $Initiates(ic\_initiateScheduler, initiateScheduler, t_1)$
A_3	$Happens(ic\_initiateCrossReferencing, t_1, R(t_1, t_1)) \Rightarrow$ $Initiates(ic\_initiateCrossReferencing, initiateCrossReferencing, t_1)$
A_4	$Happens(ic\_mailReview, t_1, R(t_1, t_1)) \Rightarrow$ $Initiates(ic\_mailReview, mailReview, t_1)$
A_5	$Happens(ic\_mailComposer, t_1, R(t_1, t_1)) \Rightarrow$ $Initiates(ic\_mailComposer, mailComposer, t_1)$
A_6	$Happens(ic\_mailManagement, t_1, R(t_1, t_1)) \Rightarrow$ $Initiates(ic\_mailManagement, mailManagement, t_1)$

Table 5.4: Initial set of Monitor Rules Repository in the Repository

there is no information about other context types for Mary in her user model, the framework identifies the rule pattern for role [1] and semi-instantiates it with the operations in the part of the SBS related to personal user, see Table 5.2. The semi-instantiated pattern is compared against the rules in the repository and a match with rule R_4, and assumptions A_4, A_5, and A_6 in Table 5.4 is found. The time constraints for rules and assumptions are checked against computed time constraints. In this case, it is not necessary to change the times. The rule and assumptions are identified as a suitable for monitoring Mary in the role of a "personal user". The monitor rules R_1-R_3 and the assumptions A_1-A_3 in the repository match the invariant parts of the

[1] Note that for the sake of the example we considered only the role pattern depicted in Table 4.7. This is valid for all four cases

role pattern. The process verifies that these rules are still valid rules in Wo-SBS and, since they belong to valid operations in the system, they are kept in the repository. The repository remains unmodified.

Case 2 - Rule Modification: Consider again Mary accessing the Wo-SBS in the context of a "personal user". Assume that operation *mailReview* requires nine seconds to be executed instead of its original 12 seconds (see Figure 5.1). In this case, the semi-instantiated pattern in Table 5.2 is matched with the rule R_4 and assumptions A_4-A_6 in the repository. The time constraints for rules and assumptions are checked against computed time constraints. In this case, it is necessary to change the time constraints in R_4. The process updates rule R_4 according to the new time constraints, as shown in Figure 5.5. As in the previous case, the remaining monitor rules R_1-R_3 and assumptions A_1-A_3 are kept in the repository.

Case 3 - Rule Creation: Consider Mary accessing the Wo-SBS in the *role* of a "personal user", with *medium* level of *skills*. The framework identifies the rule patterns for *role* (Table 4.7) and skills (Table 4.23) and semi-instantiates the patterns with the corresponding operations, as shown in Table 5.6. In the figure, SIP_1 is the semi-instantiated rule part of the pattern for the *role* context type. SIP_2 is the semi-instantiated rule part of the pattern for the *skills* context type. SIP_3-SIP_6 are the semi-instantiated assumptions parts of the *role* and *skills* patterns. The semi-instantiated patterns in Table 5.6 are compared against the existing rules in the repository (Table 5.5).

The semi-instantiated patterns SIP_1, SIP_3, SIP_4, SIP_5 match R_4, A_4, A_5, and A_6 in the repository. Assume that the time constraints in R_4 and A_4-A_6 are also correct (rules are identified). For the semi-instantiated parts of the pattern SIP_2 and SIP_6 however, there is no match in the repository. The time values in SIP_2 and SIP_6 are

R_1	$Happens(ic\_startWoSBS, t_1, R(t_1, t_1)) \Rightarrow$ $Happens(ic\_initiateCalendar, t_2, R(t_1, t_1 + 17.05))$
R_2	$Happens(ic\_startWoSBS, t_1, R(t_1, t_1)) \Rightarrow$ $Happens(ic\_initiateScheduler, t_2, R(t_1, t_1 + 23.06))$
R_3	$Happens(ic\_startWoSBS, t_1, R(t_1, t_1)) \Rightarrow$ $Happens(ic\_initiateCrossReferencing, t_2, R(t_1, t_1 + 71.12))$
R_4	$Happens(ic\_startWoSBS, t_1, R(t_1, t_1)) \Rightarrow$ $Happens(ic\_mailReview, t_2, R(t_1, t_1 + 17.05)) \wedge$ $Happens(ic\_mailComposer, t_3, R(t_2, t_1 + 26.06)) \wedge$ $Happens(ic\_mailManagement, t_4, R(t_3, t_1 + 35.07))$
A_1	$Happens(ic\_initiateCalendar, t_1, R(t_1, t_1)) \Rightarrow$ $Initiates(ic\_initiateCalendar, initiateCalendar, t_1)$
A_2	$Happens(ic\_initiateScheduler, t_1, R(t_1, t_1)) \Rightarrow$ $Initiates(ic\_initiateScheduler, initiateScheduler, t_1)$
A_3	$Happens(ic\_initiateCrossReferencing, t_1, R(t_1, t_1)) \Rightarrow$ $Initiates(ic\_initiateCrossReferencing, initiateCrossReferencing, t_1)$
A_4	$Happens(ic\_mailReview, t_1, R(t_1, t_1)) \Rightarrow$ $Initiates(ic\_mailReview, mailReview, t_1)$
A_5	$Happens(ic\_mailComposer, t_1, R(t_1, t_1)) \Rightarrow$ $Initiates(ic\_mailComposer, mailComposer, t_1)$
A_6	$Happens(ic\_mailManagement, t_1, R(t_1, t_1)) \Rightarrow$ $Initiates(ic\_mailManagement, mailManagement, t_1)$

Table 5.5: Monitor Rules after Modification in R_4

instantiated (see Table 5.7) and added to the repository as R_5 and A_7. Note that in the case of the assumptions, i.e. SIP_3-SIP_6, the time values appear as defined (t_1). This is due to the fact that in the patterns, the assumptions triggering the initialisation of a fluent are assumed to occur at a single instant (t_1), hence no time computation is needed when dealing with these kind of assumptions.

Case 4 - Rule Removal: Suppose the situation in which the Wo-SBS has been modified and it does not longer support the user of type "event coordinator". Assume that the activity concerned with cross-referencing (operation *initiateCrossReferencing*) is no longer supported by the Wo-SBS. Consider again Mary accessing the Wo-

SIP_1	$Happens(ic\_startWoSBS, t_1, R(t_1, t_1)) \Rightarrow$ $Happens(ic\_mailReview, t_2, R(t_1, t_1 + t_{n1}))\wedge$ $Happens(ic\_mailComposer, t_3, R(t_2, t_1 + t_{n2}))\wedge$ $Happens(ic\_mailManagement, t_4, R(t_3, t_1 + t_{n3}))$
SIP_2	$Happens(ic\_startWoSBS, t_1, R(t_1, t_1))\wedge$ $Happens(ir\_exit, t_3, R(t_1, t_1 + t_n)) \Rightarrow$ $Happens(ic\_opstandardUIMailUserOperation, t_2, R(t_1, t_3))$
SIP_3	$Happens(ic\_mailReview, t_1, R(t_1, t_1)) \Rightarrow$ $Initiates(ic\_mailReview, mailReview, t_1)$
SIP_4	$Happens(ic\_mailComposer, t_1, R(t_1, t_1)) \Rightarrow$ $Initiates(ic\_mailComposer, mailComposer, t_1)$
SIP_5	$Happens(ic\_mailManagement, t_1, R(t_1, t_1)) \Rightarrow$ $Initiates(ic\_mailManagement, mailManagement, t_1)$
SIP_6	$Happens(ic\_opstandardUIMailUserOperation, t_1, R(t_1, t_1)) \Rightarrow$ $Initiates(ic\_opstandardUIMailUserOperation,$ $opstandardUIMailUserOperation, t_1)$

Table 5.6: Semi-instantiated Patterns for *role* and *skills*

R_5 R_5	$Happens(ic\_startWoSBS, t_1, R(t_1, t_1))\wedge$ $Happens(ir\_exit, t_3, R(t_1, t_1 + 61.09)) \Rightarrow$ $Happens(ic\_opstandardUIMailUserOperation, t_2, R(t_1, t_3))$
A_7	$Happens(ic\_opstandardUIMailUserOperation, t_1, R(t_1, t_1)) \Rightarrow$ $Initiates(ic\_opstandardUIMailUserOperation,$ $opstandardUIMailUserOperation, t_1)$

Table 5.7: Instantiated *role* and *skills* rules

SBS in the *role* of a "personal user" with a medium level of *skills*. In this case, R_4, R_5, A_4-A_7 are identified in the repository as matching the semi-instantiated patterns. Rules and assumptions (see Table 5.5) are identified as matching the invariant part of the role pattern. However, the operation *initiateCrossReferencing*, appearing in the rule part R_3 and in the assumption part A_3, is no longer part of the Wo-SBS, hence, they are removed from the repository. The resulting repository is shown in Table 5.8.

Overall, the automation of the monitor process improves the specification of monitor rules and avoids, at the same time, faults prone to human participation. For illustration purposes a set monitor rules specified according to the monitor component

R_1	$Happens(ic\_startWoSBS, t_1, R(t_1, t_1)) \Rightarrow$ $Happens(ic\_initiateCalendar, t_2, R(t_1, t_1 + 17.05))$
R_2	$Happens(ic\_startWoSBS, t_1, R(t_1, t_1)) \Rightarrow$ $Happens(ic\_initiateScheduler, t_2, R(t_1, t_1 + 23.06))$
R_4	$Happens(ic\_startWoSBS, t_1, R(t_1, t_1)) \Rightarrow$ $Happens(ic\_mailReview, t_2, R(t_1, t_1 + 17.05)) \wedge$ $Happens(ic\_mailComposer, t_3, R(t_2, t_1 + 26.06)) \wedge$ $Happens(ic\_mailManagement, t_4, R(t_3, t_1 + 35.07))$
R_5	$Happens(ic\_startWoSBS, t_1, R(t_1, t_1)) \wedge$ $Happens(ir\_exit, t_3, R(t_1, t_1 + 61.09)) \Rightarrow$ $Happens(ic\_opstandardUIMailUserOperation, t_2, R(t_1, t_3)) \wedge$
A_1	$Happens(ic\_initiateCalendar, t_1, R(t_1, t_1)) \Rightarrow$ $Initiates(ic\_initiateCalendar, initiateCalendar, t_1)$
A_2	$Happens(ic\_initiateScheduler, t_1, R(t_1, t_1)) \Rightarrow$ $Initiates(ic\_initiateScheduler, initiateScheduler, t_1)$
A_4	$Happens(ic\_mailReview, t_1, R(t_1, t_1)) \Rightarrow$ $Initiates(ic\_mailReview, mailReview, t_1)$
A_5	$Happens(ic\_mailComposer, t_1, R(t_1, t_1)) \Rightarrow$ $Initiates(ic\_mailComposer, mailComposer, t_1)$
A_6	$Happens(ic\_mailManagement, t_1, R(t_1, t_1)) \Rightarrow$ $Initiates(ic\_mailManagement, mailManagement, t_1)$
A_7	$Happens(ic\_opstandardUIMailUserOperation, t_1, R(t_1, t_1)) \Rightarrow$ $Initiates(ic\_opstandardUIMailUserOperation,$ $opstandardUIMailUserOperation, t_1)$

Table 5.8: Monitor Rules after Removal

can be found in [119].

5.5 Remarks

It is important to note that although the previous cases showed the identification, creation, modification, and removal for particular context type patterns, several monitoring patterns can be instantiated for monitoring the same operation(s). More specifically, for a given workflow, the framework identifies the related context(s), and applies the set of patterns that are related to the context(s) (as explained in section 5.3).

As a result, several monitor rules are generated and deployed in the monitor component. Having several monitor rules verifying a context property assures the property is verified from different perspectives. For example, while a single monitor rule verifying the invocation of a set of two or more operations at once, can be violated; monitoring the same set of operations separately (i.e. a single monitor rule per operation), can help identifying whether the violation occurred because of the malfunction of specific operations. In addition to the latter, monitor rules can be concerned with the execution of the whole system, or just a part of it; again, it might be possible for one rule to be violated while the other holds.

5.6 Summary

In this chapter we focus on the adaptation process. The process relies on the use of a non-intrusive annotation strategy, so relations can be made between specific context type values and parts of the BPEL process without modifying the original specification. The framework relies on the use of patterns, previously introduced in chapter 4, concerned with different user context types. Patterns are instantiated with relevant information from the associated part of the service composition, in order to obtain templates of monitor rules, i.e. semi-instantiated patterns, which are used by the framework for the identification, creation, modification, and removal of monitor rules from a repository.

In the following chapter we present the results of the experiments conducted in our framework. More specifically, we show how our framework performs in the activities regarding the identification, creation, modification, and removal of monitor rules. We examine several different configurations in order to stress all the possible cases for each activity.

Chapter 6

Experiments and Evaluation

In this chapter we present the results of the experiments conducted to evaluate our framework. A prototype tool of the framework, previously described in Chapter 5, has been implemented in Java and was integrated with the *monitor component* described in [220]. More specifically, we have implemented the *Rule Adaptor*, *Path Identifier*, and *Rule Verifier* components in Java and have deployed the specified monitor rules in the *monitor component*. The framework was evaluated in terms of its performance to execute its various activities and in terms of the correctness of the identified, modified, created, and removed rules, triggered by different events.

We evaluated the performance of the framework for the identification, creation, modification, and removal of monitor rules related to the various context types. We considered different sets of monitor rules in the rules repository. The context annotations, for each context type, were defined so they pointed at different parts in the service-based system specification. We also analysed the correctness of the monitor rules generated by the framework. The following section describes the setup of the experiments.

6.1 Experiments Setup

The evaluations were conducted on a 1.5GHz Core-2 Duo machine, with 2Gb of memory, running on a Linux (kernel 3.0.0-16 generic) using an extended version of the *Wo-SBS* application previously introduced (see Section 3.5). For comparative purposes we constructed a second application, an Air-traffic Control Service-based System (*Atc-SBS*), which supports different activities in an *airport* according to the specified user contexts. More specifically, the *Atc-SBS* supports the activities related to general airport operations, emergency, and assignment of resources for different types of users. Both applications have been specified in BPEL4WS [120] and can be found in Appendix A.In what follows we present the different services and operations along with the contexts instantiations used for each scenario.

6.1.1 Scenarios

The *Wo-SBS* is composed of eight services, with a total of 14 service operations, and 10 user operations. Table 6.1 summarises the services, services operations, and user operations in the *Wo-SBS*. The user operations in Table 6.1 have the prefix "op" and the suffix "UserOperation". Table 6.2 summarises the various instances for the different context types used in the experiments.

The Air-traffic Control Service-based System (*Atc-SBS*) is composed of five services, with a total of 11 service operations, and 12 user operations. Table 6.3 summarises the services, operations, and user operations in the *Atc-SBS*. Table 6.4 summarises the various instances for the different context types used in the experiments.

The creation of a second scenario allows us to establish whether the performance of our framework is consistent. More specifically, we are interested in establishing whether there is a similarity in the performance for different scenarios, considering

Service	Operations
Agenda	*initiateCalendar*, *stopCalendar*, *getLatestDates*, *stopScheduler* *initiateScheduler*, *getLatestAnnotations*
Basic User Operations	*opselectMessagingUserOperation*, *opselectNeedUserOperation* *opselectMailExpertiseUserOperation*
Check Access	*accessChecker*
Extra	*initiateCrossReferencing*
Mail	*mailReview*, *mailComposer*, *mailManagement*
Messaging	*enableMessagingService*, *disableMessagingService*
GUI	*opAgendaUserOperation*, *opselectFeatureUserOperation*, *oploginUserOperation*, *opstandardUIMailUserOperation* *opadvancedMailUserOperation*, *opguiUserOperation* *opmailAndAgendaUserOperation*
Shut down	exit

Table 6.1: Services and operations in the Wo-SBS scenario

Context Type	Instances
Role	*"personal user"*, *"personal manager"*, *"event coordinator"*
Cognition	*"low"*, *"average"*, *"high"*
Skills	*"beginner"*, *"average"*, *"advanced"*
Preferences	*time constraints of "6" and "15" seconds*
Need	*"up-to-date"*, *"fast"*

Table 6.2: Instances for the different context types in the Wo-SBS scenario

the same amount of rules in the rules repository and types of contexts. Furthermore, considering two different scenarios permits us to establish whether the different context types behave similarly, i.e. their performance. The two scenarios consider similar processes structures, see [119], and are annotated in different parts of the BPEL4WS specifications. Also, as it can be observed in Tables 6.1 and 6.3, the two scenarios differ in the amount of the participating services, as well as in the amount and ratio of automatic operations and operations involving the user participation.

Service	Operations
Radar	*opreqTrajectoryUserOperation*, *opinCoordinatesUserOperation*, *opreqCoordinatesUserOperation*, *getPositioning*, *opinDestinyUserOperation*, *opinDepartureUserOperation*
GUI and Access	*oprequestSpaceUserOperation*, *opselectionUserOperation*, *generalLayout*, *exit*, *oploginUserOperation*
Status	*hangarsStat*, *controlTowersStat*, *terminalsStat*, *takeoffsStat*, *landingsStat*, *runwaysStat*
Resources	*oprequestResourcesUserOperation*, *availableResources*, *opdistributionAvailableResourcesUserOperation*, *updateResources*
Scheduler	*opnormalTaskOrganiserUserOperation*, *opemergencyTaskOrganiserUserOperation*

Table 6.3: Services and operations in the Atc-SBS scenario

Context Type	Instances
Role	*"pilot"*, *"regular maintainer"*, *"emergency operator"*
Cognition	*"low"*, *"average"*, *"high"*
Skills	*"average"*, *"advanced"*
Preferences	*time constraints of "4" and "8" seconds*
Need	*"critical"*

Table 6.4: Instances for the different context types in the Atc-SBS scenario

6.1.2 Patterns and Repositories

In the experiments we evaluated the performance of the adaptation activities, i.e. identification, modification, creation, and removal of monitor rules, for the various context types under different configurations[1]. More specifically, for each service-based system we considered one context pattern, and a variable set of monitor rules ranging from an empty repository, to repositories with 100, 200, and 300 monitor rules.

The reason for considering a specific pattern for a context type, is due to the fact that different patterns for the same context type may imply a different amount of

[1] Experiments were conducted, separately, for the two scenarios: *Atc-SBS* and *Wo-SBS*.

operations to be performed by our framework. In addition to the above, the amount of resulting monitor rules, for the same context annotation, may also change from one pattern to another. For example, assume an empty repository and an annotation encompassing ten *user operations* in the service specification. Using the *skills* pattern depicted in Table 4.23 would result in the creation of ten different monitor rules, one per user operation. Similarly the *skills* pattern depicted in Table 4.21, is also concerned with the invocations of *user operations*, the pattern however, considers all the invocations of *user operations* at once. As a result using the pattern depicted in 4.21 generates a single monitor rule instead of ten.

We were also concerned with the effect that existing monitor rules in the repository may have on the performance of the adaptation process when executing the various adaptation activities. We considered variable sets of monitor rules (empty, 100, 200, and 300) for each context type. For each non-empty set of monitor rules we specified 20% of the rules, as monitor rules related to the specific context type pattern. More specifically, 20% of the monitor rules matched the invariant part of the context pattern. The remaining 80% of the rules were specified as not relevant to the context type, i.e. monitor rules concerned with other behavioural aspects of the application.

Experiments were conducted for a range of one to ten monitor rules for the different context types and activities in the adaptation process. We also considered, for each service-based system specification, a range of one to ten different annotations for each context type, i.e. one annotation per monitor rule. Our experience has shown that for a trigger situation of the monitor adaptation process, the creation, identification, modification, and removal of ten monitor rules is a high number and unlikely to happen in a normal situation. However, we used a maximum of ten monitor rules, to stress the cases. Each experiment was repeated 40 times. For each experiment we computed the *minimum*, *maximum*, *mean*, *median*, and the *standard deviation* values of the times required to identify, modify, create, or remove monitor rules. The tables with all the results for the two scenarios can be found in Appendix B.

We evaluated the different activities performed by the framework separately. For each activity - with the exception of rules identification - the rules repository was affected differently. More specifically, in the case of rules identification the repository remain the same. In the case of rules modification, the amount of monitor rules in the repository remained the same and a subset of monitor rules (from one to ten) was modified. In the case of rules creation the repository was incremented (from one to ten new monitor rules). Finally in the case of removal the repository was reduced (from one to ten monitor rules).

6.2 Experiments Results

The aims of the experiments are *i)* to analyse the performance of the framework for the different activities and context types for the two scenarios, *ii)* to analyse the performance of the framework when varying the amount of rules in the repository matching a context type pattern, *iii)* to analyse whether the annotations in the SBS specifications could affect the performance of the framework, and *iv)* to analyse the correctness of the framework.

The following subsections present the results of the different experiments. The patterns used in our experiments were the ones specified in Table 4.7 (*role*), Table 4.23 (*skills*), Table 4.36 (*cognition*), Table 4.50 (*preferences*), and Table 4.56 (*need*). In section 6.2.1 we present the results of the performance when identifying, creating, modifying, and removing monitor rules considering the various context types separately (*i)* above). In section 6.2.2 we show the performance of the framework when varying the ratio of monitor rules in the repository matching a pattern invariant (*ii)* above). In section 6.2.3 we explain how the annotations, more specifically, the part in the BPEL specification at which an annotation is pointing at, may influence the performance (*iii)* above). Finally in section 6.2.4 we analyse the correctness of the framework of the identified, modified, and created monitor rules (*iv)* above).

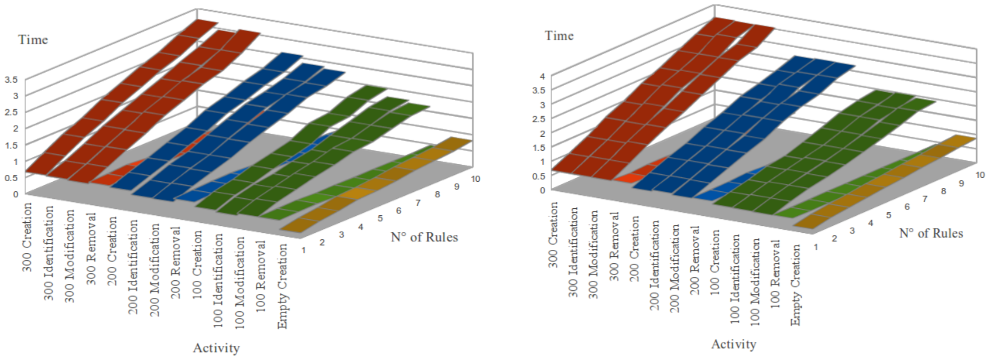
6.2.1 Performance of the Different Context Types

We evaluated the time that it takes the monitor adaptation process to execute each different activity for each direct user context type. The results of the experiment for context types *role*, *skills*, *cognition*, *preferences*, and *needs* are shown in Figures 6.1, 6.2, 6.3, 6.4, and 6.5 respectively. These figures show the performance in seconds in the 'Y' axis (*Time*), for all the different activities executed against various repositories sizes in the 'X' axis (*Activity*), considering a range from one to ten monitor rules in the 'Z' axis (*N^{o} of Rules*).

In the graphs the performance for the creation of monitor rules, considering an empty repository, are represented by a yellow curve. The performance for the removal, modification, identification, and creation of monitor rules, considering 100 rules in the repository, are represented by curves in green. Similarly the performance considering 200 and 300 rules in the repository are represented by curves in blue and red respectively. Note that in the case of an empty repository, only the creation of monitor rules is possible.

From the graphs showing the performance of the framework, we made the following inferences:

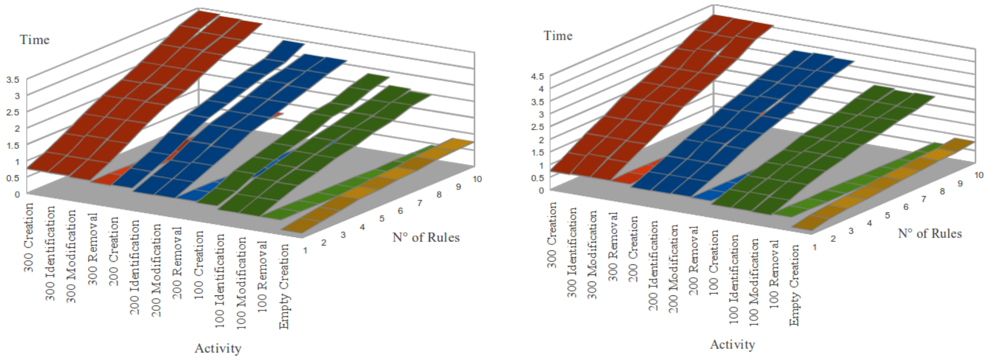
Despite the fact that the two scenarios (*Atc-SBS* and *Wo-SBS*) differed in size, services, amount of automated and user operations, and general structure; the performance results for the two scenarios were quite similar. In fact when considering the same context, rule pattern, amount of rules in the repository and activity (e.g. creation), the increment of the performance remained below the 15% from one scenario



(a) Atc-SBS scenario

(b) Wo-SBS scenario

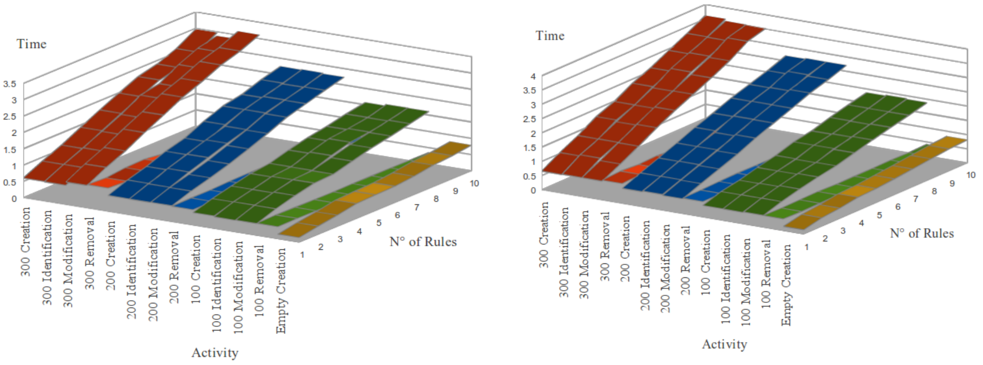
Figure 6.1: Results of the performance for the role context type



(a) Atc-SBS scenario

(b) Wo-SBS scenario

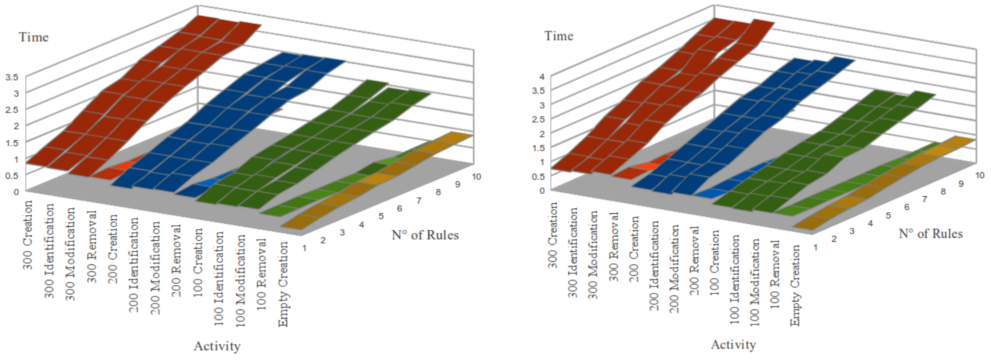
Figure 6.2: Results of the performance for the skills context type



(a) Atc-SBS scenario

(b) Wo-SBS scenario

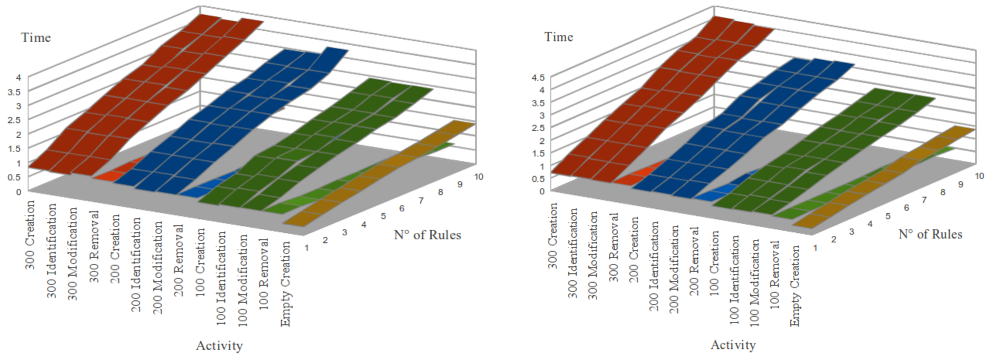
Figure 6.3: Results of the performance for the cognition context type



(a) Atc-SBS scenario

(b) Wo-SBS scenario

Figure 6.4: Results of the performance for the preferences context type



(a) Atc-SBS scenario

(b) Wo-SBS scenario

Figure 6.5: Results of the performance for the need context type

(*Atc-SBS*) to the other (*Wo-SBS*). Furthermore, the difference in the case of the creation, when considering an empty rules repository, dropped below ten percent. The fluctuations in the performance can be explained, in part, by *i)* the size of the repository and the amount of rules in it matching the invariant of the pattern and *ii)* the annotations, i.e. where the operations to be monitored appear in the BPEL specification. These aspects are further addressed in sections 6.2.2 and 6.2.3 respectively.

Although the performance for the different context types under the same configuration[2] remained quite similar, it was still possible to identify certain trends. More-

[2] This means the activity carried on, repository size and amount of rules considered in the experiment

over these trends were observed in the two scenarios. First, the performance for the different context types was quite similar for *cognition*, *role*, and *skills*. This can be explained by the similarity in the structure of the resulting rules and the required computations to generate those rules. Second, for the *need* context type, for all the different types of activities and different sizes of the repository, the amount of time required to perform a certain activity, was higher when compared to the amount of time required for the same activity and repository size for the other context types. In this case, the rules obtained from the *need* pattern are more complex - in terms of amount of predicates and computations - than the rules obtained for the other context types. Third, in several occasions the experiments related to the *preferences* context type required less time than the *need* context type and more than the *cognition*, *role* and *skills* context types. In this case the amount of predicates of the resulting rules was higher compared to rules generated for the *cognition*, *role*, and *skills* context types.

From the experiments we deduced that, in addition to the size of the repository and the annotations in the BPEL specification, the performance is also influenced by the structure of the pattern. The more complex the pattern, the higher amount of operations performed by the framework in order to establish whether a pattern is compatible with a monitor rule or not. More specifically, in order to establish whether a rule matches a pattern, the framework performs a set of comparisons, one per predicate. The higher the amount of comparisons the higher the time required by the framework to establish whether a rule matches a pattern. This observation was corroborated when comparing the performance of the different context types considering an empty repository.

Also regarding the performance, it is important to note that the arithmetic operations carried out to compute the time periods, including those operations required to compute the user interaction time - as in the *cognition* pattern - did not produce significant differences. For example, the change in the level of *cognition* of a user from "low" to "high"[3] did not affect the performance.

As mentioned before, in order to stress the scenarios, we conducted experiments considering from one to ten monitor rules for each context type, varying the sizes in the repositories. For the identification, creation, and modification, the increment in time, when augmenting in one the monitor rules and maintaining the size in the repository, bordered - in the worst cases - the 75%. The peaks in the performance were observed in the creation of monitor rules considering an empty repository. In fact, in those experiments considering non-empty repositories, the increment in the performance when augmenting in one the monitor rules, bordered - in the worst cases - the 35%. For the particular case of rules removal, i.e. considering non-empty repositories, an increment of 100 monitor rules in the repository caused an increment in the time required to remove a set of monitor rules that bordered 60%. Also in the case of rules removal, there was no significant difference in the performance when removing from one to ten monitor rules.

As mentioned before, in order to stress the scenarios, we conducted experiments considering from one to ten monitor rules for each context type, varying the sizes in the repositories. We observed that an increment of one monitor rule during the activities related to the identification, creation, and modification of monitor rules did not increase the performance of the framework - in terms of average time - over 75%. In fact in most cases the increment in the performance of the framework - when increasing by one the monitor rules - was rather low (see Appendix B). The peaks in the performance were observed in the creation of monitor rules when considering an empty repository. In those experiments considering non-empty repositories, the increment in the performance when augmenting in one the monitor rules, did not

[3] Note that a "low" level of cognition does not require to compute the time for a user operation (100%), while a "medium" (75%) or "high" (50%) level of cognition, do require a computation.

increase the performance of the framework over 35%. For the particular case of rules removal, i.e. considering non-empty repositories, an increment of 100 monitor rules in the repository caused an increment in the time required to remove a set of monitor rules that bordered 60%. Also in the case of rules removal, there was no significant difference in the performance when removing, one, two, or up to ten monitor rules.

6.2.2 Modifying the Ratio of Monitor Rules Matching the Invariant

In our experiments, we wanted to analyse if the amount of rules in the repository matching a context type pattern would influence the performance of the framework when identifying, modifying, creating or removing monitor rules. This assumption was based on the fact that a higher number of monitor rules matching a pattern would imply on a higher number of comparisons. Note that, as described in Section 5.1, comparisons are performed at different levels in order to establish whether a monitor rule *a)* matches only the invariant part of a pattern, *b)* matches the invariant parts of a pattern and the placeholders, or *c)* matches the invariant of a pattern, the placeholders, and the time constraints. The framework relies in these comparisons in order to establish whether a monitor rule has been identified, or should be modified, created or removed from the rules repository.

We conducted a series of experiments to evaluate the performance when increasing only the amount of monitor rules in the repository matching a pattern. More specifically, we considered different sets of rules in the repository matching the invariant of patterns related to *role* and *need* context types. The choice of the context types was based on the results obtained when evaluating the performance for the five context types. As explained in section 6.2.1, the *role*, *cognition*, and *skills* patterns required less time[4] per activity, to obtain the needed monitor rules when compared to the *preferences* and *need* patterns.

[4] This is considering average values

In the experiments we considered cases of repositories composed of 300 monitor rules. We augmented the percentages of monitor rules in the repository matching the invariant part of the pattern from the original 20% (60 monitor rules), to 40%, 60%, 80%, and 100% (120, 180, 240, and 300 monitor rules). The results of the experiments conducted for the *role* context type are shown in Figures 6.6 and 6.7. Similarly the results of the experiments conducted for the *need* context type are shown in Figures 6.8 and 6.9. Each graph represents the results for a particular scenario (Atc-SBS and Wo-SBS).

The performance of the activities concerned with the identification, creation, and modification of monitor rules remained quite similar for both context types when considering the same percentages of monitor rules. In the particular case of removal of monitor rules, the time required for it, was considerably lower. More specifically, the removal of monitor rules required close to one fifth of the time (19%) compared to the other activities[5]. This proportion was increased as the percentage of rules matching a pattern increased. The performance, when removing a monitor rule, is explained by the fact that unlike in the other activities, when eliminating rules from the repository, the framework does not perform comparisons of the pattern structure as in the other cases.

The performance increased linearly every time the amount of rules in the repository matching the invariant of a pattern increased in 20%. This was observed for each context type.

In average, an increment of 20% in the rules matching the invariant part of the pattern for the activities of identification, modification, and creation, incremented the performance in 39% for the *role* context type and 36% for the need context type. The lowest increment was observed in the creation for the *role* context type (13%) in the *Wo-SBS* scenario when increasing the matching monitor rules from 80% to 100%.

[5] This was observed when considering a 20% matching in the repository

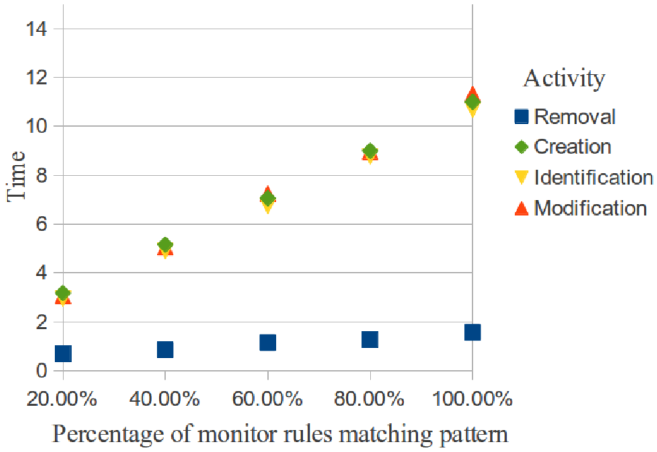


Figure 6.6: Performance for the role context type for the Atc-SBS scenario

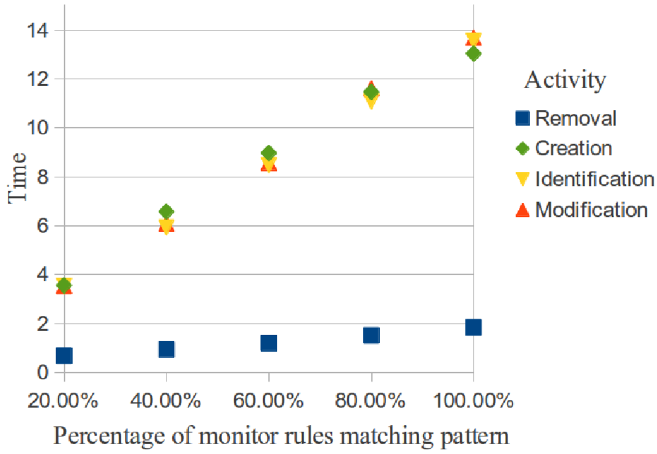


Figure 6.7: Performance for the role context type for the Wo-SBS scenario

The highest increment (84%) was also observed in the creation, for the same context type and scenario when increasing the matching monitor rules from 20% to 40%.

6.2.3 Modifying Annotations Locations

In our evaluation, we were also interested to analyse if the annotations in the SBS specifications could affect the performance of the monitoring activities. More specifically, we analysed the performance for the various activities in the process, for each

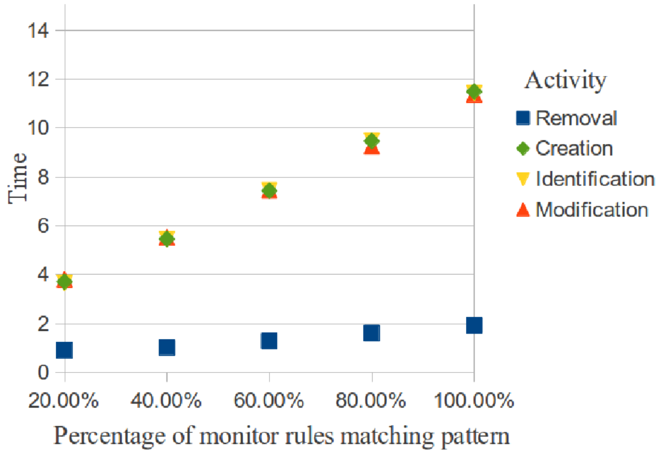


Figure 6.8: Performance for the need context type for the Atc-SBS scenario

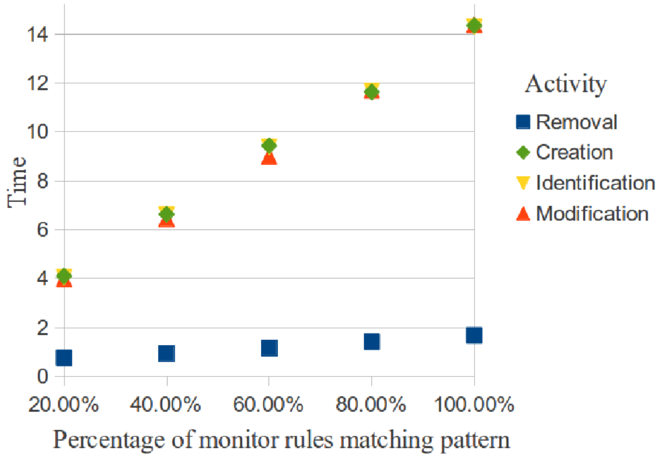


Figure 6.9: Performance for the need context type for the Wo-SBS scenario

context type, considering different annotations for these various context types in different parts of the service-based specification.

For each context type, we considered the original annotations in the service-based specification for that context type. We evaluated the performance for the set of ten original monitor rules increasing the number of operations in the specification before the annotation. More specifically, we considered situations with four, 12, and 36 operations before a specific annotation, as shown in Figure 6.10. In the figure, nodes 2 to 36 represent the additional operations occurring before an annotation. The annotation in the figure is represented by operations *A*, *B*, and *C*.

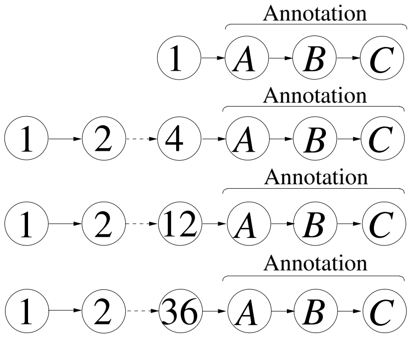


Figure 6.10: Example of the increment of operations occurring before an annotation

Table 6.5 presents the results of the experiments for the identification, modification, and creation of ten monitor rules, for the original ten annotations[6]. More specifically, there is an annotation for each monitor rule. The results shown in Table 6.5 are the mean values for the *Atc-SBS* and *Wo-SBS* scenarios. The results correspond to the performance times, in seconds, for a repository consisting of 300 monitor rules. In the experiment we considered 60 monitor rules (i.e. 20% of the 300 monitor rules in the repository) as rules matching the invariant of the pattern of the context type. For each context type, we considered the same patterns specified in Section 6.2.

The results in the table are the mean values for the performance times for each one of the two scenarios used in our evaluation. Throughout the cases, as shown in the table, the increment in the number of operations in the specification before the respective annotation caused an increment in the performance times. This was expected given that, for all the various activities, it is necessary to traverse the specification to identify the part concerned with the context type.

The results show that in this case an increase in the number of the rules did not affect the performance.

[6] Note that for the case of monitor rules removal, the framework does not depend on the annotations.

		Role		*Cognition*		*Need*		*Skills*		*Preferences*	
		Atc-SBS	Wo-SBS	Atc-SBS	Wo-SBS	Atc-SBS	Wo-SBS	Atc-SBS	Wo-SBS	Atc-SBS	Wo-SBS
Create	original	3.17517	3.56110	2.97185	3.61442	3.70792	4.10166	3.41175	4.10318	3.19158	3.54562
	4 oper.	3.32804	3.95880	3.16637	3.72819	4.03250	4.47694	3.71973	4.40454	3.31873	3.59563
	12 oper.	3.55682	4.19009	3.25926	3.92778	4.38272	4.77596	3.96884	4.78886	3.42801	3.66719
	36 oper.	4.02128	4.61228	3.43314	4.11095	5.28767	5.61016	4.41226	5.21787	3.57110	3.84041
Identify	original	2.94714	3.53824	2.87140	3.53910	3.65399	4.05025	3.35453	4.09584	3.17657	3.55509
	4 oper.	3.26055	3.82396	3.20592	3.72310	4.03573	4.45900	3.66903	4.40328	3.33122	3.67117
	12 oper.	3.42960	4.31389	3.33264	3.88988	4.35642	4.64480	3.85164	4.70320	3.42845	3.70764
	36 oper.	3.95150	4.68087	3.43326	4.14641	5.21005	5.59405	4.26343	5.45732	3.68391	3.93061
Modify	original	3.06779	3.54530	3.11506	3.52472	3.81210	3.99032	3.33367	4.08818	3.17657	3.21509
	4 oper.	3.23117	3.96709	3.17166	3.68476	4.03260	4.49194	3.57755	4.40886	3.31769	3.27034
	12 oper.	3.44187	4.21280	3.37040	3.83862	4.32074	4.76692	3.82669	4.81016	3.42197	3.35363
	36 oper.	3.85157	4.68429	3.47784	3.95831	5.09915	5.59843	4.32815	5.29459	3.49504	3.42441
Remove	all	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A

Table 6.5: Variations

For example, in the case of rules creation for the *skills* context type, the increment in the time performance, when augmenting the amount of operations to four, 12, and 36 was 9.0%, 6.6%, and 11.1% for the *Atc-SBS* scenario, and 7.3%, 8.7%, and 8.9% for the *Wo-SBS* scenario. For the case of *need* and the activity of identifying monitor rules, the increment of operations to four, 12, and 36, resulted in an increment of the time performance of 10.4%, 7.9%, and 19.5% in the *Atc-SBS* scenario and 10.0%, 4.1%, and 20.4% in the *Wo-SBS* scenario. Note that the *need* context type required more time for the different activities compared to the other contexts.

We also analysed the results and we found out that the size of the system specification also affects the performance. This claim is based on the fact that when considering, for two different scenarios, *i)* the same amount of operations preceding the annotations , *ii)* the same size of the repository, and *iii)* the same pattern for a context type and amount of rules matching the invariant of the pattern in the repository; there are still differences in the time performances. For example, the results in Table 6.5 show that for the two similar scenarios (i.e. amount of services, operations and structure of the service specification) the performance for the *Wo-SBS* scenario required more time than the *Atc-SBS* scenario in all cases. When comparing the structures of both service compositions we found that the *Wo-SBS* specification included a higher amount of partner links, variables, and WSDLs than the *Atc-SBS* scenario. We deduced such a difference, which is also reflected in the final size of both specifications (refer to [119] for the BPEL specifications), affected the performance of the framework when carrying out the same activity - under the same configuration - for each scenario. More specifically, the time required to identify the part of the service composition (*path identifier* component) related to a particular context type value, is influenced by the location of the annotation and the size of the BPEL specification, defined partner links, variables, and WSDLs.

6.2.4 Correctness

In order to analyse the correctness of the identified, modified, and created monitor rules, we have used the monitor component implemented in [220]. The evaluation was executed for each different context type. More specifically, we considered patterns created for the various context types (as described in section 6.2); executed the identification, creation, and modification of monitor rules with respect to these patterns; and run those rules in the monitor component. The patterns used for these experiments were chosen based on the types of monitor rules that the monitor component can accept (see Chapter 7). The rules were executed in the monitor component for different sets of events for the various context types.

We used four different events for almost all context types and five different events for context type *need*. The larger number of events used for context type *need* was due to the characteristics of the pattern and the associated part in the specification that involved the invocation of five operations. Examples of these events are the occurrence of a certain service operation when evaluating monitor rules for context type *role*, or the occurrence of a user operation when evaluating monitor rules for context type *skills*. In all the cases, the monitor rules were successfully used and the behaviour of the service-based system was verified.

Tables 6.6, 6.7, 6.8, 6.9, and 6.10 shows the results of these experiments. Each table shows for a context type *i)* the associated monitor rules, *ii)* the events and their time constraints; *iii)* the time stamps for the events. We specify whether the time stamp of the occurrence of an event cause a rule violation, by adding a *violation* or *correct* label next to the time stamp of the event. All the events and time constraints correspond to rules associated to the *Wo-SBS* scenario [119].

For the *role* rules in Table 6.6 there was a violation of rule R_3. More specifically, the event related to the invocation of the operation *mailComposer* (note the 'ic'

<table>
<tr><td colspan="3">$R_1 : Happens(ic\_WebOrganizerv01Request, t_1, R(t_1, t_1)) =>$
$Happens(ic\_mailReviewer, t_2, R(t_1, t_1 + 17.05))$
$R_2 : Happens(ic\_WebOrganizerv01Request, t_1, R(t_1, t_1)) =>$
$Happens(ic\_mailComposer, t_2, R(t_1, t_1 + 29.06))$
$R_3 : Happens(ic\_WebOrganizerv01Request, t_1, R(t_1, t_1)) =>$
$Happens(ic\_accountManager, t_2, R(t_1, t_1 + 38.07))$
$R_4 : Happens(ic\_WebOrganizerv01Request, t_1, R(t_1, t_1)) =>$
$Happens(ic\_enableMessagingService, t_2, R(t_1, t_1 + 9.03))$
$R_5 : Happens(ic\_WebOrganizerv01Request, t_1, R(t_1, t_1)) =>$
$Happens(ic\_opGUIuserOperation, t_2, R(t_1, t_1 + 56.09))$</td></tr>
<tr><td><i>event</i></td><td><i>Time constraint</i></td><td><i>Time stamp</i></td></tr>
<tr><td>$ic\_WebOrganizerv01Request$</td><td>no constraint</td><td>0</td></tr>
<tr><td>$ic\_mailReviewer$</td><td>0 – 17.05</td><td>9.1 (<i>correct</i>)</td></tr>
<tr><td>$ic\_mailComposer$</td><td>0 – 29.06</td><td>42.5 (<i>violation</i> R_3)</td></tr>
<tr><td>$ic\_accountManager$</td><td>0 – 53.08</td><td>51.9 (<i>correct</i>)</td></tr>
<tr><td>$ic\_enableMessagingService$</td><td>0 – 9.03</td><td>8.9 (<i>correct</i>)</td></tr>
<tr><td>$ic\_opGUIuserOperatione_6$</td><td>0 – 56.09</td><td>22.6 (<i>correct</i>)</td></tr>
</table>

Table 6.6: Role Rules for the Wo-SBS scenario

prefix) occurs after the specified time limit. Similarly, for the *skills* context type in Table 6.7, rules R_1 and R_4 were violated; more specifically invocations of operations *opSelectFeatureuserOperation* and *opGUIuserOperation*. For the *cognition* monitor rules in Table 6.8 operations *opSelectMailExpertiseLeveluserOperation* (rule R_2), *opStandardMailuserOperation* (rule R_3) and *opGUIuserOperation* (rule R_5) the responses of operations (note the 'ir' prefix) occurs after the specified time limit. Similarly, for the *preferences* monitor rules in Table 6.9 the reply of operations *opSelectFeatureuserOperation*, *disableMessagingService*, and *WebOrganizerv01* (depicted in monitor rules R_3, R_4, and R_5 respectively) violate the time constraints. Finally, for the *need* monitor rules depicted in Table 6.10 the invocation of operations *mailReviewer* and *initiateCrossReferencing* occur at times that violate the constraints specified in the corresponding rules (monitor rules R_1 and R_4 respectively).

<table>
<tr><td colspan="3">$R_1 : Happens(ic\_WebOrganizerv01Request, t_1, R(t_1, t_1)) \wedge$
$Happens(ir\_WebOrganizerv01, t_3, R(t_1, t_1 + 67.120)) =>$
$Happens(ic\_opSelectFeatureuserOperation, t_2, R(t_1, t_3))$
$R_2 : Happens(ic\_WebOrganizerv01Request, t_1, R(t_1, t_1)) \wedge$
$Happens(ir\_WebOrganizerv01, t_3, R(t_1, t_1 + 67.120)) =>$
$Happens(ic\_opStandardMailuserOperation, t_2, R(t_1, t_3))$
$R_3 : Happens(ic\_WebOrganizerv01Request, t_1, R(t_1, t_1)) \wedge$
$Happens(ir\_WebOrganizerv01, t_3, R(t_1, t_1 + 67.120)) =>$
$Happens(ic\_opAdvancedMailuserOperation, t_2, R(t_1, t_3))$
$R_4 : Happens(ic\_WebOrganizerv01Request, t_1, R(t_1, t_1)) \wedge$
$Happens(ir\_WebOrganizerv01, t_3, R(t_1, t_1 + 67.120)) =>$
$Happens(ic\_opGUIuserOperation, t_2, R(t_1, t_3))$
$R_5 : Happens(ic\_WebOrganizerv01Request, t_1, R(t_1, t_1)) \wedge$
$Happens(ir\_WebOrganizerv01, t_3, R(t_1, t_1 + 67.120)) =>$
$Happens(ic\_opAgendauserOperation, t_2, R(t_1, t_3))$</td></tr>
<tr><th><i>event</i></th><th><i>Time constraint</i></th><th><i>Time stamp</i></th></tr>
<tr><td>$ic\_WebOrganizerv01Request$</td><td>no constraint</td><td>0</td></tr>
<tr><td>$ir\_WebOrganizerv01$</td><td>0 – 67.12</td><td>56.61 (<i>correct</i>)</td></tr>
<tr><td>$ic\_opSelectFeatureuserOperation$</td><td>0 – 56.61</td><td>69.33 (<i>violation</i> R_1)</td></tr>
<tr><td>$ic\_opStandardMailuserOperation$</td><td>0 – 56.61</td><td>55.86 (<i>correct</i>)</td></tr>
<tr><td>$ic\_opAdvancedMailuserOperation$</td><td>0 – 56.61</td><td>35.08 (<i>correct</i>)</td></tr>
<tr><td>$ic\_opGUIuserOperation$</td><td>0 – 56.61</td><td>73.75 (<i>violation</i> R_4)</td></tr>
<tr><td>$ic\_opAgendauserOperation$</td><td>0 – 56.61</td><td>49.20 (<i>correct</i>)</td></tr>
</table>

Table 6.7: Skills Rules for the Wo-SBS scenario

<table>
<tr><td colspan="3">$R_1 : Happens(ic\_opSelectFeatureuserOperation, t_1, R(t_1, t_1)) =>$
$Happens(Happens(ir\_opSelectFeatureuserOperation, t_2, R(t_1, t_1 + 2.5))$
$R_2 : Happens(ic\_opSelectMailExpertiseLeveluserOperation, t_1, R(t_1, t_1)) =>$
$Happens(ir\_opSelectMailExpertiseLeveluserOperation, t_2, R(t_1, t_1 + 1.5))$
$R_3 : Happens(ic\_opStandardMailuserOperation, t_1, R(t_1, t_1)) =>$
$Happens(ir\_opStandardMailuserOperation, t_2, R(t_1, t_1 + 2.0))$
$R_4 : Happens(ic\_opAdvancedMailuserOperation, t_1, R(t_1, t_1)) =>$
$Happens(ir\_opAdvancedMailuserOperation, t_2, R(t_1, t_1 + 3.0))$
$R_5 : Happens(ic\_opGUIuserOperation, t_1, R(t_1, t_1)) =>$
$Happens(ir\_opGUIuserOperation, t_2, R(t_1, t_1 + 1.25))$</td></tr>
<tr><th><i>event</i></th><th><i>Time constraint</i></th><th><i>Time stamp</i></th></tr>
<tr><td>All ‘<i>ic</i>′ events</td><td>no constraint</td><td>0</td></tr>
<tr><td>$ir\_opSelectFeatureuserOperation$</td><td>0 – 2.5</td><td>2.35 (<i>correct</i>)</td></tr>
<tr><td>$ir\_opSelectMailExpertiseLeveluserOperation$</td><td>0 – 1.5</td><td>2.14 (<i>violation</i> R_2)</td></tr>
<tr><td>$ir\_opStandardMailuserOperation$</td><td>0 – 2.0</td><td>2.79 (<i>violation</i> R_3)</td></tr>
<tr><td>$ir\_opAdvancedMailuserOperation$</td><td>0 – 3.0</td><td>2.58 (<i>correct</i>)</td></tr>
<tr><td>$ir\_opGUIuserOperation$</td><td>0 – 1.25</td><td>1.68 (<i>violation</i> R_5)</td></tr>
</table>

Table 6.8: Cognition Rules for the Wo-SBS scenario

<table>
<tr><td colspan="3">$R_1 : Happens(ic\_opSelectFeatureuserOperation, t_1, R(t_1, t_1)) =>$
$Happens(ir\_accessChecker, t_2, R(t_1, t_1 + 60))$
$R_2 : Happens(ic\_WebOrganizerv01Request, t_1, R(t_1, t_1)) =>$
$Happens(ir\_enableMessagingService, t_2, R(t_1, t_1 + 50))$
$R_3 : Happens(ic\_enableMessagingService, t_1, R(t_1, t_1)) =>$
$Happens(ir\_opSelectFeatureuserOperation, t_2, R(t_1, t_1 + 30))$
$R_4 : Happens(ic\_accessChecker, t_1, R(t_1, t_1)) =>$
$Happens(ir\_disableMessagingService, t_2, R(t_1, t_1 + 40))$
$R_5 : Happens(ic\_opStandardMailuserOperation, t_1, R(t_1, t_1)) =>$
$Happens(ir\_WebOrganizerv01, t_2, R(t_1, t_1 + 50))$</td></tr>
<tr><td><i>event</i></td><td><i>Time constraint</i></td><td><i>Time stamp</i></td></tr>
<tr><td>$ic\_opSelectFeatureuserOperation$</td><td>no constraint</td><td>0</td></tr>
<tr><td>$ir\_accessChecker$</td><td>0 – 60</td><td>55.80 (<i>correct</i>)</td></tr>
<tr><td>$ic\_WebOrganizerv01Request$</td><td>no constraint</td><td>0</td></tr>
<tr><td>$ir\_enableMessagingService$</td><td>0 – 50</td><td>48.13 (<i>correct</i>)</td></tr>
<tr><td>$ic\_enableMessagingService$</td><td>no constraint</td><td>0</td></tr>
<tr><td>$ir\_opSelectFeatureuserOperation$</td><td>0 – 30</td><td>44.41 (<i>violation</i> R_3)</td></tr>
<tr><td>$ic\_accessChecker$</td><td>no constraint</td><td>0</td></tr>
<tr><td>$ir\_disableMessagingService$</td><td>0 – 40</td><td>80.25 (<i>violation</i> R_4)</td></tr>
<tr><td>$ic\_opStandardMailuserOperation$</td><td>no constraint</td><td>0</td></tr>
<tr><td>$ir\_WebOrganizerv01$</td><td>0 – 50</td><td>76.73 (<i>violation</i> R_5)</td></tr>
</table>

Table 6.9: Preferences Rules for the Wo-SBS scenario

$R_1 : Happens(ic\_WebOrganizerv01Request, t_1, R(t_1, t_1)) =>$
$Happens(ic\_mailReviewer, t_2, R(t_1, t_1 + 17.05))\wedge$
$Happens(ir\_mailReviewer, t_3, R(t_2, t_2 + 12))\wedge$
$Happens(ic\_mailComposer, t_4, R(t_3, t_3 + 0.01))\wedge$
$Happens(ir\_mailComposer, t_5, R(t_4, t_4 + 9))$
$R_2 : Happens(ic\_WebOrganizerv01Request, t_1, R(t_1, t_1)) =>$
$Happens(ic\_initiateScheduler, t_2, R(t_1, t_1 + 83.66))\wedge$
$Happens(ir\_initiateScheduler, t_3, R(t_2, t_2 + 6))\wedge$
$Happens(ic\_getLatestDates, t_4, R(t_3, t_3 + 0.01))\wedge$
$Happens(ir\_getLatestDates, t_5, R(t_4, t_4 + 3))$
$R_3 : Happens(ic\_WebOrganizerv01Request, t_1, R(t_1, t_1)) =>$
$Happens(ic\_accountManager, t_2, R(t_1, t_1 + 38.07))\wedge$
$Happens(ir\_accountManager, t_3, R(t_2, t_2 + 15))\wedge$
$Happens(ic\_initiateCalendar, t_4, R(t_3, t_3 + 0.01))\wedge$
$Happens(ir\_initiateCalendar, t_5, R(t_4, t_4 + 6))$
$R_4 : Happens(ic\_WebOrganizerv01Request, t_1, R(t_1, t_1)) =>$
$Happens(ic\_getLatestAnnotations, t_2, R(t_1, t_1 + 95.69))\wedge$
$Happens(ir\_getLatestAnnotations, t_3, R(t_2, t_2 + 3))\wedge$
$Happens(ic\_initiateCrossReferencing, t_4, R(t_3, t_3 + 0.01))\wedge$
$Happens(ir\_initiateCrossReferencing, t_5, R(t_4, t_4 + 6))$
$R_5 : Happens(ic\_WebOrganizerv01Request, t_1, R(t_1, t_1)) =>$
$Happens(ic\_opMailandAgendauserOperation, t_2, R(t_1, t_1 + 71.12))\wedge$
$Happens(ir\_opMailandAgendauserOperation, t_3, R(t_2, t_2 + 3))\wedge$
$Happens(ic\_closeEverything, t_4, R(t_3, t_3 + 0.01))\wedge$
$Happens(ir\_closeEverything, t_5, R(t_4, t_4 + 4))$

event	*Time constraint*	*Time stamp*
$ic\_WebOrganizerv01Request$	no constraint	0
$ic\_mailReviewer$	0 – 17.05	17.53 (*violation* R_1)
$ir\_mailReviewer$	17.53 – 29.53	23.93 (*correct*)
$ic\_mailComposer$	23.93 – 23.94	23.93 (*correct*)
$ir\_mailComposer$	23.93 – 32.93	30.14 (*correct*)
$ic\_initiateScheduler$	0 – 83.66	31.42 (*correct*)
$ir\_initiateScheduler$	31.42 – 37.42	36.45 (*correct*)
$ic\_getLatestDates$	36.45 – 36.46	36.45 (*correct*)
$ir\_getLatestDates$	36.45 – 39.45	39.18 (*correct*)
$ic\_accountManager$	0 – 38.07	23.45 (*correct*)
$ir\_accountManager$	23.45 – 38.45	37.18 (*correct*)
$ic\_initiateCalendar$	37.18 – 37.19	37.18 (*correct*)
$ir\_initiateCalendar$	37.18 – 43.18	39.13 (*correct*)
$ic\_getLatestAnnotations$	0 – 95.69	59.14 (*correct*)
$ir\_getLatestAnnotations$	59.14 – 62.14	60.41 (*correct*)
$ic\_initiateCrossReferencing$	60.41 – 60.42	63.49 (*violation* R_4)
$ir\_initiateCrossReferencing$	63.49 – 69.49	66.26 (*correct*)
$ic\_opMailandAgendauserOperation$	0 – 71.12	56.95 (*correct*)
$ir\_opMailandAgendauserOperation$	56.95 – 59.95	57.56 (*correct*)
$ic\_closeEverything$	57.56 – 57.57	57.56 (*correct*)
$ir\_closeEverything$	57.56 – 61.56	60.63 (*correct*)

Table 6.10: Need Rules for the Wo-SBS scenario

6.3 Summary

In this chapter we described the setup for the experiments and the scenarios used for the analysis and evaluation of our framework.

Regarding the results, we conducted different experiments to evaluate the performance and correctness of our framework. We analysed the results and realised that the performance of the framework was quite similar when comparing one context to another. More specifically, when considering the same amount of monitor rules in a repository, the same amount of monitor rules matching the invariant of a context pattern, and the same amount of identified, modified, created or removed monitor rules, the performance did not change significantly for the different context types. This tendency in the performance was observed even when increasing the amount of monitor rules from one to ten. Also regarding the results, we analysed the annotations and study how their positioning in the service composition can can influence the performance of our framework.

Finally, in order to show the correctness of our framework, we deployed a set of monitor rules in the monitor component [220]. We verified - for each context type - whether monitor rules were followed when the events reflecting the behaviour of the system occurred at the expected times, and whether a subset of monitor rules was violated, when the events reflecting the behaviour of the system occurred at wrong times.

In the following chapter we tackle the limits of our work and how it can be further extended. We analyse the validity of our study and end with the conclusions.

Chapter 7

Conclusions and Future Work

We have described a framework able to deal with the adaptation of the monitor rules specified in Event Calculus. The framework can identify, modify, create, and remove monitor rules concerned with different context types. Furthermore, the framework can deal with the modification and evolution of the system as long as the changes are reflected in the BPEL specification. As described in the introduction, our work has brought contributions to the field of Software Engineering, in particular in the area of web service monitoring adaptation. Under the perspective of service-oriented computing, we have contributed to fill the existing gap among user context, user interaction, and service-based system monitoring (see Chapter 1).

In what follows we list the hypotheses and objectives, previously introduced in Section 1.5, and explain how we they were addressed in the thesis.

General hypothesis

1. *It is possible to adapt monitor rules for Service-based Systems due to users interaction with the system, the different types of user context, and changes in the system itself in order to verify its behaviour.*

Sub-hypotheses

a) *It is possible to automatically identify monitor rules that should be used to monitor SBSs due to changes in different user context types and user interaction with the system.*

b) *It is possible to create and modify monitor rules that should be used to monitor SBSs due to changes in different user context types, user interaction with the system, and service composition specification.*

c) *It is possible to remove monitor rules that are not relevant to the SBS due to changes in different user context types, user interaction with the system, and in the service composition specification.*

d) *It is possible to use the automatically identified, created, or modified monitor rule in a monitor component to verify the correct execution of the system and notify that there has been violation in the behaviour of a system that requires changes in the system.*

Objectives

i) To provide a literature review and state of the art of the work performed in the area. More specifically, we will focus on user context, human computer interaction, SBS monitoring, monitoring adaptation, and their relations.

ii) To identify the distinctive user context types likely to affect the execution of a SBS and to require adaptation of monitor rules.

iii) To develop a model for the representation of the distinctive user context types.

iv) To provide a classification for the various monitor rules with respect to the different user context types.

v) To provide a formalism to represent the various monitor rules for the different context types.

vi) To develop techniques to support the identification, creation, modification and removal of monitor rules.

vii) To implement a prototype tool to support objectives 5 and 6 above.

viii) To evaluate the results of the work in case studies and in a suitable monitor component.

Hypothesis *a)*. As described in Sections 4.2 and 5.3 our approach includes a pattern-based strategy, which allows to uniquely identify rule patterns and those monitor rules matching the patterns. Our user model (see Section 3.3) allows to represent the different characteristics of the user. Furthermore, along with our user model we proposed a strategy to identify user interactions in a BPEL specification (see Section 3.7). This allows our framework to automatically identify monitor rules, proving that hypothesis *a)* was correct.

Hypotheses *b)* and *c)*. A set of cases and experiments were performed (see Sections 5.4 and 6.2) to prove the hypothesis *b)* was correct. More specifically, we changed the user context types, the user interactions with the system, and service composition specifications. Furthermore, we changed the annotations used in the composition and removed operations from the service specifications to prove that hypothesis *c)* was also correct.

Hypothesis *d)*. The experiments conducted in Section 6.2 also showed that it was possible to use our generated rules in a monitor component, to verify the correct execution of the system. This proved that hypothesis *d)* was correct.

The hypotheses *a)*, *b)*, *c)*, and *d)* prove the general hypothesis *1)* was correct.

Regarding the objectives. In Chapter 2 we provided a review and state of the art of the work performed in the area (objective *i)*). In Sections 3.1, 3.2, 3.3, and 3.4 we identified the distinctive user context types likely to affect the execution of a SBS, proposed a model to represent distinctive user context types, and described the benefits of the model (objectives *ii)* and *iii)*). In Section 4.1 we presented and explained the formalism used to represent monitor rules (objective *iv)*). Based on the formalism, in Section 4.2, we proposed a classification for the various monitor rules with respect to the different user context types (objective *v)*). Also in Section 4.2, we introduced our strategy - based on the use of patterns - for the identification, creation, modification, and removal of monitor rules (objective *vi)*). In Section 3.8 we introduced the framework of the prototype tool used to evaluate our work (objective *vii)*). Finally the evaluation of our work was shown in Chapter 6 (objective *viii)*).

Decisions and Assumptions

The applicability of the framework presented in this thesis relies on a series of decisions and assumptions that we recapitulate below.

- Context types. The framework covers the user context types defined in the ontology. In addition, it is assumed that the contextual information from the user if given.

- Annotations. For the identification of the part(s) of the composition related to the context type value(s) of a user, the framework relies on the use of annotations. These annotations are to be performed by a designer, familiar with the service composition and operation.

- Service specification and services. The framework obtains the information - for the semi-instantiation of the monitor rules - from the service specification, which is assumed to be specified in BPEL (*de facto* standard). In addition the framework relies on information obtained from the SLAs - from the different services - for the computation of the time constraints for the monitor rules.

- Operations names. The approach assumes a special syntax in the names of the operations (a *prefix* and a *suffix*), to differentiate between those operations involving user interaction, and those operations not involving user interaction.
- For some patterns we rely on the use of *states*, e.g. security, which are assumed to be given/known.

Regarding the adoption of the approach in a real setting, we believe it is plausible for our framework to be used, under real conditions, seamlessly. More specifically, the above assumptions do not require major modifications of existing services composition. In fact, most of the approach relies on the use of external information and non-intrusive service specification. Even more, we have already carried out some work (see section *Future Work* below) that integrates our work with runtime service adaptation [70]. The research and the results obtained from this integration, in a realistic scenario, demonstrates that our work can be used to complement existing approaches without huge amounts of work. Furthermore, although in our work we considered a set of assumptions, we also believe these assumptions can be relaxed, or even removed, as part of the future work (section *Future Work*).

An important aspect that contributes to consider our proposal - for its adoption under real settings - is the performance of the framework. As showed in Chapter 6, even when increasing the amount of participating services, the number of operations involved, the annotations; and when modifying the service specification the framework handles the retrieval of monitoring rules in an efficient[1] way. Considering that

[1] In most cases, monitor rules require manual intervention, which is a slow and error-prone process

the user context and the monitoring automation has been - as far as we are aware - neglected from the service adaptation process, and also that in most cases the adaptation relies on manual intervention (specially after service composition modifications), the proposed framework offers a viable solution for the dynamic monitoring process.

Regarding the efficiency; as shown in Chapter 6, our framework is capable of dealing with the retrieval several monitor rules in just a few seconds. This time is considerably shorter than the time required for a system[2] to execute. Furthermore, even in those cases in which the monitor rules have been specified, it is still important to verify whether they are still suitable for a specific user and/or the system being monitored. This verification, usually a manual process, also increases the time required for the deployment of the monitor rules.

Discussion

We have described a series of patterns for the specification of monitor rules. Although each one of these patterns focuses on a specific context type property, it can be argued that the monitor rules based on these patterns may also be suitable to monitor the system according to behavioural aspects not necessarily related to the contexts. This is, in fact, a possibility.

We realised throughout our research that the more specialised the constraints used to check the correct behaviour of a system, the less likely these constraints would be, to be used in a different system. In our framework the trade off generality/specificity of the patterns, and hence of the monitor rules based on these patterns, allows the framework to be applicable to almost any service-based system. Furthermore, a detailed pattern - i.e. a pattern that goes beyond the logic (structure) of the process and its syntax, and involves the semantic of the process being monitored - implies the

[2] As observed from the existing services compositions

designer possesses additional information concerned with the behavioural aspects of the system being monitored. We also believe the specification of detailed patterns, in constantly changing systems, requires the constant involvement of the designer to guarantee the applicability of the pattern in a given scenario. Moreover, we believe detailed patterns are likely to be suitable for a restricted set of service-based systems, under a known set of assumptions.

The time required for the specification of a set of monitor rules, for a given set of contexts and a given service-based system, is considerably low. Even in those cases in which we stressed the framework with several monitor rules, the performance never exceeded five seconds. The benefits of an automatic specification and deployment of a set of monitor rules[3] include not only an optimisation in terms of time, but also avoids human participation (e.g. from the designer) which is prone to errors.

Limits

There are some limits regarding the applicability our framework, including:

- Limits in the specification of patterns. Although in our framework it is possible to specify any pattern in terms of Event Calculus (EC) [210], we are limited by the capabilities of the monitor component. More specifically, we cannot make use of all predicates defined in EC. Furthermore the structure of the predicates and the order in the occurrence of the events for the rules specified in the monitor component, is more restricted than the structure and order defined in the original formalism. Note that in those cases in which a pattern is not supported by the monitor component, it is still possible (in some cases) to reformulate it so it can be deployed.

[3] Examples of monitor rules following the specification given in [220], can be found in [119].

- Deployment and modification of monitor rules. Monitor rules must be deployed in the monitor component before the process is executed and it is not possible to modified them while performing the monitoring activity.

- Representation of general states. The *fluents* representing the states of the service compositions are defined in general terms (e.g. an operation has been triggered). A more precise specification of a *fluent* is not practical (although possible) since it requires observation of the system, over an extended period of time. Observation over a period of time is a problem in constant and rapidly adaptive systems such as SBSs.

- Annotations. Although the process is automatised, it still relies on the annotations of the process, which are manually created by the designer. Furthermore in our work we assume annotations are still valid after modifications in the SBS, such as the replacement of an operation.

- Synchronous process. Our framework assumes the business process is *synchronous*, i.e. it expects a response for each operation invocation; however it could also be modified to use asynchronous processes (i.e. there is no response for an invocation) by removing patterns involving operations responses.

Validity

There are a number of threats to validity, which may have impacted our work. One threat to internal validity is concerned with the fact that the monitor rules that are identified, modified, created, or removed in our experiments depend on the existence of annotations specified in service-based systems. However, the use of annotations is not unusual and has been advocated by other approaches to support different tasks in service-based systems.

The issue of correctness and consistency of the annotations can be mitigated if their creation is delegated to people with long-standing experience in service-based computing and BPEL specifications. In practice, we foresee that our framework should be used by applications that are created and annotated by developers with experience in service-based computing and the respective domain of the application. Moreover, the requirements of the applications should also be used to support the definition of annotations.

In the case of evaluating the correctness of the monitor rules, for each context type pattern, we deployed a set of five monitor rules in the monitor component. In the experiments we could only use patterns related to monitor rules that are accepted by the monitor component, given that there are limitations in the monitor component as pointed out in the previous section. Although the experiment considered a restricted number of patterns, this was sufficient since differences between the rules associated with a certain pattern are concerned with the instances of the relevant operations for that pattern (i.e. operation names) and not the semantic described by the patterns.

A threat to external validity is concerned with the data sets used in the experiments. We used two service-based systems and assumed two initial sets of monitor rules in the various repositories to be relevant to the experiments. The validity issues were mitigated in two ways. First the service-based systems used in the experiments were sufficiently complex and with several variations in terms of context types, number and type of operations and services[4], and size of the parts of the systems to be considered, providing a solid variation and context for the conducted experiments. Second the repositories used in the experiments contained a plurality of different monitor rules including rules related to the various context types and rules concerned with other behavioural aspects of the system. One other threat is related to the fact

[4] Note that the composition, in terms of user operations and services, varied from one scenario to the other

that in the experiments we used fixed percentages for the number of monitor rules that match the invariant part for a certain context type pattern (from 20% to a 100%) and for the monitor rules that are not related to the pattern (from 80% to a 0%) in the repositories. This was done to allow for a better comparison of the performance of the framework for different configurations of context types, repository sizes, and number of rules. Although a more random sample of pattern-related and pattern-unrelated monitor rules in the repositories could be considered a more realistic scenario, the use of a random sample for the numbers of monitor rules in the repositories will not provide a coherent way for comparison.

Future Work

Throughout this work we have described, analysed, and proposed a solution for the adaptation of monitor activity centered on the context of the user. Our solution considers a framework which allows to identify, modify, create, and remove user context monitor rules based on a pattern strategy. We evaluated the framework, analysed the results, and covered different topics dealing with the limits and validity of our work. However, there are still issues that can be further addressed.

Expansion of patterns and context types. Our work can be extended by the creation of new patterns for the existing context types (see Chapter 3). Another way of extending our work involves the identification of new context types complementing the old ones. In the case of identifying new context types, it would be also necessary to create the associated patterns, used as templates, for the obtention/removal of monitor rules.

The adaptation of the monitor process at runtime. As it was previously mentioned, the current monitor component is not able to deal with the runtime deployment of monitor rules. In order to deal with this limitation we have conducted some preliminary work consisting of strategies where monitor rules are able to cope with runtime modifications (see [70]), however the deployment, at runtime, of monitor rules remains a strong limitation. In a similar sense we believe another way in which our research can be further extended is by the development of techniques capable of supporting the adaptation of patterns themselves.

Annotations specifications. Our process is not fully-automated, it relies on a designer for the specification of the annotations of the different parts of the service-based system related to the context types. A strategy supporting the automated or semi-automated specification of annotations would reduce the manual intervention (prone to errors) and ensure annotations are valid after modifications in the system. A way of doing this could be by information collection. For example, by observing the behaviour of the user with respect to the system. Also, it is important to note - related to the annotations - that the association of a set of operations to a particular user is quite a complex task. Furthermore, the retrieval of user information - needed for the association with service specification - is a challenging topic and ongoing research in different areas of computer science.

Time constraints. Currently we based the computation of the time constraints from the information provided by the SLAs. We believe that this can be improved by monitoring the actual services and operations over a given period of time. The benefits of such strategy would include up-to-date performance information considering, for example, work loads or amount. The main drawback with this strategy is the need of time to compute/infer the time constraint per participating service.

Final Remarks

Our work supports the verification of the behaviour of web service compositions. We showed that our work is particularly suitable to deal with dynamic scenarios where the specification of the system or the user configuration change constantly. Our work addresses problems in the area that have been somehow neglected by existing approaches such as user context and user interactions. We propose an integrated solution for the monitoring process of web service compositions.

As it was mentioned before, our ontology can be expanded to include other types of context and, therefore, to support other types of monitoring rules that are not only concerned with user context types.

For the patterns, we have chosen Event Calculus to represent them. However, given the generalisation of the patterns we have proposed, we believe that they can be specified in other formalisms that support the representation of states and events

We believe that our pattern-based approach can be adapted to be be used in other types of software systems that involve interaction with users. Furthermore, we believe that our work can be extended to support additional features, such as the ranking of web services - or service providers - based on the runtime behaviour, or the automatic service composition based on the context.

Finally, our work addresses the problems related to the behaviour of a system and the user context in a general manner, contributing - but not limited - to the fields of SOA and HCI, and providing a valid approach to deal with the user context and the user interaction.

Bibliography

[1] *IEEE International Conference on Web Services, ICWS 2011, Washington, DC, USA, July 4-9, 2011*. IEEE Computer Society, 2011.

[2] R. Agostini, C. Bettini, N. Cesa-bianchi, D. Riboni, M. Ruberl, and C. Sala. Towards Highly Adaptive Services for Mobile Computing. In *In Proceedings of IFIP TC8 Working Conference on Mobile Information Systems (MOBIS*, pages 121–134. Springer, 2004.

[3] M. Agranov, E. Potamites, A. Schotter, and C. Tergiman. Beliefs and endogenous cognitive levels: An experimental study. *Games and Economic Behavior*, 75(2):449–463, 2012.

[4] Eshaa M. Alkhalifa. Effects of the cognitive level of thought on learning complex material. *Educational Technology & Society*, 8(2):40–53, 2005.

[5] N. Amálio and G. Spanoudakis. From Monitoring Templates to Security Monitoring and Threat Detection. In *Proceedings of the 2008 Second International Conference on Emerging Security Information, Systems and Technologies*, SECURWARE '08, pages 185–192, Washington, DC, USA, 2008. IEEE Computer Society.

[6] D. Ameller and X. Franch. Service level agreement monitor (salmon). In *Proceedings of the Seventh International Conference on Composition-Based*

Software Systems (ICCBSS 2008), ICCBSS '08, pages 224–227, Washington, DC, USA, 2008. IEEE Computer Society.

[7] M.G. Ames and A.K. Dey. Description of Design Dimensions and Evaluation for Ambient Displays. Technical Report UCB/CSD-02-1211, EECS Department, University of California, Berkeley, 2002.

[8] A. Andrieux, K. Czajkowski, A. Dan, K. Keahey, H. Ludwig, T. Nakata, J. Pruyne, J. Rofrano, S. Tuecke, and M. Xu. Web services agreement specification (ws-agreement), 2007.

[9] V. Andrikopulus, P. Bertoli, S. Bindelli, E. Di Nitto, A. Gehlert, L. Germanovich, R. Kazhamiakin, B. Pernici, P. Pelbani, and T. Weyer. State of the Art Report on Software Engineering Design Knowledge and Survey of HCI and Contextual Knowledge. Technical Report PO-JRA-1.1.1, S-Cube Network of Excellence, 2008.

[10] D. Ardagna, M. Comuzzi, E. Mussi, B. Pernici, and P. Plebani. Paws: A framework for executing adaptive web-service processes. *IEEE Softw.*, 24(6):39–46, 2007.

[11] L. Ardissono, R. Furnari, A. Goy, G. Petrone, and M. Segnan. Fault Tolerant Web Service Orchestration by Means of Diagnosis. In *Proceedings of the Third European conference on Software Architecture*, EWSA'06, pages 2–16, Berlin, Heidelberg, 2006. Springer-Verlag.

[12] A. Arkin, A. Boisvert, A. Midon, T. van Lessen, G. Nodet, M. Riou, Rusin R., M. Szefler, L. Waterman, and J. Yu. Apache ODE. http://ode.apache.org/, 2011. (last visited May 2012).

[13] R. Aschoff and A. Zisman. QoS-Driven Proactive Adaptation of Service Composition. In G. Kappel, Z. Maamar, and H.R. Motahari Nezhad, editors, *IC-*

SOC, volume 7084 of *Lecture Notes in Computer Science*, pages 421–435. Springer, 2011.

[14] R. Aschoff and A. Zisman. Proactive Adaptation of Service Composition. In *SEAMS*, June 2012.

[15] M. Autili, P. Benedetto, and P. Inverardi. Context-aware adaptive services: The plastic approach. In *Proceedings of the 12th International Conference on Fundamental Approaches to Software Engineering: Held as Part of the Joint European Conferences on Theory and Practice of Software, ETAPS 2009*, FASE '09, pages 124–139, Berlin, Heidelberg, 2009. Springer-Verlag.

[16] E. Badidi and L. Esmahi. A Cloud-based Approach for Context Information Provisioning. *CoRR*, abs/1105.2213, 2011.

[17] 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.

[18] K. Bandara, M. Wang, and C. Pahl. Context Modeling and Constraints Binding in Web Service Business Processes. In *Proceedings of the first international workshop on Context-aware software technology and applications*, CASTA '09, pages 29–32, New York, NY, USA, 2009. ACM.

[19] K. Bandara, M. Wang, and C. Pahl. Dynamic Integration of Context Model Constraints in Web Service Processes. In *International Software Engineering Conference*, February 2009.

[20] F. Barbon, P Traverso, M. Pistore, and M. Trainotti. Run-time monitoring of instances and classes of web service compositions. In *Proceedings of the IEEE International Conference on Web Services*, ICWS '06, pages 63–71, Washington, DC, USA, 2006. IEEE Computer Society.

[21] L. Baresi, C. Ghezzi, and S. Guinea. Smart Monitors for Composed Services. In *Proceedings of the 2nd international conference on Service oriented computing*, ICSOC '04, pages 193–202, New York, NY, USA, 2004. ACM.

[22] L. Baresi, C. Ghezzi, and S. Guinea. Towards self-healing service compositions. In *PriSE04, First Conference on the Principles of Software Engineering*, 2004.

[23] L. Baresi and S. Guinea. Towards Dynamic Monitoring of WS-BPEL Processes. In *ICSOC 2005, Third International Conference of Service-Oriented Computing, volume 3826 of Lecture Notes in Computer Science*, pages 269–282. Springer, 2005.

[24] L. Baresi, S. Guinea, O. Nano, and G. Spanoudakis. Comprehensive Monitoring of BPEL Processes. *IEEE Internet Computing*, 14(3):50–57, May 2010.

[25] L. Baresi, S. Guinea, M. Pistore, and M. Trainotti. Dynamo + Astro: An Integrated Approach for BPEL Monitoring. *2009 IEEE International Conference on Web Services*, pages 230–237, 2009.

[26] L. Baresi, S. Guinea, and P. Plebani. WS-Policy for Service Monitoring. In *6th VLDB Intl. Workshop on Technologies for E-Services, volume 3811 of Lect. Notes in Computer Science*, pages 72–83. Springer, 2006.

[27] M. Baudet. Random polynomial-time attacks and dolev-yao models. *Journal of Automata, Languages and Combinatorics*, 11(1):7–21, 2006.

[28] J. Beaton, B. Myers, J. Stylos, S. Jeong, and Y. Xie. Usability Evaluation for Enterprise SOA APIs. In *Proceedings of the 2nd international workshop on Systems development in SOA environments*, SDSOA '08, pages 29–34, New York, NY, USA, 2008. ACM.

[29] C. Beeri, A. Eyal, T. Milo, and A. Pilberg. Monitoring Business Processes with Queries. In *Proceedings of the 33rd Int'l Conference on Very large data bases (VLDB '07)*, pages 603–614, Wien, Austria, sep 2007. VLDB Endowment.

[30] Simone Beets and Janet Wesson. Using information visualisation to support visual web service discovery. In *Proceedings of the 26th Annual BCS Interaction Specialist Group Conference on People and Computers*, BCS-HCI '12, pages 11–20, Swinton, UK, UK, 2012. British Computer Society.

[31] Thoms Bell. The concept of dynamic analysis. *SIGSOFT Softw. Eng. Notes*, 24:216–234, October 1999.

[32] G. Belzquez, A. Berlanga, and JM. Molina. In Contexto: Multisensor Architecture to Obtain People Context from Smartphones. *International Journal of Distributed Sensor Networks*, 15, 2012.

[33] S. Benbernou et al. State of the Art Report, Gap Analysis of Knowledge on Principles, Techniques and Methodologies and Adaptation of SBAs. Technical Report PO-JRA-1.2.1, S-Cube Network of Excellence, 2008.

[34] P. Berry, M. Gervasio, B. Peintner, and N. Yorke-Smith. PTIME: Personalized Assistance for Calendaring. *ACM Trans. Intell. Syst. Technol.*, 2(4):40:1–40:22, July 2011.

[35] C. Bettini, D. Maggiorini, and D. Riboni. Distributed Context Monitoring for the Adaptation of Continuous Services. *World Wide Web*, 10(4):503–528, December 2007.

[36] M. Bhatti, F. Hussain, and V. KalianiSundram. An SOA Based Real Time Information Exchange Between Military And Government Owned Rescue Services. *Far East Journal of Psychology and Business*, 7 No 1 Paper 3 April(3):29–55, 2012.

[37] D. Bianculli and C. Ghezzi. Monitoring Conversational Web Services. In *2nd international workshop on Service oriented software engineering: in conjunction with the 6th ESEC/FSE joint meeting*, IW-SOSWE '07, pages 15–21, New York, NY, USA, 2007. ACM.

[38] D. Bianculli, C. Ghezzi, C. Pautasso, and P. Senti. Specification Patterns from Research to Industry: a Case Study in Service-based Applications. In *Proceedings of the 34th International Conference on Software Engineering (ICSE 2012), Zürich, Switzerland*, pages 968–976. IEEE Computer Society Press, June 2012. SEiP track acceptance rate: 18.5% (20/108).

[39] W. Binder, D. Bonetta, C. Pautasso, A. Peternier, D. Milano, H. Schuldt, N. Stojnic, B. Faltings, and I. Trummer. Towards Self-Organizing Service-Oriented Architectures. In *SERVICES*, pages 115–121. IEEE Computer Society, 2011.

[40] M. Blake, D. Kahan, and M. Nowlan. Context-aware Agents for User-oriented Web Services Discovery and Execution. *Distrib. Parallel Databases*, 21(1):39–58, February 2007.

[41] E. Blanco, H. C Cankaya, and D. Moldovan. Composition of Semantic Relations : Model and Applications. *Proceedings of the 23rd International Conference on Computational Linguistics Posters*, (August):72–80, 2010.

[42] C. Bolchini, C. Curino, E. Quintarelli, F. Schreiber, and L. Tanca. A Data-oriented Survey of Context Models. *SIGMOD Rec.*, 36:19–26, December 2007.

[43] C. Bolchini, G. Orsi, E. Quintarelli, F.A. Schreiber, and L. Tanca. Context Modeling and Context Awareness: steps forward in the Context-ADDICT project. *IEEE Data Eng. Bull.*, 34(2):47–54, 2011.

[44] M. Boukhebouze, W. Ferreira, and E. Lim. Yet Another BPEL Extension for User Interactions. In O. De Troyer, C. Bauzer, R. Billen, P. Hallot, A. Simitsis, and H. Van Mingroot, editors, *ER Workshops*, volume 6999 of *Lecture Notes in Computer Science*, pages 24–33. Springer, 2011.

[45] P. Branco and M.L. Encarnação. Affective Computing for Behavior-Based UI Adaptation.

[46] K. Bratanis, D. Dranidis, and A. Simons. An Extensible Architecture for Runtime Monitoring of Conversational Web Services. In *Proceedings of the 3rd International Workshop on Monitoring, Adaptation and Beyond*, MONA '10, pages 9–16, New York, NY, USA, 2010. ACM.

[47] A. Brown and M. Ryan. Context-Aware Monitoring of Untrusted Mobile Applications. In *MobiSec*, pages 83–96, 2009.

[48] A. Bucchiarone, C. Cappiello, E. Di Nitto, R. Kazhamiakin, V. Mazza, and M. Pistore. Design for Adaptation of Service-Based Applications: Main Issues and Requirements. In Asit Dan, Frdric Gittler, and Farouk Toumani, editors, *Service-Oriented Computing. ICSOC/ServiceWave 2009 Workshops*, volume 6275 of *Lecture Notes in Computer Science*, pages 467–476. Springer Berlin / Heidelberg, 2010.

[49] A. Bucchiarone, R. Kazhamiakin, C. Cappiello, E. di Nitto, and V. Mazza. A Context-driven Adaptation Process for Service-based Applications. In *Proceedings of the 2nd International Workshop on Principles of Engineering Service-Oriented Systems*, PESOS '10, pages 50–56, New York, NY, USA, 2010. ACM.

[50] A. Bucchiarone, A. Lafuente, A. Marconi, and M. Pistore. A Formalisation of Adaptable Pervasive Flows. In *Proceedings of the 6th international confer-*

ence on Web services and formal methods, WS-FM'09, pages 61–75, Berlin, Heidelberg, 2010. Springer-Verlag.

[51] A. Bucchiarone, M. Pistore, H. Raik, and R. Kazhamiakin. Adaptation of Service-based Business Processes by Context-aware Replanning. In K-J Lin, C. Huemer, M.B. Blake, and B. Benatallah, editors, *SOCA*, pages 1–8. IEEE, 2011.

[52] J. Budzik, S. Bradshaw, X. Fu, and K. 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, January 2002.

[53] O. Cabrera, M. Oriol, X. Franch, L. López, J. Marco, O. Fragoso, and R. Santaolaya. WeSSQoS: A Configurable SOA System for Quality-aware Web Service Selection. *ArXiv e-prints*, October 2011.

[54] G. Calvary, J. Coutaz, and D. 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.

[55] L.P. Carvalho and P.C. da Silva. CCMF, Computational Context Modeling Framework - An Ontological Approach to Develop Context-Aware Web Applications. In M. Tanvir Afzal, editor, *Semantics in Action - Applications and Scenarios*, chapter 3, pages 63–84. INTECH, 2012.

[56] R. Casado, J. Tuya, and M. Younas. Testing the reliability of web services transactions in cooperative applications. In *Proceedings of the 27th Annual ACM Symposium on Applied Computing*, SAC '12, pages 743–748, New York, NY, USA, 2012. ACM.

[57] J. Cassady and R. Johnson. Cognitive test anxiety and academic performance. *Contemporary Educational Psychology*, 27(2):270 – 295, 2002.

[58] A.T.S. Chan and S.N. Chuang. MobiPADS: A Reflective Middleware for Context-Aware Mobile Computing. *IEEE Transactions on Software Engineering*, 29(12):1072–1085, 2003.

[59] A. Charfi, T. Dinkelaker, and M. Mezini. A Plug-in Architecture for Self-Adaptive Web Service Compositions. In *Proceedings of the 2009 IEEE International Conference on Web Services*, ICWS '09, pages 35–42, Washington, DC, USA, 2009. IEEE Computer Society.

[60] G. Chen and D. Kotz. A Survey of Context-Aware Mobile Computing Research. Technical report, Hanover, NH, USA, 2000.

[61] H. Chen, T. Finin, and A. Joshi. An Ontology for Context-aware Pervasive Computing Environments. *Knowl. Eng. Rev.*, 18(3):197–207, September 2003.

[62] I. Chen, S. Yang, and J. Zhang. Ubiquitous provision of context aware web services. In *IEEE SCC*, pages 60–68. IEEE Computer Society, 2006.

[63] Z. Chen and T. Li. Addressing diverse user preferences: A framework for query results navigation. *IEEE Data Eng. Bull.*, 32(4):41–48, 2009.

[64] K. Cheverst, K. Mitchell, and N. Davies. Investigating Context-aware Information Push vs. Information Pull to Tourists. In *In Proceedings of Mobile HCI 01*, page 2001, 2001.

[65] E. Christensen, F. Curbera, G. Meredith, and S. Weerawarana. Web Service Definition Language (WSDL). Technical report, March 2001.

[66] K. Christos, C. Vassilakis, E. Rouvas, and P. Georgiadis. QoS-Driven Adaptation of BPEL Scenario Execution. In *Proceedings of the 2009 IEEE Interna-*

tional Conference on Web Services, ICWS '09, pages 271–278, Washington, DC, USA, 2009. IEEE Computer Society.

[67] J. Clark and S. DeRose. XML Path Language (XPath) version 1.0. Recommendation, World Wide Web Consortium, November 1999. See http://www.w3.org/TR/xpath.html.

[68] M. Cole, J. Gwizdka, and N. Belkin. Physiological Data as Metadata. In *SIGIR 2011 Workshop on Enriching Information Retrieval (ENIR2011)*, 2011.

[69] M. Comuzzi and G. Spanoudakis. Describing and Verifying Monitoring Capabilities for SLA-driven Service-based Systems. In *CaiSE Forum*, Amsterdam, 2009.

[70] R. Contreras, A. Marconi, M. Pistore, and A. Zisman. A Framework for Dynamic Monitoring of Adaptable Service-based Systems. In *Proceedings of the 4rd International Workshop on Principles of Engineering Service-Oriented Systems*, PESOS '12, New York, NY, USA, 2012. ACM.

[71] R. Contreras and A. Zisman. A Pattern-based Approach for Monitor Adaptation. In *Software Science, Technology and Engineering (SWSTE), 2010 IEEE International Conference on*, pages 30 –37, june 2010.

[72] R. Contreras and A. Zisman. Identifying, Modifying, Creating, and Removing Monitor Rules for Service-oriented Computing. In *Proceedings of the 3rd International Workshop on Principles of Engineering Service-Oriented Systems*, PESOS '11, pages 43–49, New York, NY, USA, 2011. ACM.

[73] D. Cooray, S. Malek, and R. Roshandel. Context-driven optimization of mobile service-oriented systems for improving their resilience. In *Proceedings of the 2010 6th World Congress on Services*, SERVICES '10, pages 677–682, Washington, DC, USA, 2010. IEEE Computer Society.

[74] E. Crawford and M. Veloso. Mechanism design for multi-agent meeting scheduling including time preferences, availability, and value of presence. In *Proceedings of the IEEE/WIC/ACM International Conference on Intelligent Agent Technology*, IAT '04, pages 253–259, Washington, DC, USA, 2004. IEEE Computer Society.

[75] J. Crowley, J. Coutaz, G. Rey, and P. Reignier. Perceptual Components for Context Aware Computing. In *Proceedings of the 4th international conference on Ubiquitous Computing*, UbiComp '02, pages 117–134, London, UK, UK, 2002. Springer-Verlag.

[76] M. Cruz-Torres, V. Noël, T. Holvoet, and J. Arcangeli. MAS Organisations to Adapt your Composite Service. In *Proceedings of the 3rd International Workshop on Monitoring, Adaptation and Beyond*, MONA '10, pages 33–39, New York, NY, USA, 2010. ACM.

[77] M de Leoni. Adaptive Process Management in Highly Dynamic and Pervasive Scenarios. In M. H. ter Beek, editor, *YR-SOC*, volume 2 of *EPTCS*, pages 83–97, 2009.

[78] A-M Dery-Pinna, J. Fierstone, and E. Picard. Component Model and Programming: A First Step to Manage Human Computer Interaction Adaptation. In Luca Chittaro, editor, *Human-Computer Interaction with Mobile Devices and Services*, volume 2795 of *Lecture Notes in Computer Science*, pages 456–460. Springer Berlin / Heidelberg, 2003.

[79] A. Dey. *Providing Architectural Support for Building Context-aware Applications*. PhD thesis, Atlanta, GA, USA, 2000. AAI9994400.

[80] A. Dey. Understanding and Using Context. *Personal Ubiquitous Comput.*, 5(1):4–7, January 2001.

[81] A.K. Dey and G.D. Abowd. The Context Toolkit: Aiding the Development of Context-Aware Applications. pages 434–441. ACM Press, 1999.

[82] I. Di Pietro, F. Pagliarecci, L. Spalazzi, A. Marconi, and M. Pistore. Semantic Web Service Selection at the Process-Level: The eBay/Amazon/PayPal Case Study. In *Web Intelligence and Intelligent Agent Technology, 2008. WI-IAT '08. IEEE/WIC/ACM International Conference on*, volume 1, pages 605 –611, dec. 2008.

[83] M.B. Dwyer, G.S. Avrunin, and J.C. Corbett. Property Specification Patterns for Finite-state Verification. In *Proceedings of the second workshop on Formal methods in software practice*, FMSP '98, pages 7–15, New York, NY, USA, 1998. ACM.

[84] M.B. Dwyer, G.S. Avrunin, and J.C. Corbett. Patterns in Property Specifications for Finite-State Verification. In B.W. Boehm, D. Garlan, and J. Kramer, editors, *ICSE*, pages 411–420. ACM, 1999.

[85] H. Eberle, S. Föll, K. Herrmann, F. Leymann, A. Marconi, T. Unger, and H. Wolf. Enforcement from the Inside: Improving Quality of Business in Process Management. In *ICWS*, pages 405–412. IEEE, 2009.

[86] R. J. Ellison. Trustworthy Integration: Challenges for the Practitioner. Technical report, Software Engineering Institute, Cernegie Mellon University, 2005.

[87] L. Encarnação. Multi-Level User Support through Adaptive Hypermedia: A Highly Application-Independent Help Component. In *Proceedings of the 2nd international conference on Intelligent user interfaces*, IUI '97, pages 187–194, New York, NY, USA, 1997. ACM.

[88] I. Epifani, C. Ghezzi, R. Mirandola, and G. Tamburrelli. Model evolution by run-time parameter adaptation. In *Proceedings of the 31st International*

Conference on Software Engineering, ICSE '09, pages 111–121, Washington, DC, USA, 2009. IEEE Computer Society.

[89] Thomas Erl. *Service-Oriented Architecture: Concepts, Technology, and Design.* Prentice Hall PTR, Upper Saddle River, NJ, USA, 2005.

[90] M.D. Ernst. Static and Dynamic Analysis: Synergy and Duality. In *WODA 2003: Workshop on Dynamic Analysis*, pages 24–27, Portland, Oregon, May 9, 2003.

[91] A. Erradi, P. Maheshwari, and V. Tosic. Ws-policy based monitoring of composite web services. In *Proceedings of the Fifth European Conference on Web Services*, ECOWS '07, pages 99–108, Washington, DC, USA, 2007. IEEE Computer Society.

[92] D. Evans, D. Eyers, and J. Bacon. Linking policies to the spatial environment. *Policies for Distributed Systems and Networks, IEEE International Workshop on*, 0:73–76, 2010.

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

[94] F. Felhi and J. Akaichi. Adaptation of web services to the context based on workflow: Approach for self-adaptation of service-oriented architectures to the context. *CoRR*, abs/1211.4867, 2012.

[95] E. Fernández, O. Ajaj, I. Buckley, N. Delessy-Gassant, K. Hashizume, and M. Larrondo-Petrie. A survey of patterns for web services security and reliability standards. *Future Internet*, 4(2):430–450, 2012.

[96] G. Fischer. User Modeling in Human-Computer Interaction. *User Modeling and User-Adapted Interaction*, 11(1-2):65–86, March 2001.

[97] M.E. Foster. Generating Embodied Descriptions Tailored to User Preferences. In *Proceedings of the 7th international conference on Intelligent Virtual Agents*, IVA '07, pages 264–271, Berlin, Heidelberg, 2007. Springer-Verlag.

[98] D. Francois, D. Polani, and K. Dautenhahn. Towards Socially Adaptive Robots: A Novel Method for Real-time Recognition of Human-robot Interaction Styles. *Humanoid Robots 2008 Humanoids 2008 8th IEEERAS International Conference on*, pages 353–359, 2009.

[99] D. Franklin, J. Budzik, and K. Hammond. Plan-based Interfaces: Keeping Track of User Tasks and Acting to Cooperate. In *Proceedings of the 7th international conference on Intelligent user interfaces*, IUI '02, pages 79–86, New York, NY, USA, 2002. ACM.

[100] E. Freeman, E. Freeman, B. Bates, and K. Sierra. *Head First Design Patterns*. O' Reilly & Associates, Inc., 2004.

[101] E. Gamma, R. Helm, R. Johnson, and J. Vlissides. *Design Patterns: Elements of Reusable Object-oriented Software*. Addison-Wesley Professional, 1995.

[102] J.H. Gennari, M.A. Musen, R.W. Fergerson, W.E. Grosso, M. Crubezy, H. Eriksson, N.F. Noy, and S.W. Tu. The evolution of Protégé;: an Environment for Knowledge-based Systems Development. *Int. J. Hum.-Comput. Stud.*, 58(1):89–123, January 2003.

[103] G. Giacomo, Y. Lespérance, H.J. Levesque, and S. Sardina. IndiGolog: A High-Level Programming Language for Embedded Reasoning Agents. pages 31–72. 2009.

[104] C. Glahn, M. Specht, and R. Koper. Towards a Service Oriented Architecture for Giving Feedback in Informal Learning Environments. 2007.

[105] M. Golemati, A. Katifori, C. Vassilakis, G. Lepouras, and C. Halatsis. Creating an Ontology for the User Profile: Method and Applications. In C. Rolland, O. Pastor, and J-L. Cavarero, editors, *RCIS*, pages 407–412, 2007.

[106] A. Goodloe and L. Pike. Monitoring distributed real-time systems: A survey and future directions. Technical Report NASA/CR-2010-216724, NASA Langley Research Center, July 2010.

[107] Denis Gopan and Thomas W. Reps. Guided static analysis. In Hanne Riis Nielson and Gilberto Filé, editors, *SAS*, volume 4634 of *Lecture Notes in Computer Science*, pages 349–365. Springer, 2007.

[108] A. Gotsman, F. Massacci, and M. Pistore. Towards an Independent Semantics and Verification Technology for the HLPSL Specification Language. *Electron. Notes Theor. Comput. Sci.*, 135(1):59–77, July 2005.

[109] T.R. Gruber. A Translation Approach to Portable Ontology Specifications. *Knowl. Acquis.*, 5(2):199–220, June 1993.

[110] T. Gu, X. Wang, H. Pung, and D. Zhang. An ontology-based context model in intelligent environments. In *IN PROCEEDINGS OF COMMUNICATION NETWORKS AND DISTRIBUTED SYSTEMS MODELING AND SIMULATION CONFERENCE*, pages 270–275, 2004.

[111] G. Hackmann, C. Julien, J. Payton, and G. Roman. Supporting Generalized Context Interactions. In *In: Proceedings of the 4 th International Workshop on Software Engineering for Middleware*, pages 91–106, 2004.

[112] S. Hanks and D. McDermott. Nonmonotonic logic and temporal projection. *Artificial Intelligence*, 33(3):379 – 412, 1987.

[113] J. Heflin and Z. Pan. A Model Theoretic Semantics for Ontology Versioning. In S.A. McIlraith, D. Plexousakis, and F. van Harmelen, editors, *International Semantic Web Conference*, volume 3298 of *Lecture Notes in Computer Science*, pages 62–76. Springer, 2004.

[114] T. Hewett, R. Baecker, S. Card, et al. ACM SIGCHI Curricula for Human-computer Interaction. Technical report, New York, NY, USA, 1992.

[115] J. Hielscher, A. Metzger, and R. Kazhamiakin. Taxonomy of Adaptation Principles and Mechanisms. S-Cube project deliverable. Technical Report CD-JRA-1.2.2, S-Cube Network of Excellence, 2009.

[116] R. Hirschfeld, P. Costanza, and O. Nierstrasz. Context-oriented Programming. *Journal of Object Technology*, 7(3):125–151, 2008.

[117] P. Holleis, F. Otto, H. Hussmann, and A. Schmidt. Keystroke-level Model for Advanced Mobile Phone Interaction. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, CHI ’07, pages 1505–1514, New York, NY, USA, 2007. ACM.

[118] P. Holleis and A. Schmidt. Makeit: Integrate user interaction times in the design process of mobile applications. In *Proceedings of the 6th International Conference on Pervasive Computing*, Pervasive ’08, pages 56–74, Berlin, Heidelberg, 2008. Springer-Verlag.

[119] http://vega.soi.city.ac.uk/~abdw747/MADap. Monitor Adaptation Framework.

[120] http://www.ibm.com/developerworks/library/specification/ws bpel/. Business Process Execution Language for Web Services version 1.1, February 2007.

[121] IBM. WebSphere Process Server. http://www-01.ibm.com/software/integration/wps/, 2011. (last visited May 2012).

[122] N. Ibrahim, V.S. Alagar, and M. Mohammad. Specification and Verification of Context-dependent Services. In L. Kovács, R. Pugliese, and F. Tiezzi, editors, *WWV*, volume 61 of *EPTCS*, pages 17–33, 2011.

[123] Y.E. Ioannidis, M. Vayanou, T. Georgiou, K. Iatropoulou, M. Karvounis, V. Katifori, M. Kyriakidi, N. Manola, A. Mouzakidis, L. Stamatogiannakis, and M. Triantafyllidi. Profiling attitudes for personalized information provision. *IEEE Data Eng. Bull.*, 34(2):35–40, 2011.

[124] F. Ishikawa, B. Suleiman, K. Yamamoto, and S. Honiden. Physical interaction in pervasive computing: formal modeling, analysis and verification. In *Proceedings of the 2009 international conference on Pervasive services*, ICPS '09, pages 133–140, New York, NY, USA, 2009. ACM.

[125] Jacko J. and C. Stephanidis. *Human-Computer Interaction: Theory and Practice (Part 1)*. Human Factors and Ergonomics Series. Taylor & Francis Group, 2003.

[126] M. Jackson. *Problem Frames: Analyzing and Structuring Software Development Problems*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 2001.

[127] L. Jiang, S. Parekh, and J. Walrand. Time-Dependent Network Pricing and Bandwidth Trading. In *In Proceedings IEEE BoD*, 2008.

[128] B. John and D. Kieras. Using GOMS for User Interface Design and Evaluation: Which Technique? *ACM Transactions on Computer-Human Interaction*, 3:287–319, 1996.

[129] R.W. Jones. *Security, Strategy, and Critical Theory*. Critical Security Studies. Lynne Rienner Publishers, 1999.

[130] M.B. Juric and M. Krizevnik. *Ws-bpel 2.0 for Soa Composite Applications With Oracle Soa Suite 11g*. Packt Publishing, 2010.

[131] S. Kang, J. Lee, H. Jang, H. Lee, Y. Lee, S. Park, T. Park, and J. Song. SeeMon: Scalable and Energy-efficient Context Monitoring Framework for Sensor-rich Mobile Environments. In *MobiSys '08: Proceeding of the 6th international conference on Mobile systems, applications, and services*, pages 267–280, New York, NY, USA, June 2008. ACM Press.

[132] G.M. Kapitsaki and A. Achilleas. Applying Model-driven Engineering for Linking Web Service and Context Models. In *Proceedings of the 13th International Conference on Information Integration and Web-based Applications and Services*, iiWAS '11, pages 511–514, New York, NY, USA, 2011. ACM.

[133] N. Kavantzas, D. Burdett, G. Ritzinger, T. Fletcher, Y. Lafon, and C. Barreto. Web Services Choreography Description Language Version 1.0. World Wide Web Consortium, Candidate Recommendation CR-ws-cdl-10-20051109, November 2005.

[134] R. Kazhamiakin. *Formal Analysis of Web Service Compositions*. PhD thesis, Trento, Italy, March 2007. AAI9994400.

[135] R. Kazhamiakin et al. Baseline of Adaptation and Monitoring Principles, Techniques, and Methodologies across Functional SBA Layers. *S-Cube project deliverable*, 2009.

[136] R. Kazhamiakin, G. Kecskemeti, Jl. Poizat, F. Silvestri, M. Uhlig, B. Wetzstein, S. Benbernou, and L. Cavallaro. State of the Art Report, Gap Analysis of Knowledge on Principles, Techniques and Methodologies for Monitoring and Adaptation of SBAs. *Portal*, 215483:3101–3119, 2008.

[137] E. Keller and H. Ludwig. The WSLA Framework: Specifying and Monitoring Service Level Agreements for Web Services. *Journal of Network and Systems Management*, 11:2003, 2003.

[138] A. Khan, Sharma N.K., and Dixit S. Effect of Cognitive Load and Paradigm on Time Perception. *Journal of the Indian Academy of Applied Psychology*, 32:37–42, 2006.

[139] K. Kirschoff, D. Capurro, and A. Turner. Evaluating User Preferences in Machine Translation Using Conjoint Analysis. In *Proceedings of the 16th EAMT Conference, 28-30 May 2012*, 2012.

[140] M. Kloppmann, D. Koenig, F. Leymann, A. Rickayzen, C. von Riegen, P. Schmidt, and I. Trickovic. WS-BPEL Extension for People - BPEL4People. White paper, IBM / SAP, July 2007.

[141] J. Knottenbelt. *Contract Related Agents*. PhD thesis, Department of Computing, Imperial College London, December 2006.

[142] W. Kongdenfha, H.R. Motahari-Nezhad, B. Benatallah, F. Casati, and R. Saint-Paul. Mismatch Patterns and Adaptation Aspects: A Foundation for Rapid Development of Web Service Adapters. *IEEE Trans. Serv. Comput.*, 2(2):94–107, April 2009.

[143] R Kowalski and M Sergot. A Logic-based Calculus of Events. *New Gen. Comput.*, 4(1):67–95, January 1986.

[144] R. Krummenacher and T. Strang. Ontology-Based Context Modeling. In *In Workshop on Context-Aware Proactive Systems*, 2007.

[145] S. Kujala. *User Studies: A Practical Approach to User Involvement for Gathering User Needs and Requirements*. Acta polytechnica Scandinavica: Mathematics and computing series. Finnish Academies of Technology, 2002.

[146] G. Lakshmanan, P. Keyser, A. Slominski, F. Curbera, and R. Khalaf. A Business Centric End-to-End Monitoring Approach for Service Composites. In *IEEE SCC*, pages 409–416. IEEE Computer Society, 2010.

[147] D. Lamanna, J. Skene, and W. Emmerich. Slang: A language for defining service level agreements. In *Proceedings of the The Ninth IEEE Workshop on Future Trends of Distributed Computing Systems*, FTDCS '03, pages 100–, Washington, DC, USA, 2003. IEEE Computer Society.

[148] D.N. Le, N.S. Nguyen, K. Mous, R.K.L. Ko, and A.E.S. Goh. Generating Request Web Services from Annotated BPEL. In *RIVF*, pages 1–8. IEEE, 2009.

[149] Y. Lee and S-B. Cho. Human Activity Inference Using Hierarchical Bayesian Network in Mobile Contexts. In B-L. Lu, L. Zhang, and J.T. Kwok, editors, *ICONIP (1)*, volume 7062 of *Lecture Notes in Computer Science*, pages 38–45. Springer, 2011.

[150] C. Li. User Preferences, Information Transactions and Location-based Services: A Study of Urban Pedestrian Wayfinding. *Computers Environment and Urban Systems*, 30(6):726–740, 2006.

[151] H. Lieberman. Letizia: An Agent that Assists Web Browsing. In *Proceedings of the 14th international joint conference on Artificial intelligence - Volume 1*, IJCAI'95, pages 924–929, San Francisco, CA, USA, 1995. Morgan Kaufmann Publishers Inc.

[152] M. Loskyll, Ines Heck, Jochen Schlick, and Michael Schwarz. Context-based orchestration for control of resource-efficient manufacturing processes. *Future Internet*, 4(3):737–761, 2012.

[153] H. Ludwig, A. Dan, and R. Kearney. Cremona: an Architecture and Library for Creation and Monitoring of WS-agreents. In *Proceedings of the 2nd international conference on Service oriented computing*, ICSOC '04, pages 65–74, New York, NY, USA, 2004. ACM.

[154] L. Luo and D. Siewiorek. Klem: A method for predicting user interaction time and system energy consumption during application design. In *Proceedings of the 2007 11th IEEE International Symposium on Wearable Computers*, ISWC '07, pages 1–8, Washington, DC, USA, 2007. IEEE Computer Society.

[155] A. Maedche and S. Staab. Ontology Learning for the Semantic Web. *IEEE Intelligent Systems*, 16(2):72–79, March 2001.

[156] B. Magableh and S. Barrett. Context-oriented Software Development. *Journal of Emerging Technologies in Web Intelligence*, 4(2), 2012.

[157] P. Maglio, R. Barrett, C. Campbell, and T. Selker. SUITOR: An Attentive Information System. In *Proceedings of the 5th international conference on Intelligent user interfaces*, IUI '00, pages 169–176, New York, NY, USA, 2000. ACM.

[158] K. Mahbub. *Runtime Monitoring of Service-based Systems*. PhD thesis, City University London, November 2007.

[159] K. Mahbub and G. Spanoudakis. Run-time Monitoring of Requirements for Systems Composed of Web-Services: Initial Implementation and Evaluation Experience. In *In ICWS 05*, pages 257–265. IEEE Computer Society, 2005.

[160] K. Mahbub and G. Spanoudakis. Monitoring WS-Agreements: An Event Calculus Based Approach. In *In Test and Analysis of Web Services, (eds) Baresi L., di Nitto E*, pages 265–306. Springer Verlang, 2007.

[161] M.F. Mahfouf, M.and Abbod and D.A. Linkens. A Survey of Fuzzy Logic Monitoring and Control Utilisation in Medicine. *Artificial Intelligence in Medicine*, 21(13):27 – 42, 2001. Fuzzy Theory in Medicine.

[162] N. Maiden. Codified Human-Computer Interaction (HCI) Knowledge and Context Factors. Technical Report PO-JRA-1.1.3, S-Cube Network of Excellence, 2009.

[163] M. Mancioppi, B. Pernici, M. Carro, and D. Ivanovic. Consolidated and Updated State of the Art Report on Service-based Applications. Technical Report CD-IA-1.1.7, S-Cube Network of Excellence, 2011.

[164] M. Marzolla and R. Mirandola. Qos analysis for web service applications: a survey of performance-oriented approaches from an architectural viewpoint. Technical Report UBLCS-2010-05, Department of Computer Science, University of Bologna, February 2010.

[165] R. Mayer and R. Moreno. Nine Ways to Reduce Cognitive Load in Multimedia Learning. *Educational Psychologist*, 38(1):43–52, March 2003.

[166] R. Mayrhofer, H. Radi, and A. Ferscha. Recognizing and Predicting Context by Learning from User Behavior. pages 25–35, 2003.

[167] K. Mccarthy, L. Mcginty, and B. Smyth. Dynamic Critiquing: An Analysis of Cognitive Load, 2005.

[168] J. Mendling and M. Hafner. From WS-CDL Choreography to BPEL Process Orchestration. *J. Enterprise Inf. Management*, 21(5):525–542, 2008.

[169] A. Metzger, O. Sammodi, K. Pohl, and M. Rzepka. Towards pro-active adaptation with confidence: augmenting service monitoring with online testing. In

Proceedings of the 2010 ICSE Workshop on Software Engineering for Adaptive and Self-Managing Systems, SEAMS '10, pages 20–28, New York, NY, USA, 2010. ACM.

[170] D. Metzler and W. B. Croft. Beyond Bags of Words: Modeling Implicit User Preferences in Information Retrieval. In *Association for the Advancement of Artificial Intelligence (AAAI)*, 2006.

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

[172] K. Mohan and B. Ramesh. Ontology-Based Support for Variability Management in Product and Service Familie. In *HICSS*, page 75, 2003.

[173] C. Momm, R. Malec, and S. Abeck. Towards a Model-driven Development of Monitored Processes. In Andreas Oberweis, Christof Weinhardt, Henner Gimpel, Agnes Koschmider, Victor Pankratius, and Bjrn Schnizler, editors, *Wirtschaftsinformatik (2)*, pages 319–336. Universitaetsverlag Karlsruhe, 2007.

[174] O. Moser, F. Rosenberg, and S. Dustdar. Non-intrusive Monitoring and Service Adaptation for WS-BPEL. In *Proceedings of the 17th international conference on World Wide Web*, WWW '08, pages 815–824, New York, NY, USA, 2008. ACM.

[175] O. Moser, F. Rosenberg, and S. Dustdar. VieDAME - Flexible and Robust BPEL Processes Through Monitoring and Adaptation. In W. Schäfer, M.B. Dwyer, and V. Gruhn, editors, *ICSE Companion*, pages 917–918. ACM, 2008.

[176] I. Nébel, B. Smith, and R. Paschke. A User Profiling Component with the Aid of User Ontologies. In *In: Workshop Learning - Teaching - Knowledge - Adaptivity (LLWA 03)*, 2003.

[177] C. Niederée, A. Stewart, B. Mehta, and M. Hemmje. A Multi-Dimensional, Unified User Model for Cross-System Personalization. In *E4PIA Workshop 2004*, 2004.

[178] N.F. Noy, R.W. Fergerson, and M.A. Musen. The Knowledge Model of Protg-2000: Combining Interoperability and Flexibility. *Lecture Notes in Computer Science*, 1937:69–82, 2000.

[179] R. Ocampo, L. Cheng, K. Jean, A. Galis, and A. G. Prieto. Towards a Context Monitoring System for Ambient Networks. In *Communications and Networking in China, 2006. ChinaCom '06. First International Conference on*, pages 1–3, 2006.

[180] Oracle. Fusion middleware developer's guide for oracle soa suite 11g. http://docs.oracle.com/cd/E15523_01/index.htm, 2010. (last visited October 2012).

[181] Oracle. Oracle BPEL Process Manager. http://www.oracle.com/bpel, 2011. (last visited May 2012).

[182] M. Oriol, X. Franch, and J. Marco. SALMon: A SOA System for Monitoring Service Level Agreements. Technical Report LSI-10-18-R, Universitat Politècnica de Cataluny, 2010.

[183] M. Oriol, J. Marco, X. Franch, and D. Ameller. Monitoring adaptable soa-systems using salmon. In *Workshop on Service Monitoring, Adaptation and Beyond (Mona+)*, pages 19–28, June 2008.

[184] A. Ouda, H. Lutfiyya, and M. Bauer. Automatic Policy Mapping to Management System Configurations. In *Policies for Distributed Systems and Networks (POLICY), 2010 IEEE International Symposium on*, pages 87 –94, july 2010.

[185] K. Pant. *Business Process Driven SOA using BPMN and BPEL: From Business Process Modeling to Orchestration and Service Oriented Architecture*. Packt Publishing, August 2008.

[186] M. Papazoglou. *Web Services: Principles and Technology*. Pearson, Prentice Hall, 2008.

[187] S. Parry and G. Fischer. Design, adoption, and assessment of a socio-technical environment supporting independence for persons with cognitive disabilities. In *ACM Conference on Human Factors in Computing Systems*, page 606, 2008.

[188] L. Pasquale. From Goals to Reliable Service Compositions. In *17th International Conference on Requirements Engineering*, 2009.

[189] K.T. Pathan, S. Reiff-Marganiec, A. Shaikh, and N. Channa. Reaching Activities by Places in the Context-Aware Environments Using Software Sensors. *Journal of Emerging Trends in Computing and Information Sciences*, 2:665–673, 2011.

[190] C. Peltz. Web Services Orchestration and Choreography. *Computer*, 36(10):46–52, October 2003.

[191] P. Pirolli, W. Fu, R. Reeder, and S. Card. A User-tracing Architecture for Modeling Interaction with the World Wide Web. In *Proceedings of the Working Conference on Advanced Visual Interfaces*, AVI '02, pages 75–83, New York, NY, USA, 2002. ACM.

[192] M. Pistore and P. Traverso. *Assumption-based Composition and Monitoring of Web Services*, pages 307–335. Springer, 2007.

[193] M. Raento, A. Oulasvirta, R. Petit, and H. Toivonen. ContextPhone: A Prototyping Platform for Context-Aware Mobile Applications. *IEEE Pervasive Computing*, 4:51–59, 2005.

[194] R. Reichle, M. Wagner, M. Khan, K. Geihs, J. Lorenzo, M. Valla, C. Fra, N. Paspallis, and G. Papadopoulos. A Comprehensive Context Modeling Framework for Pervasive Computing Systems. In *Proceedings of the 8th IFIP WG 6.1 international conference on Distributed applications and interoperable systems*, DAIS'08, pages 281–295, Berlin, Heidelberg, 2008. Springer-Verlag.

[195] R. Reiter. *Knowledge in Action: Logical Foundations for Specifying and Implementing Dynamical Systems*. The MIT Press, Massachusetts, MA, illustrated edition edition, 2001.

[196] E.M. Roth, E.S. Patterson, and R.J. Mumaw. Cognitive Engineering: Issues in User-Centered System Design. In J.J. Marciniak, editor, *Encyclopedia of Software Engineering*. Wiley-Interscience, John Wiley & Sons, New York, second edition, 2001.

[197] N. Ruiz, R. Taib, Y. Shi, E. Choi, and F. Chen. Using pen input features as indices of cognitive load. In *Proceedings of the 9th international conference on Multimodal interfaces*, ICMI '07, pages 315–318, New York, NY, USA, 2007. ACM.

[198] C. Ruz, F. Baude, and B. Sauvan. Component-based Generic Approach for Reconfigurable Management of Component-based SOA Applications. In *Proceedings of the 3rd International Workshop on Monitoring, Adaptation and Beyond*, MONA '10, pages 25–32, New York, NY, USA, 2010. ACM.

[199] A. Sahai, V. Machiraju, M. Sayal, L. Jin, and F. Casati. Automated SLA Monitoring for Web Services. In *IEEE/IFIP DSOM*, pages 28–41. Springer-Verlag, 2002.

[200] M. Salehie and L. Tahvildari. Self-adaptive Software: Landscape and Research Challenges. *ACM Trans. Auton. Adapt. Syst.*, 4(2):14:1–14:42, May 2009.

[201] M. Salifu, B. Nuseibeh, L. Rapanotti, and T. Tun. Using Problem Descriptions to Represent Variabilities for Context-aware Applications. In *Proceedings of 1st International workshop on Variability Modeling of Software-intensive Systems (VaMoS 2007)*, pages 149–156, Limerick, Ireland, January 2007. LERO (The Irish Software Engineering Research Centre),Lero Technical Report 2007-01; Klaus Pohl, Patrick Heymans, Kyo-Chul Kang, Andreas Metzger (eds.).

[202] M. Salifu, Y. Yu, and B. Nuseibeh. Specifying Monitoring and Switching Problems in Context. In *In: Proc. 15th Intl. Conference on Requirements Engineering*, pages 211–220, 2007.

[203] D. Schall, C. Dorn, H-L. Truong, and S. Dustdar. Service-Oriented Computing - ICSOC 2008 Workshops. chapter On Supporting the Design of Human-Provided Services in SOA, pages 91–102. Springer-Verlag, Berlin, Heidelberg, 2009.

[204] B. Schilit, N. Adams, and R. Want. Context-Aware Computing Applications. In *Proceedings of the 1994 First Workshop on Mobile Computing Systems and Applications*, WMCSA '94, pages 85–90, Washington, DC, USA, 1994. IEEE Computer Society.

[205] A. Schmidt. Implicit Human Computer Interaction Through Context. *Personal and Ubiquitous Computing*, 4(2):191–199, June 2000.

[206] A. Schmidt. *Ubiquitous Computing - Computing in Context*. PhD thesis, Lancaster University, November 2002.

[207] A. Schmidt, M. Beigl, and H. Gellersen. There is More to Context than Location. *Computers and Graphics*, 23:893–901, 1998.

[208] D. Schmidt, M. Stal, H. Rohnert, and F. Buschmann. *Pattern-Oriented Software Architecture, Volume 2: Patterns for Concurrent and Networked Objects*. Wiley, 2000.

[209] B. Selic. UML for Real. chapter Architectural Patterns for Real-time Systems, pages 171–188. Kluwer Academic Publishers, Norwell, MA, USA, 2003.

[210] M. Shanahan. The Event Calculus Explained. In *Artificial Intelligence Today: Recent Trends and Developments*, pages 409–430. 1999.

[211] M. Shaw. The coming-of-age of software architecture research. In *Proceedings of the 23rd International Conference on Software Engineering*, ICSE '01, pages 656–, Washington, DC, USA, 2001. IEEE Computer Society.

[212] M. Shehab, A. Ghafoor, and E. Bertino. Secure Collaboration in a Mediator-Free Distributed Environment. *IEEE Trans. Parallel Distrib. Syst.*, 19(10):1338–1351, 2008.

[213] H. Shen and Y. Cheng. A Semantic Context-Based Model for Mobile Web Services Access Control. volume 3, pages 18–25. 2011.

[214] Y. W. Sim, C. Wang, L. Gilbert, and G. B. Wills. An overview of service-oriented architecture. Technical report, University of Southampton, July 2005.

[215] J. Simmonds, M. Chechik, S. Nejati, E. Litani, and B. O'Farrell. Runtime Verification. chapter Property Patterns for Runtime Monitoring of Web Service Conversations, pages 137–157. Springer-Verlag, Berlin, Heidelberg, 2008.

[216] A. Sirbu, A. Marconi, M. Pistore, H. Eberle, F. Leymann, and T. Unger. Dynamic Composition of Pervasive Process Fragments. In *ICWS* [1], pages 73–80.

[217] I. Sommerville. *Software Engineering*. Addison-Wesley, Harlow, England, 9. edition, 2010.

[218] G. Spanoudakis, C. Kloukinas, and K. Androutsopoulos. Towards Security Monitoring Patterns. In *Proceedings of the 2007 ACM symposium on Applied computing*, SAC '07, pages 1518–1525, New York, NY, USA, 2007. ACM.

[219] G. Spanoudakis and K. Mahbub. Requirements Monitoring for Service-Based Systems: Towards a framework based on Event Calculus. *Automated Software Engineering, International Conference on*, 0:379–384, 2004.

[220] G. Spanoudakis and K. Mahbub. Non Intrusive Monitoring of Service-based Systems. *International Journal of Cooperative Information Systems*, 15:325–358, 2006.

[221] J. Sun, X. Wu, S. Yan, L-F. Cheong, T-S. Chua, and J. Li. Hierarchical Spatio-temporal Context Modeling for Action Recognition. *Computer Vision and Pattern Recognition, IEEE Computer Society Conference on*, 0:2004–2011, 2009.

[222] V. Talwar, C. S. Shankar, R. Rafaeli, D. Milojicic, S. Iyer, K. Farkas, and Y. Chen. Adaptive monitoring automated change management for monitoring systems. In *Proceedings of the 13th Workshop of the HP OpenView University Association (HP-OVUA 2006)*, pages 21–24, 2006.

[223] V. Tan, P. Groth, S. Miles, S. Jiang, S. Munroe, S. Tsasakou, and L. Moreau. Security issues in a soa-based provenance system. In *Third International*

Provenance and Annotation Workshop. Springer, 2006. Event Dates: May 2006.

[224] R. Tarjan. Depth-first search and linear graph algorithms. *SIAM J. Comput.*, 1(2):146–160, 1972.

[225] T. Than-Tun, Y. Yu, C.B. Haley, and B. Nuseibeh. Model-Based Argument Analysis for Evolving Security Requirements. In *SSIRI*, pages 88–97. IEEE Computer Society, 2010.

[226] D. Thevenin and J. Coutaz. Adaptation and Plasticity of User Interfaces. 1999.

[227] T. Thiel. *Design and Implementation of a Service-oriented Information System Architecture Based on a Case Study(Konzeption und Realisierung einer service-orientierten IS-Architektur anhand eines Fallbeispiels)*. GRIN Verlag GmbH, 2007.

[228] D. Thvenin. ARTStudio; Tool for Multi-target UI Design.

[229] F. Tip. A Survey of Program Slicing Techniques. *Journal of Programming Languages*, 3:121–189, 1995.

[230] M.H. Tran, A. Colman, and J. Han. Service-Based Development of Context-Aware Automotive Telematics Systems. In *Proceedings of the 2010 15th IEEE International Conference on Engineering of Complex Computer Systems*, ICECCS '10, pages 53–62, Washington, DC, USA, 2010. IEEE Computer Society.

[231] G. Tretola and E. Zimeo. Autonomic Internet-scale Workflows. In *Proceedings of the 3rd International Workshop on Monitoring, Adaptation and Beyond*, MONA '10, pages 48–56, New York, NY, USA, 2010. ACM.

[232] A.K. Tripathy and M.R. Patra. An Event-based, Non-intrusive Monitoring Framework for Web Service-based Systems. In *Proceedings of 6th International Conference on Next Generation Web Service Practices: NWeSP 2010*, pages 201–206, Gwalior, India, 2010.

[233] W-T Tsai, D. Zhang, Y. Chen, H. Huang, R. Paul, and N. Liao. A software reliability model for web services. In M. H. Hamza, editor, *IASTED Conf. on Software Engineering and Applications*, pages 144–149. IASTED/ACTA Press, 2004.

[234] A. Tversky and D. Kahneman. Judgment under uncertainty: Heuristics and biases. *Science*, 185(4157):11241131, September 1974.

[235] K. Tyagi and A. Sharma. Reliability of component based systems: a critical survey. *SIGSOFT Softw. Eng. Notes*, 36(6):1–6, November 2011.

[236] N. Ubayashi and S. Nakajima. Separation of Context Concerns – Applying Aspect Orientation to VDM. In *Talk at the 2nd Overture Workshop, FM'06*, 2006.

[237] K. Vallerio, L. Zhong, and N. Jha. Energy-efficient graphical user interface design. *IEEE Trans. Mob. Comput.*, pages 846–859, 2006.

[238] F. van Harmelen, , V. Lifschitz, and B. Porter. *Handbook of Knowledge Representation*. Elsevier Science, San Diego, USA, 2007.

[239] M. Vuković. Context aware service composition. Technical Report UCAM-CL-TR-700, University of Cambridge, Computer Laboratory, October 2007.

[240] M. Waite and P. Logan. Model based user needs analysis. In *Systems Engineering and Test and Evaluation Conference (SETE2011), Canberra*, 2011.

[241] X. Wang, D. Zhang, T. Gu, and H. Pung. Ontology based context modeling and reasoning using owl. In *PerCom Workshops*, pages 18–22. IEEE Computer Society, 2004.

[242] D. Wieland, M.and Nicklas and F. Leymann. Context Model for Representation of Business Process Management Artifacts. In Chun Hua Lin and Ming Zhang, editors, *International Proceedings of Economics Development and Research: IPEDR*, volume 9 of *Economics and Business Information*, pages 46–51. IACSIT PRESS, Mai 2011.

[243] H. Yahyaoui, L. Wang, A. Mourad, M. Almulla, and Q. Sheng. Towards Context-adaptable Web Service Policies. *Procedia CS*, 5:610–617, 2011.

[244] E. Zahoor, O. Perrin, and C. Godart. An Event-Based Reasoning Approach to Web Services Monitoring. In *ICWS* [1], pages 628–635.

[245] A. Zengin, R. Kazhamiakin, and M. Pistore. CLAM: Cross-Layer Management of Adaptation Decisions for Service-Based Applications. In *ICWS* [1], pages 698–699.

[246] Y. Zhai, J. Zhang, and K. Lin. SOA Middleware Support for Service Process Reconfiguration with End-to-End QoS Constraints. In *Proceedings of the 2009 IEEE International Conference on Web Services*, ICWS '09, pages 815–822, Washington, DC, USA, 2009. IEEE Computer Society.

[247] K. Zhang. A theory for system security. In *Proceedings of the 10th IEEE workshop on Computer Security Foundations*, CSFW '97, pages 148–, Washington, DC, USA, 1997. IEEE Computer Society.

Appendix A

Wo-SBS Scenario Specification

Appendix A

Web-Organiser SBS

```
<!-- WebOrganizerv01 BPEL Process [Generated by the Eclipse BPEL Designer] -->
<bpel:process name="WebOrganizerv01"
 targetNamespace="http://weborganizerv01"
       suppressJoinFailure="yes"
       xmlns:tns="http://weborganizerv01"
       xmlns:bpel="http://docs.oasis-open.org/wsbpel/2.0/process/executable"
       xmlns:ns="http://code">

   <!-- Import the client WSDL -->
   <bpel:import namespace="http://code" location="GUIMailandAgendaServices.wsdl"
importType="http://schemas.xmlsoap.org/wsdl/"></bpel:import>
   <bpel:import namespace="http://code" location="ExtraServices.wsdl"
importType="http://schemas.xmlsoap.org/wsdl/"></bpel:import>
   <bpel:import namespace="http://code" location="GUIAgendaServices.wsdl"
importType="http://schemas.xmlsoap.org/wsdl/"></bpel:import>
   <bpel:import namespace="http://code" location="AgendaServices.wsdl"
importType="http://schemas.xmlsoap.org/wsdl/"></bpel:import>
   <bpel:import namespace="http://code" location="GUIMsgServices.wsdl"
importType="http://schemas.xmlsoap.org/wsdl/"></bpel:import>
   <bpel:import namespace="http://code" location="Shutdown.wsdl"
importType="http://schemas.xmlsoap.org/wsdl/"></bpel:import>
   <bpel:import namespace="http://code" location="GUIMailServices.wsdl"
importType="http://schemas.xmlsoap.org/wsdl/"></bpel:import>
   <bpel:import namespace="http://code" location="BasicUserOptions.wsdl"
importType="http://schemas.xmlsoap.org/wsdl/"></bpel:import>
   <bpel:import namespace="http://code" location="MailServices.wsdl"
importType="http://schemas.xmlsoap.org/wsdl/"></bpel:import>
   <bpel:import namespace="http://code" location="SelectMainFeature.wsdl"
importType="http://schemas.xmlsoap.org/wsdl/"></bpel:import>
   <bpel:import namespace="http://code" location="MessagingServices.wsdl"
importType="http://schemas.xmlsoap.org/wsdl/"></bpel:import>
   <bpel:import namespace="http://code" location="CheckAccess.wsdl"
importType="http://schemas.xmlsoap.org/wsdl/"></bpel:import>
   <bpel:import namespace="http://code" location="Login.wsdl"
importType="http://schemas.xmlsoap.org/wsdl/"></bpel:import>
   <bpel:import location="WebOrganizerv01Artifacts.wsdl" namespace="http://weborganizerv01"
              importType="http://schemas.xmlsoap.org/wsdl/" />

   <!-- =================================================================
-->
   <!-- PARTNERLINKS                                              -->
   <!-- List of services participating in this BPEL process             -->
   <!-- =================================================================
-->
   <bpel:partnerLinks>
      <!-- The 'client' role represents the requester of this service. -->
      <bpel:partnerLink name="client"
```

```
            partnerLinkType="tns:WebOrganizerv01"
            myRole="WebOrganizerv01Provider"
            />
      <bpel:partnerLink name="login" partnerLinkType="tns:loginPL"
partnerRole="loginProvider"></bpel:partnerLink>
      <bpel:partnerLink name="check access" partnerLinkType="tns:checkAccessPL"
partnerRole="checkAccessProvider"></bpel:partnerLink>
      <bpel:partnerLink name="mesaggingSS" partnerLinkType="tns:messagingSSPL"
partnerRole="messagingSSProvider"></bpel:partnerLink>
      <bpel:partnerLink name="featureSelectionWO" partnerLinkType="tns:featureSelectionWOPL"
partnerRole="SelectMainFeatureProvider"></bpel:partnerLink>
      <bpel:partnerLink name="mailSS" partnerLinkType="tns:mailSSPL"
partnerRole="MailServicesProvider"></bpel:partnerLink>
      <bpel:partnerLink name="basicUserOperations"
partnerLinkType="tns:basicUserOperationsPL"
partnerRole="basicUserOperationsProvider"></bpel:partnerLink>
      <bpel:partnerLink name="mailSSUserInteraction"
partnerLinkType="tns:mailSSUserInteractionPL"
partnerRole="GUIMailServicesProvider"></bpel:partnerLink>
      <bpel:partnerLink name="exit" partnerLinkType="tns:shutdownPL"
partnerRole="shutdown"></bpel:partnerLink>
      <bpel:partnerLink name="msgSSUserInteraction" partnerLinkType="tns:GUIMsgServicesPL"
partnerRole="GUIMsgServicesProvider"></bpel:partnerLink>
      <bpel:partnerLink name="agendaSS" partnerLinkType="tns:agendaServicesPL"
partnerRole="agendaServicesProvider"></bpel:partnerLink>
      <bpel:partnerLink name="agendaSSUserInteraction"
partnerLinkType="tns:agendaSSUserInteractionPL"
partnerRole="GUIAgendaServicesProvider"></bpel:partnerLink>
      <bpel:partnerLink name="extraSS" partnerLinkType="tns:extraSSPL"
partnerRole="extraServicesProvider"></bpel:partnerLink>
      <bpel:partnerLink name="mailandagendaUserInteraction"
partnerLinkType="tns:mailAndAgendaPL"
partnerRole="GUIMailandAgendaProvider"></bpel:partnerLink>
   </bpel:partnerLinks>

   <!-- ================================================================
-->
   <!-- VARIABLES                                                    -->
   <!-- List of messages and XML documents used within this BPEL process  -->
   <!-- ================================================================
-->
   <bpel:variables>
      <!-- Reference to the message passed as input during initiation -->
      <bpel:variable name="input"
            messageType="tns:WebOrganizerv01RequestMessage"/>

      <!--
        Reference to the message that will be returned to the requester
        -->
      <bpel:variable name="output"
            messageType="tns:WebOrganizerv01ResponseMessage"/>
      <bpel:variable name="loginResponse"
messageType="ns:opSelectFeatureuserOperationResponse"></bpel:variable>
```

```
    <bpel:variable name="loginRequest"
messageType="ns:opSelectFeatureuserOperationRequest"></bpel:variable>
    <bpel:variable name="check accessResponse"
messageType="ns:accessCheckerResponse"></bpel:variable>
    <bpel:variable name="check accessRequest"
messageType="ns:accessCheckerRequest"></bpel:variable>
    <bpel:variable name="mesagingSSResponse"
messageType="ns:enableMessagingServiceResponse"></bpel:variable>
    <bpel:variable name="mesagingSSRequest"
messageType="ns:enableMessagingServiceRequest"></bpel:variable>
    <bpel:variable name="featureSelectionWOResponse"
messageType="ns:opSelectFeatureuserOperationResponse"></bpel:variable>
    <bpel:variable name="featureSelectionWORequest"
messageType="ns:opSelectFeatureuserOperationRequest"></bpel:variable>
    <bpel:variable name="mailSSResponse"
messageType="ns:mailReviewerResponse"></bpel:variable>
    <bpel:variable name="mailSSRequest"
messageType="ns:mailReviewerRequest"></bpel:variable>
    <bpel:variable name="mailSSResponse1"
messageType="ns:mailComposerResponse"></bpel:variable>
    <bpel:variable name="mailSSRequest1"
messageType="ns:mailComposerRequest"></bpel:variable>
    <bpel:variable name="mailSSResponse2"
messageType="ns:accountManagerResponse"></bpel:variable>
    <bpel:variable name="mailSSRequest2"
messageType="ns:accountManagerRequest"></bpel:variable>
    <bpel:variable name="basicUserOperationsResponse"
messageType="ns:opSelectMailExpertiseLeveluserOperationResponse"></bpel:variable>
    <bpel:variable name="basicUserOperationsRequest"
messageType="ns:opSelectMailExpertiseLeveluserOperationRequest"></bpel:variable>
    <bpel:variable name="mailSSUserInteractionResponse"
messageType="ns:opStandardMailuserOperationResponse">
        <bpel:from>
        </bpel:from>
    </bpel:variable>
    <bpel:variable name="mailSSUserInteractionRequest"
messageType="ns:opStandardMailuserOperationRequest"></bpel:variable>
    <bpel:variable name="mailSSUserInteractionResponse1"
messageType="ns:opAdvancedMailuserOperationResponse"></bpel:variable>
    <bpel:variable name="mailSSUserInteractionRequest1"
messageType="ns:opAdvancedMailuserOperationRequest"></bpel:variable>
    <bpel:variable name="exitResponse"
messageType="ns:closeEverythingResponse"></bpel:variable>
    <bpel:variable name="exitRequest"
messageType="ns:closeEverythingRequest"></bpel:variable>
    <bpel:variable name="mesagingSSResponse1"
messageType="ns:disableMessagingServiceResponse"></bpel:variable>
    <bpel:variable name="mesagingSSRequest1"
messageType="ns:disableMessagingServiceRequest"></bpel:variable>
    <bpel:variable name="msgSSUserInteractionResponse"
messageType="ns:opGUIuserOperationResponse"></bpel:variable>
    <bpel:variable name="msgSSUserInteractionRequest"
messageType="ns:opGUIuserOperationRequest"></bpel:variable>
```

```
        <bpel:variable name="agendaSSResponse"
messageType="ns:initiateCalendarResponse"></bpel:variable>
        <bpel:variable name="agendaSSRequest"
messageType="ns:initiateCalendarRequest"></bpel:variable>
        <bpel:variable name="agendaSSResponse1"
messageType="ns:initiateSchedulerResponse"></bpel:variable>
        <bpel:variable name="agendaSSRequest1"
messageType="ns:initiateSchedulerRequest"></bpel:variable>
        <bpel:variable name="basicUserOperationsResponse1"
messageType="ns:opSelectNeeduserOperationResponse"></bpel:variable>
        <bpel:variable name="basicUserOperationsRequest1"
messageType="ns:opSelectNeeduserOperationRequest"></bpel:variable>
        <bpel:variable name="agendaSSResponse2"
messageType="ns:getLatestDatesResponse"></bpel:variable>
        <bpel:variable name="agendaSSRequest2"
messageType="ns:getLatestDatesRequest"></bpel:variable>
        <bpel:variable name="agendaSSResponse3"
messageType="ns:getLatestAnnotationsResponse"></bpel:variable>
        <bpel:variable name="agendaSSRequest3"
messageType="ns:getLatestAnnotationsRequest"></bpel:variable>
        <bpel:variable name="agendaSSUserInteractionResponse"
messageType="ns:opAgendauserOperationResponse"></bpel:variable>
        <bpel:variable name="agendaSSUserInteractionRequest"
messageType="ns:opAgendauserOperationRequest"></bpel:variable>
        <bpel:variable name="extraSSResponse"
messageType="ns:initiateCrossReferencingResponse"></bpel:variable>
        <bpel:variable name="extraSSRequest"
messageType="ns:initiateCrossReferencingRequest"></bpel:variable>
        <bpel:variable name="mailandagendaUserInteractionResponse"
messageType="ns:opMailandAgendauserOperationResponse"></bpel:variable>
        <bpel:variable name="mailandagendaUserInteractionRequest"
messageType="ns:opMailandAgendauserOperationRequest"></bpel:variable>
    </bpel:variables>

    <!-- ================================================================
-->
    <!-- ORCHESTRATION LOGIC                                          -->
    <!-- Set of activities coordinating the flow of messages across the    -->
    <!-- services integrated within this business process               -->
    <!-- ================================================================
-->
    <bpel:sequence name="main">

        <!-- Receive input from requester.
             Note: This maps to operation defined in WebOrganizerv01.wsdl
             -->
        <bpel:receive name="receiveInput" partnerLink="client"
                portType="tns:WebOrganizerv01"
                operation="process" variable="input"
                createInstance="yes"/>
        <!-- Generate reply to synchronous request -->
        <bpel:assign validate="no" name="Assign1">
            <bpel:copy>
        <bpel:from>
          <bpel:literal xml:space="preserve"><impl:opSelectFeatureuserOperation
xmlns:impl="http://code" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <impl:str></impl:str>
</impl:opSelectFeatureuserOperation>
</bpel:literal>
        </bpel:from>
        <bpel:to variable="loginRequest" part="parameters"></bpel:to>
      </bpel:copy>
      <bpel:copy>
        <bpel:from part="payload" variable="input">
          <bpel:query queryLanguage="urn:oasis:names:tc:wsbpel:2.0:sublang:xpath1.0"><!
[CDATA[tns:input]]></bpel:query>
        </bpel:from>
        <bpel:to part="parameters" variable="loginRequest">
          <bpel:query queryLanguage="urn:oasis:names:tc:wsbpel:2.0:sublang:xpath1.0"><!
[CDATA[ns:str]]></bpel:query>
        </bpel:to>
      </bpel:copy>
    </bpel:assign>
    <bpel:invoke name="Login" partnerLink="login" operation="opSelectFeatureuserOperation"
portType="ns:Login" inputVariable="loginRequest"
outputVariable="loginResponse"></bpel:invoke>

    <bpel:assign validate="no" name="Assign2">
      <bpel:copy>
        <bpel:from>
          <bpel:literal xml:space="preserve"><impl:accessChecker xmlns:impl="http://code"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <impl:str></impl:str>
</impl:accessChecker>
</bpel:literal>
        </bpel:from>
        <bpel:to variable="check accessRequest" part="parameters"></bpel:to>
      </bpel:copy>
      <bpel:copy>
        <bpel:from part="parameters" variable="loginResponse">
          <bpel:query queryLanguage="urn:oasis:names:tc:wsbpel:2.0:sublang:xpath1.0"><!
[CDATA[ns:opSelectFeatureuserOperationReturn]]></bpel:query>
        </bpel:from>
        <bpel:to part="parameters" variable="check accessRequest">
          <bpel:query queryLanguage="urn:oasis:names:tc:wsbpel:2.0:sublang:xpath1.0"><!
[CDATA[ns:str]]></bpel:query>
        </bpel:to>
      </bpel:copy>
    </bpel:assign>
    <bpel:invoke name="Check Access" partnerLink="check access" operation="accessChecker"
portType="ns:CheckAccess" inputVariable="check accessRequest" outputVariable="check
accessResponse"></bpel:invoke>
    <bpel:assign validate="no" name="Assign3">


      <bpel:copy>
```

```
            <bpel:from>
               <bpel:literal xml:space="preserve"><impl:enableMessagingService
xmlns:impl="http://code" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <impl:str></impl:str>
</impl:enableMessagingService>
</bpel:literal>
            </bpel:from>
            <bpel:to variable="mesaggingSSRequest" part="parameters"></bpel:to>
         </bpel:copy>
         <bpel:copy>
            <bpel:from part="parameters" variable="check accessResponse">
               <bpel:query queryLanguage="urn:oasis:names:tc:wsbpel:2.0:sublang:xpath1.0"><!
[CDATA[ns:accessCheckerReturn]]></bpel:query>
            </bpel:from>
            <bpel:to part="parameters" variable="mesaggingSSRequest">
               <bpel:query queryLanguage="urn:oasis:names:tc:wsbpel:2.0:sublang:xpath1.0"><!
[CDATA[ns:str]]></bpel:query>
            </bpel:to>
         </bpel:copy>
      </bpel:assign>
      <bpel:invoke name="MessagingSS" partnerLink="mesaggingSS"
operation="enableMessagingService" portType="ns:MessagingServices"
inputVariable="mesaggingSSRequest" outputVariable="mesaggingSSResponse"></bpel:invoke>
      <bpel:assign validate="no" name="Assign4">
         <bpel:copy>
            <bpel:from>
               <bpel:literal xml:space="preserve"><impl:opSelectFeatureuserOperation
xmlns:impl="http://code" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <impl:str></impl:str>
</impl:opSelectFeatureuserOperation>
</bpel:literal>
            </bpel:from>
            <bpel:to variable="featureSelectionWORequest" part="parameters"></bpel:to>
         </bpel:copy>
         <bpel:copy>
            <bpel:from part="parameters" variable="mesaggingSSResponse">
               <bpel:query queryLanguage="urn:oasis:names:tc:wsbpel:2.0:sublang:xpath1.0"><!
[CDATA[ns:enableMessagingServiceReturn]]></bpel:query>
            </bpel:from>
            <bpel:to part="parameters" variable="featureSelectionWORequest">
               <bpel:query queryLanguage="urn:oasis:names:tc:wsbpel:2.0:sublang:xpath1.0"><!
[CDATA[ns:str]]></bpel:query>
            </bpel:to>
         </bpel:copy>
      </bpel:assign>
      <bpel:invoke name="WO Feature Selection" partnerLink="featureSelectionWO"
operation="opSelectFeatureuserOperation" portType="ns:SelectMainFeature"
inputVariable="featureSelectionWORequest"
outputVariable="featureSelectionWOResponse"></bpel:invoke>
      <bpel:if name="Main Feature WO">
         <bpel:condition><!
[CDATA[contains($featureSelectionWOResponse.parameters/ns:opSelectFeatureuserOperationRetur
n, '@email@')]]></bpel:condition>
```

```
      <bpel:sequence name="Sequence0">
        <bpel:assign validate="no" name="Assign5">
          <bpel:copy>
            <bpel:from>
              <bpel:literal xml:space="preserve"><impl:mailReviewer xmlns:impl="http://code"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
 <impl:str></impl:str>
</impl:mailReviewer>
</bpel:literal>
            </bpel:from>
            <bpel:to variable="mailSSRequest" part="parameters"></bpel:to>
          </bpel:copy>
          <bpel:copy>
            <bpel:from part="parameters" variable="featureSelectionWOResponse">
              <bpel:query queryLanguage="urn:oasis:names:tc:wsbpel:2.0:sublang:xpath1.0">
                <![CDATA[ns:opSelectFeatureuserOperationReturn]]>
              </bpel:query>
            </bpel:from>
            <bpel:to part="parameters" variable="mailSSRequest">
              <bpel:query queryLanguage="urn:oasis:names:tc:wsbpel:2.0:sublang:xpath1.0"><!
[CDATA[ns:str]]></bpel:query>
            </bpel:to>
          </bpel:copy>
        </bpel:assign>
        <bpel:invoke name="Mail SS (review)" partnerLink="mailSS" operation="mailReviewer"
portType="ns:MailServices" inputVariable="mailSSRequest"
outputVariable="mailSSResponse"></bpel:invoke>
        <bpel:assign validate="no" name="Assign6">
          <bpel:copy>
            <bpel:from>
              <bpel:literal xml:space="preserve"><impl:mailComposer xmlns:impl="http://code"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
 <impl:str></impl:str>
</impl:mailComposer>
</bpel:literal>
            </bpel:from>
            <bpel:to variable="mailSSRequest1" part="parameters"></bpel:to>
          </bpel:copy>
          <bpel:copy>
            <bpel:from part="parameters" variable="mailSSResponse">
              <bpel:query queryLanguage="urn:oasis:names:tc:wsbpel:2.0:sublang:xpath1.0"><!
[CDATA[ns:mailReviewerReturn]]></bpel:query>
            </bpel:from>
            <bpel:to part="parameters" variable="mailSSRequest1">
              <bpel:query queryLanguage="urn:oasis:names:tc:wsbpel:2.0:sublang:xpath1.0"><!
[CDATA[ns:str]]></bpel:query>
            </bpel:to>
          </bpel:copy>
        </bpel:assign>
        <bpel:invoke name="Mail SS (composer)" partnerLink="mailSS"
operation="mailComposer" portType="ns:MailServices" inputVariable="mailSSRequest1"
outputVariable="mailSSResponse1"></bpel:invoke>
        <bpel:assign validate="no" name="Assign7">
```

```
          <bpel:copy>
            <bpel:from>
              <bpel:literal xml:space="preserve"><impl:accountManager
xmlns:impl="http://code" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <impl:str></impl:str>
</impl:accountManager>
</bpel:literal>
            </bpel:from>
            <bpel:to variable="mailSSRequest2" part="parameters"></bpel:to>
          </bpel:copy>
          <bpel:copy>
            <bpel:from part="parameters" variable="mailSSResponse1">
              <bpel:query queryLanguage="urn:oasis:names:tc:wsbpel:2.0:sublang:xpath1.0"><!
[CDATA[ns:mailComposerReturn]]></bpel:query>
            </bpel:from>
            <bpel:to part="parameters" variable="mailSSRequest2">
              <bpel:query queryLanguage="urn:oasis:names:tc:wsbpel:2.0:sublang:xpath1.0"><!
[CDATA[ns:str]]></bpel:query>
            </bpel:to>
          </bpel:copy>
        </bpel:assign>
        <bpel:invoke name="Mail SS (management)" partnerLink="mailSS"
operation="accountManager" portType="ns:MailServices" inputVariable="mailSSRequest2"
outputVariable="mailSSResponse2"></bpel:invoke>
        <bpel:assign validate="no" name="Assign8">
          <bpel:copy>
            <bpel:from>
              <bpel:literal
xml:space="preserve"><impl:opSelectMailExpertiseLeveluserOperation xmlns:impl="http://code"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <impl:str></impl:str>
</impl:opSelectMailExpertiseLeveluserOperation>
</bpel:literal>
            </bpel:from>
            <bpel:to variable="basicUserOperationsRequest" part="parameters"></bpel:to>
          </bpel:copy>
          <bpel:copy>
            <bpel:from part="parameters" variable="mailSSResponse2">
              <bpel:query queryLanguage="urn:oasis:names:tc:wsbpel:2.0:sublang:xpath1.0"><!
[CDATA[ns:accountManagerReturn]]></bpel:query>
            </bpel:from>
            <bpel:to part="parameters" variable="basicUserOperationsRequest">
              <bpel:query queryLanguage="urn:oasis:names:tc:wsbpel:2.0:sublang:xpath1.0"><!
[CDATA[ns:str]]></bpel:query>
            </bpel:to>
          </bpel:copy>
        </bpel:assign>
        <bpel:invoke name="Basic User Operations" partnerLink="basicUserOperations"
operation="opSelectMailExpertiseLeveluserOperation" portType="ns:BasicUserOptions"
inputVariable="basicUserOperationsRequest"
outputVariable="basicUserOperationsResponse"></bpel:invoke>
        <bpel:if name="Level of Skills">
          <bpel:condition><!
```

```
[CDATA[contains($basicUserOperationsResponse.parameters/ns:opSelectMailExpertiseLeveluserO
perationReturn, '@EmailSkillLL@')]]></bpel:condition>
          <bpel:sequence name="Sequence1">
            <bpel:assign validate="no" name="Assign9"><bpel:copy>
                <bpel:from>
                  <bpel:literal xml:space="preserve"><impl:opStandardMailuserOperation
xmlns:impl="http://code" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <impl:str></impl:str>
</impl:opStandardMailuserOperation>
</bpel:literal>
                </bpel:from>
                <bpel:to variable="mailSSUserInteractionRequest"
part="parameters"></bpel:to>
              </bpel:copy>
              <bpel:copy>
                <bpel:from part="parameters" variable="basicUserOperationsResponse">
                  <bpel:query
queryLanguage="urn:oasis:names:tc:wsbpel:2.0:sublang:xpath1.0"><!
[CDATA[ns:opSelectMailExpertiseLeveluserOperationReturn]]></bpel:query>
                </bpel:from>
                <bpel:to part="parameters" variable="mailSSUserInteractionRequest">
                  <bpel:query
queryLanguage="urn:oasis:names:tc:wsbpel:2.0:sublang:xpath1.0"><!
[CDATA[ns:str]]></bpel:query>
                </bpel:to>
              </bpel:copy>
            </bpel:assign>
            <bpel:repeatUntil name="Standard Mail Op">
              <bpel:sequence name="Sequence2">
                <bpel:invoke name="Standard UI Mail" partnerLink="mailSSUserInteraction"
operation="opStandardMailuserOperation" inputVariable="mailSSUserInteractionRequest"
outputVariable="mailSSUserInteractionResponse"></bpel:invoke>
                <bpel:assign validate="no" name="Standard Mail O/I">
                  <bpel:copy>
                    <bpel:from>
                      <bpel:literal xml:space="preserve"><impl:opStandardMailuserOperation
xmlns:impl="http://code" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <impl:str></impl:str>
</impl:opStandardMailuserOperation>
</bpel:literal>
                    </bpel:from>
                    <bpel:to variable="mailSSUserInteractionRequest"
part="parameters"></bpel:to>
                  </bpel:copy>
                  <bpel:copy>
                    <bpel:from part="parameters" variable="mailSSUserInteractionResponse">
                      <bpel:query
queryLanguage="urn:oasis:names:tc:wsbpel:2.0:sublang:xpath1.0"><!
[CDATA[ns:opStandardMailuserOperationReturn]]></bpel:query>
                    </bpel:from>
                    <bpel:to part="parameters" variable="mailSSUserInteractionRequest">
                      <bpel:query
queryLanguage="urn:oasis:names:tc:wsbpel:2.0:sublang:xpath1.0"><!
```

```
[CDATA[ns:str]]></bpel:query>
                              </bpel:to>
                           </bpel:copy>
                        </bpel:assign>
                        <bpel:if name="Enable disable Messaging">
                           <bpel:condition><!
[CDATA[contains($mailSSUserInteractionResponse.parameters/ns:opStandardMailuserOperationRe
turn, '@messagingServiceEnabled@')]]></bpel:condition>
                           <bpel:sequence name="Sequence3">
                              <bpel:assign validate="no" name="Assign10">
                                 <bpel:copy>
                                    <bpel:from>
                                       <bpel:literal xml:space="preserve"><impl:enableMessagingService
xmlns:impl="http://code" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <impl:str></impl:str>
</impl:enableMessagingService>
</bpel:literal>
                                    </bpel:from>
                                    <bpel:to variable="mesaggingSSRequest"
part="parameters"></bpel:to>
                                 </bpel:copy>
                                 <bpel:copy>
                                    <bpel:from part="parameters"
variable="mailSSUserInteractionResponse">
                                       <bpel:query
queryLanguage="urn:oasis:names:tc:wsbpel:2.0:sublang:xpath1.0"><!
[CDATA[ns:opStandardMailuserOperationReturn]]></bpel:query>
                                    </bpel:from>
                                    <bpel:to part="parameters" variable="mesaggingSSRequest">
                                       <bpel:query
queryLanguage="urn:oasis:names:tc:wsbpel:2.0:sublang:xpath1.0"><!
[CDATA[ns:str]]></bpel:query>
                                    </bpel:to>
                                 </bpel:copy>
                              </bpel:assign>
                              <bpel:invoke name="MessagingSS" partnerLink="mesaggingSS"
operation="enableMessagingService" portType="ns:MessagingServices"
inputVariable="mesaggingSSRequest" outputVariable="mesaggingSSResponse"></bpel:invoke>
                              <bpel:assign validate="no" name="Assign11">
                                 <bpel:copy>
                                    <bpel:from>
                                       <bpel:literal xml:space="preserve"><impl:closeEverything
xmlns:impl="http://code" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <impl:str></impl:str>
</impl:closeEverything>
</bpel:literal>
                                    </bpel:from>
                                    <bpel:to variable="exitRequest" part="parameters"></bpel:to>
                                 </bpel:copy>
                                 <bpel:copy>
                                    <bpel:from part="parameters" variable="mesaggingSSResponse">
                                       <bpel:query
queryLanguage="urn:oasis:names:tc:wsbpel:2.0:sublang:xpath1.0"><!
```

```
[CDATA[ns:enableMessagingServiceReturn]]></bpel:query>
                                    </bpel:from>
                                    <bpel:to part="parameters" variable="exitRequest">
                                       <bpel:query
queryLanguage="urn:oasis:names:tc:wsbpel:2.0:sublang:xpath1.0"><!
[CDATA[ns:str]]></bpel:query>
                                    </bpel:to>
                                 </bpel:copy>
                              </bpel:assign>
                           </bpel:sequence>
                           <bpel:else>
                              <bpel:sequence name="Sequence4">
                                 <bpel:assign validate="no" name="Assign13">
                                    <bpel:copy>
                                       <bpel:from>
                                          <bpel:literal
xml:space="preserve"><impl:disableMessagingService xmlns:impl="http://code"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <impl:str></impl:str>
</impl:disableMessagingService>
</bpel:literal>
                                       </bpel:from>
                                       <bpel:to variable="mesaggingSSRequest1"
part="parameters"></bpel:to>
                                    </bpel:copy>
                                    <bpel:copy>
                                       <bpel:from part="parameters"
variable="mailSSUserInteractionResponse">
                                          <bpel:query
queryLanguage="urn:oasis:names:tc:wsbpel:2.0:sublang:xpath1.0"><!
[CDATA[ns:opStandardMailuserOperationReturn]]></bpel:query>
                                       </bpel:from>
                                       <bpel:to part="parameters" variable="mesaggingSSRequest1">
                                          <bpel:query
queryLanguage="urn:oasis:names:tc:wsbpel:2.0:sublang:xpath1.0"><!
[CDATA[ns:str]]></bpel:query>
                                       </bpel:to>
                                    </bpel:copy>
                                 </bpel:assign>
                                 <bpel:invoke name="MessagingSS" partnerLink="mesaggingSS"
operation="disableMessagingService" portType="ns:MessagingServices"
inputVariable="mesaggingSSRequest1" outputVariable="mesaggingSSResponse1"></bpel:invoke>
                                 <bpel:assign validate="no" name="Assign14">

                                    <bpel:copy>
                                       <bpel:from>
                                          <bpel:literal xml:space="preserve"><impl:closeEverything
xmlns:impl="http://code" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <impl:str></impl:str>
</impl:closeEverything>
</bpel:literal>
                                       </bpel:from>
                                       <bpel:to variable="exitRequest" part="parameters"></bpel:to>
```

```
                        </bpel:copy>
                        <bpel:copy>
                          <bpel:from part="parameters" variable="mesaggingSSResponse1">
                            <bpel:query
queryLanguage="urn:oasis:names:tc:wsbpel:2.0:sublang:xpath1.0"><!
[CDATA[ns:disableMessagingServiceReturn]]></bpel:query>
                          </bpel:from>
                          <bpel:to part="parameters" variable="exitRequest">
                            <bpel:query
queryLanguage="urn:oasis:names:tc:wsbpel:2.0:sublang:xpath1.0"><!
[CDATA[ns:str]]></bpel:query>
                          </bpel:to>
                        </bpel:copy>
                      </bpel:assign>
                    </bpel:sequence>
                  </bpel:else>
                </bpel:if>
              </bpel:sequence>
              <bpel:condition><!
[CDATA[contains($mailSSUserInteractionResponse.parameters/ns:opStandardMailuserOperationRe
turn, '@end-GUI-Mail-Services@')
]]></bpel:condition>
            </bpel:repeatUntil>
          </bpel:sequence>
          <bpel:elseif>
            <bpel:condition><!
[CDATA[contains($basicUserOperationsResponse.parameters/ns:opSelectMailExpertiseLeveluserO
perationReturn, '@EmailSkillHH@')]]></bpel:condition>
            <bpel:sequence name="Sequence5">
              <bpel:assign validate="no" name="Assign15">


                <bpel:copy>
                  <bpel:from>
                    <bpel:literal xml:space="preserve"><impl:opAdvancedMailuserOperation
xmlns:impl="http://code" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
 <impl:str></impl:str>
</impl:opAdvancedMailuserOperation>
</bpel:literal>
                  </bpel:from>
                  <bpel:to variable="mailSSUserInteractionRequest1"
part="parameters"></bpel:to>
                </bpel:copy>
                <bpel:copy>
                  <bpel:from part="parameters" variable="basicUserOperationsResponse">
                    <bpel:query
queryLanguage="urn:oasis:names:tc:wsbpel:2.0:sublang:xpath1.0"><!
[CDATA[ns:opSelectMailExpertiseLeveluserOperationReturn]]></bpel:query>
                  </bpel:from>
                  <bpel:to part="parameters" variable="mailSSUserInteractionRequest1">
                    <bpel:query
queryLanguage="urn:oasis:names:tc:wsbpel:2.0:sublang:xpath1.0"><!
[CDATA[ns:str]]></bpel:query>
```

```
                  </bpel:to>
                </bpel:copy>
              </bpel:assign>
              <bpel:repeatUntil name="Advanced Mail Op">
                <bpel:sequence name="Sequence6">
                  <bpel:invoke name="Advanced Options"
partnerLink="mailSSUserInteraction" operation="opAdvancedMailuserOperation"
portType="ns:GUIMailServices" inputVariable="mailSSUserInteractionRequest1"
outputVariable="mailSSUserInteractionResponse1"></bpel:invoke>
                  <bpel:assign validate="no" name="Assign16"><bpel:copy>
                  <bpel:from>
                    <bpel:literal xml:space="preserve"><impl:closeEverything
xmlns:impl="http://code" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <impl:str></impl:str>
</impl:closeEverything>
</bpel:literal>
                  </bpel:from>
                  <bpel:to variable="exitRequest" part="parameters"></bpel:to>
                </bpel:copy>
                <bpel:copy>
                  <bpel:from part="parameters" variable="mailSSUserInteractionResponse1">
                    <bpel:query
queryLanguage="urn:oasis:names:tc:wsbpel:2.0:sublang:xpath1.0"><!
[CDATA[ns:opAdvancedMailuserOperationReturn]]></bpel:query>
                  </bpel:from>
                  <bpel:to part="parameters" variable="exitRequest">
                    <bpel:query
queryLanguage="urn:oasis:names:tc:wsbpel:2.0:sublang:xpath1.0"><!
[CDATA[ns:str]]></bpel:query>
                  </bpel:to>
                </bpel:copy>

                  </bpel:assign>
                </bpel:sequence>
                <bpel:condition><!
[CDATA[contains($mailSSUserInteractionResponse1.parameters/ns:opAdvancedMailuserOperation
Return, '@end-GUI-Mail-Services@')]]></bpel:condition>
              </bpel:repeatUntil>
            </bpel:sequence>
          </bpel:elseif></bpel:if>
      </bpel:sequence><bpel:elseif>
        <bpel:condition><!
[CDATA[contains($featureSelectionWOResponse.parameters/ns:opSelectFeatureuserOperationRetur
n, '@message@')]]></bpel:condition>
        <bpel:sequence name="Sequence7">
          <bpel:assign validate="no" name="Assign17">
            <bpel:copy>
              <bpel:from>
                <bpel:literal xml:space="preserve"><impl:opGUIuserOperation
xmlns:impl="http://code" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <impl:str></impl:str>
</impl:opGUIuserOperation>
</bpel:literal>
```

```
                  </bpel:from>
                  <bpel:to variable="msgSSUserInteractionRequest" part="parameters"></bpel:to>
               </bpel:copy>
               <bpel:copy>
                  <bpel:from part="parameters" variable="featureSelectionWOResponse">
                     <bpel:query
queryLanguage="urn:oasis:names:tc:wsbpel:2.0:sublang:xpath1.0"><!
[CDATA[ns:opSelectFeatureuserOperationReturn]]></bpel:query>
                  </bpel:from>
                  <bpel:to part="parameters" variable="msgSSUserInteractionRequest">
                     <bpel:query
queryLanguage="urn:oasis:names:tc:wsbpel:2.0:sublang:xpath1.0"><!
[CDATA[ns:str]]></bpel:query>
                  </bpel:to>
               </bpel:copy>
            </bpel:assign>
            <bpel:repeatUntil name="RepeatUntil">
               <bpel:sequence name="Sequence8">
                  <bpel:invoke name="Message UI SS" partnerLink="msgSSUserInteraction"
operation="opGUIuserOperation" portType="ns:GUIMsgServices"
inputVariable="msgSSUserInteractionRequest"
outputVariable="msgSSUserInteractionResponse"></bpel:invoke>

                  <bpel:assign validate="no" name="Assign18">
                     <bpel:copy>
                        <bpel:from>
                           <bpel:literal xml:space="preserve"><impl:disableMessagingService
xmlns:impl="http://code" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <impl:str></impl:str>
</impl:disableMessagingService>
</bpel:literal>
                        </bpel:from>
                        <bpel:to variable="mesaggingSSRequest1" part="parameters"></bpel:to>
                     </bpel:copy>
                     <bpel:copy>
                        <bpel:from part="parameters" variable="msgSSUserInteractionResponse">
                           <bpel:query
queryLanguage="urn:oasis:names:tc:wsbpel:2.0:sublang:xpath1.0"><!
[CDATA[ns:opGUIuserOperationReturn]]></bpel:query>
                        </bpel:from>
                        <bpel:to part="parameters" variable="mesaggingSSRequest1">
                           <bpel:query
queryLanguage="urn:oasis:names:tc:wsbpel:2.0:sublang:xpath1.0"><!
[CDATA[ns:str]]></bpel:query>
                        </bpel:to>
                     </bpel:copy>
                  </bpel:assign>
               </bpel:sequence>
               <bpel:condition><!
[CDATA[contains($msgSSUserInteractionResponse.parameters/ns:opGUIuserOperationReturn,
'@messagingServiceDisabled@')]]></bpel:condition>
            </bpel:repeatUntil>
            <bpel:invoke name="MessagingSS" partnerLink="mesaggingSS"
```

```
operation="disableMessagingService" portType="ns:MessagingServices"
inputVariable="mesaggingSSRequest1" outputVariable="mesaggingSSResponse1"></bpel:invoke>
          <bpel:assign validate="no" name="Assign19">
            <bpel:copy>
              <bpel:from>
                <bpel:literal xml:space="preserve"><impl:closeEverything
xmlns:impl="http://code" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <impl:str></impl:str>
</impl:closeEverything>
</bpel:literal>
              </bpel:from>
              <bpel:to variable="exitRequest" part="parameters"></bpel:to>
            </bpel:copy>
            <bpel:copy>
              <bpel:from part="parameters" variable="mesaggingSSResponse1">
                <bpel:query
queryLanguage="urn:oasis:names:tc:wsbpel:2.0:sublang:xpath1.0"><!
[CDATA[ns:disableMessagingServiceReturn]]></bpel:query>
              </bpel:from>
              <bpel:to part="parameters" variable="exitRequest">
                <bpel:query
queryLanguage="urn:oasis:names:tc:wsbpel:2.0:sublang:xpath1.0"><!
[CDATA[ns:str]]></bpel:query>
              </bpel:to>
            </bpel:copy>
          </bpel:assign>
        </bpel:sequence>
      </bpel:elseif><bpel:elseif>
        <bpel:condition><!
[CDATA[contains($featureSelectionWOResponse.parameters/ns:opSelectFeatureuserOperationRetur
n, '@agenda@')]]></bpel:condition>
        <bpel:sequence name="Sequence9">
          <bpel:assign validate="no" name="Assign20">
            <bpel:copy>
              <bpel:from>
                <bpel:literal xml:space="preserve"><impl:initiateCalendar
xmlns:impl="http://code" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <impl:str></impl:str>
</impl:initiateCalendar>
</bpel:literal>
              </bpel:from>
              <bpel:to variable="agendaSSRequest" part="parameters"></bpel:to>
            </bpel:copy>
            <bpel:copy>
              <bpel:from part="parameters" variable="featureSelectionWOResponse">
                <bpel:query
queryLanguage="urn:oasis:names:tc:wsbpel:2.0:sublang:xpath1.0"><!
[CDATA[ns:opSelectFeatureuserOperationReturn]]></bpel:query>
              </bpel:from>
              <bpel:to part="parameters" variable="agendaSSRequest">
                <bpel:query
queryLanguage="urn:oasis:names:tc:wsbpel:2.0:sublang:xpath1.0"><!
[CDATA[ns:str]]></bpel:query>
```

```
                      </bpel:to>
                   </bpel:copy>
                </bpel:assign>
                <bpel:invoke name="Agenda SS (calendar)" partnerLink="agendaSS"
operation="initiateCalendar" portType="ns:AgendaServices" inputVariable="agendaSSRequest"
outputVariable="agendaSSResponse"></bpel:invoke>
                <bpel:assign validate="no" name="Assign21">
                   <bpel:copy>
                      <bpel:from>
                         <bpel:literal xml:space="preserve"><impl:initiateScheduler
xmlns:impl="http://code" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <impl:str></impl:str>
</impl:initiateScheduler>
</bpel:literal>
                      </bpel:from>
                      <bpel:to variable="agendaSSRequest1" part="parameters"></bpel:to>
                   </bpel:copy>
                   <bpel:copy>
                      <bpel:from part="parameters" variable="agendaSSResponse">
                         <bpel:query
queryLanguage="urn:oasis:names:tc:wsbpel:2.0:sublang:xpath1.0"><!
[CDATA[ns:initiateCalendarReturn]]></bpel:query>
                      </bpel:from>
                      <bpel:to part="parameters" variable="agendaSSRequest1">
                         <bpel:query
queryLanguage="urn:oasis:names:tc:wsbpel:2.0:sublang:xpath1.0"><!
[CDATA[ns:str]]></bpel:query>
                      </bpel:to>
                   </bpel:copy>
                </bpel:assign>
                <bpel:invoke name="Agenda SS (scheduler)" partnerLink="agendaSS"
operation="initiateScheduler" portType="ns:AgendaServices" inputVariable="agendaSSRequest1"
outputVariable="agendaSSResponse1"></bpel:invoke>
                <bpel:assign validate="no" name="Assign22">
                   <bpel:copy>
                      <bpel:from>
                         <bpel:literal xml:space="preserve"><impl:opSelectNeeduserOperation
xmlns:impl="http://code" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <impl:str></impl:str>
</impl:opSelectNeeduserOperation>
</bpel:literal>
                      </bpel:from>
                      <bpel:to variable="basicUserOperationsRequest1" part="parameters"></bpel:to>
                   </bpel:copy>
                   <bpel:copy>
                      <bpel:from part="parameters" variable="agendaSSResponse1">
                         <bpel:query
queryLanguage="urn:oasis:names:tc:wsbpel:2.0:sublang:xpath1.0"><!
[CDATA[ns:initiateSchedulerReturn]]></bpel:query>
                      </bpel:from>
                      <bpel:to part="parameters" variable="basicUserOperationsRequest1">
                         <bpel:query
queryLanguage="urn:oasis:names:tc:wsbpel:2.0:sublang:xpath1.0"><!
```

```
[CDATA[ns:str]]></bpel:query>
            </bpel:to>
          </bpel:copy>
        </bpel:assign>

        <bpel:invoke name="Need Specification" partnerLink="basicUserOperations"
operation="opSelectNeeduserOperation" portType="ns:BasicUserOptions"
inputVariable="basicUserOperationsRequest1"
outputVariable="basicUserOperationsResponse1"></bpel:invoke>
        <bpel:if name="Up-to-Date">
          <bpel:condition><!
[CDATA[contains($basicUserOperationsResponse1.parameters/ns:opSelectNeeduserOperationRetur
n,'@up-to-date-YES@')]]></bpel:condition>
          <bpel:sequence name="Sequence10">
            <bpel:assign validate="no" name="Assign23">
              <bpel:copy>
                <bpel:from>
                  <bpel:literal xml:space="preserve"><impl:getLatestDates
xmlns:impl="http://code" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <impl:str></impl:str>
</impl:getLatestDates>
</bpel:literal>
                </bpel:from>
                <bpel:to variable="agendaSSRequest2" part="parameters"></bpel:to>
              </bpel:copy>
              <bpel:copy>
                <bpel:from part="parameters" variable="basicUserOperationsResponse1">
                  <bpel:query
queryLanguage="urn:oasis:names:tc:wsbpel:2.0:sublang:xpath1.0"><!
[CDATA[ns:opSelectNeeduserOperationReturn]]></bpel:query>
                </bpel:from>
                <bpel:to part="parameters" variable="agendaSSRequest2">
                  <bpel:query
queryLanguage="urn:oasis:names:tc:wsbpel:2.0:sublang:xpath1.0"><!
[CDATA[ns:str]]></bpel:query>
                </bpel:to>
              </bpel:copy>
            </bpel:assign>
            <bpel:invoke name="Agenda SS (dates)" partnerLink="agendaSS"
operation="getLatestDates" portType="ns:AgendaServices" inputVariable="agendaSSRequest2"
outputVariable="agendaSSResponse2"></bpel:invoke>
            <bpel:assign validate="no" name="Assign24">
              <bpel:copy>
                <bpel:from>
                  <bpel:literal xml:space="preserve"><impl:getLatestAnnotations
xmlns:impl="http://code" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <impl:str></impl:str>
</impl:getLatestAnnotations>
</bpel:literal>
                </bpel:from>
                <bpel:to variable="agendaSSRequest3" part="parameters"></bpel:to>
              </bpel:copy>
              <bpel:copy>
```

```
                    <bpel:from part="parameters" variable="agendaSSResponse2">
                      <bpel:query
queryLanguage="urn:oasis:names:tc:wsbpel:2.0:sublang:xpath1.0"><!
[CDATA[ns:getLatestDatesReturn]]></bpel:query>
                    </bpel:from>
                    <bpel:to part="parameters" variable="agendaSSRequest3">
                      <bpel:query
queryLanguage="urn:oasis:names:tc:wsbpel:2.0:sublang:xpath1.0"><!
[CDATA[ns:str]]></bpel:query>
                    </bpel:to>
                  </bpel:copy>
                </bpel:assign>
                <bpel:invoke name="Agenda SS (annotations)" partnerLink="agendaSS"
operation="getLatestAnnotations" portType="ns:AgendaServices"
inputVariable="agendaSSRequest3" outputVariable="agendaSSResponse3"></bpel:invoke>
                <bpel:assign validate="no" name="Assign25">
                  <bpel:copy>
                    <bpel:from>
                      <bpel:literal xml:space="preserve"><impl:opAgendauserOperation
xmlns:impl="http://code" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <impl:str></impl:str>
</impl:opAgendauserOperation>
</bpel:literal>
                    </bpel:from>
                    <bpel:to variable="agendaSSUserInteractionRequest"
part="parameters"></bpel:to>
                  </bpel:copy>
                  <bpel:copy>
                    <bpel:from part="parameters" variable="agendaSSResponse3">
                      <bpel:query
queryLanguage="urn:oasis:names:tc:wsbpel:2.0:sublang:xpath1.0"><!
[CDATA[ns:getLatestAnnotationsReturn]]></bpel:query>
                    </bpel:from>
                    <bpel:to part="parameters" variable="agendaSSUserInteractionRequest">
                      <bpel:query
queryLanguage="urn:oasis:names:tc:wsbpel:2.0:sublang:xpath1.0"><!
[CDATA[ns:str]]></bpel:query>
                    </bpel:to>
                  </bpel:copy>
                </bpel:assign>
                <bpel:repeatUntil name="Agenda">
                  <bpel:sequence name="Sequence11">
                    <bpel:invoke name="Agenda Access"
partnerLink="agendaSSUserInteraction" operation="opAgendauserOperation"
portType="ns:GUIAgendaServices" inputVariable="agendaSSUserInteractionRequest"
outputVariable="agendaSSUserInteractionResponse"></bpel:invoke>
                    <bpel:assign validate="no" name="Assign26"><bpel:copy>
                    <bpel:from>
                      <bpel:literal xml:space="preserve"><impl:closeEverything
xmlns:impl="http://code" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <impl:str></impl:str>
</impl:closeEverything>
</bpel:literal>
```

```
                           </bpel:from>
                           <bpel:to variable="exitRequest" part="parameters"></bpel:to>
                        </bpel:copy>
                        <bpel:copy>
                           <bpel:from part="parameters" variable="agendaSSUserInteractionResponse">
                              <bpel:query
queryLanguage="urn:oasis:names:tc:wsbpel:2.0:sublang:xpath1.0"><!
[CDATA[ns:opAgendauserOperationReturn]]></bpel:query>
                           </bpel:from>
                           <bpel:to part="parameters" variable="exitRequest">
                              <bpel:query
queryLanguage="urn:oasis:names:tc:wsbpel:2.0:sublang:xpath1.0"><!
[CDATA[ns:str]]></bpel:query>
                           </bpel:to>
                        </bpel:copy>
                           </bpel:assign>
                        </bpel:sequence>
                        <bpel:condition><!
[CDATA[contains($agendaSSUserInteractionResponse.parameters/ns:opAgendauserOperationRetur
n, 'end-GUI-Agenda')]]></bpel:condition>
                     </bpel:repeatUntil>
                  </bpel:sequence>
                  <bpel:else>
                     <bpel:sequence name="Sequence12">
                        <bpel:flow name="Flow"><bpel:sequence name="Sequence13">
                              <bpel:assign validate="no" name="Assign27">
                                 <bpel:copy>
                                    <bpel:from>
                                       <bpel:literal xml:space="preserve"><impl:getLatestDates
xmlns:impl="http://code" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <impl:str></impl:str>
</impl:getLatestDates>
</bpel:literal>
                                    </bpel:from>
                                    <bpel:to variable="agendaSSRequest2" part="parameters"></bpel:to>
                                 </bpel:copy>
                                 <bpel:copy>
                                    <bpel:from part="parameters"
variable="basicUserOperationsResponse1">
                                       <bpel:query
queryLanguage="urn:oasis:names:tc:wsbpel:2.0:sublang:xpath1.0"><!
[CDATA[ns:opSelectNeeduserOperationReturn]]></bpel:query>
                                    </bpel:from>
                                    <bpel:to part="parameters" variable="agendaSSRequest2">
                                       <bpel:query
queryLanguage="urn:oasis:names:tc:wsbpel:2.0:sublang:xpath1.0"><!
[CDATA[ns:str]]></bpel:query>
                                    </bpel:to>
                                 </bpel:copy>
                              </bpel:assign>
                              <bpel:invoke name="Agenda SS (dates)" partnerLink="agendaSS"
operation="getLatestDates" portType="ns:AgendaServices" inputVariable="agendaSSRequest2"
outputVariable="agendaSSResponse2"></bpel:invoke>
```

```
                              <bpel:assign validate="no" name="Assign28">
                                 <bpel:copy>
                                    <bpel:from>
                                       <bpel:literal xml:space="preserve"><impl:opAgendauserOperation
xmlns:impl="http://code" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <impl:str></impl:str>
</impl:opAgendauserOperation>
</bpel:literal>
                                    </bpel:from>
                                    <bpel:to variable="agendaSSUserInteractionRequest"
part="parameters"></bpel:to>
                                 </bpel:copy>
                                 <bpel:copy>
                                    <bpel:from part="parameters" variable="agendaSSResponse2">
                                       <bpel:query
queryLanguage="urn:oasis:names:tc:wsbpel:2.0:sublang:xpath1.0"><!
[CDATA[ns:getLatestDatesReturn]]></bpel:query>
                                    </bpel:from>
                                    <bpel:to part="parameters"
variable="agendaSSUserInteractionRequest">
                                       <bpel:query
queryLanguage="urn:oasis:names:tc:wsbpel:2.0:sublang:xpath1.0"><!
[CDATA[ns:str]]></bpel:query>
                                    </bpel:to>
                                 </bpel:copy>
                              </bpel:assign>
                           </bpel:sequence><bpel:sequence name="Sequence158">
                              <bpel:assign validate="no" name="Assign30">
                                 <bpel:copy>
                                    <bpel:from>
                                       <bpel:literal xml:space="preserve"><impl:getLatestAnnotations
xmlns:impl="http://code" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <impl:str></impl:str>
</impl:getLatestAnnotations>
</bpel:literal>
                                    </bpel:from>
                                    <bpel:to variable="agendaSSRequest3" part="parameters"></bpel:to>
                                 </bpel:copy>
                                 <bpel:copy>
                                    <bpel:from part="parameters"
variable="basicUserOperationsResponse1">
                                       <bpel:query
queryLanguage="urn:oasis:names:tc:wsbpel:2.0:sublang:xpath1.0"><!
[CDATA[ns:opSelectNeeduserOperationReturn]]></bpel:query>
                                    </bpel:from>
                                    <bpel:to part="parameters" variable="agendaSSRequest3">
                                       <bpel:query
queryLanguage="urn:oasis:names:tc:wsbpel:2.0:sublang:xpath1.0"><!
[CDATA[ns:str]]></bpel:query>
                                    </bpel:to>
                                 </bpel:copy>
                              </bpel:assign>
                              <bpel:invoke name="Agenda SS (annotations)" partnerLink="agendaSS"
```

```
operation="getLatestAnnotations" portType="ns:AgendaServices"
inputVariable="agendaSSRequest3" outputVariable="agendaSSResponse3"></bpel:invoke>
                    <bpel:assign validate="no" name="Assign31">



                      <bpel:copy>
                        <bpel:from>
                          <bpel:literal xml:space="preserve"><impl:opAgendauserOperation
xmlns:impl="http://code" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <impl:str></impl:str>
</impl:opAgendauserOperation>
</bpel:literal>
                        </bpel:from>
                        <bpel:to variable="agendaSSUserInteractionRequest"
part="parameters"></bpel:to>
                      </bpel:copy>
                      <bpel:copy>
                        <bpel:from part="parameters" variable="agendaSSResponse3">
                          <bpel:query
queryLanguage="urn:oasis:names:tc:wsbpel:2.0:sublang:xpath1.0"><!
[CDATA[ns:getLatestAnnotationsReturn]]></bpel:query>
                        </bpel:from>
                        <bpel:to part="parameters"
variable="agendaSSUserInteractionRequest">
                          <bpel:query
queryLanguage="urn:oasis:names:tc:wsbpel:2.0:sublang:xpath1.0"><!
[CDATA[ns:str]]></bpel:query>
                        </bpel:to>
                      </bpel:copy>
                    </bpel:assign>
                  </bpel:sequence></bpel:flow>
                <bpel:repeatUntil name="Agenda">
                  <bpel:sequence name="Sequence14">
                    <bpel:invoke name="Agenda Access"
partnerLink="agendaSSUserInteraction" operation="opAgendauserOperation"
portType="ns:GUIAgendaServices" inputVariable="agendaSSUserInteractionRequest"
outputVariable="agendaSSUserInteractionResponse"></bpel:invoke>
                    <bpel:assign validate="no" name="Assign29"><bpel:copy>
                    <bpel:from>
                      <bpel:literal xml:space="preserve"><impl:closeEverything
xmlns:impl="http://code" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <impl:str></impl:str>
</impl:closeEverything>
</bpel:literal>
                    </bpel:from>
                    <bpel:to variable="exitRequest" part="parameters"></bpel:to>
                  </bpel:copy>
                  <bpel:copy>
                    <bpel:from part="parameters"
variable="agendaSSUserInteractionResponse">
                      <bpel:query
queryLanguage="urn:oasis:names:tc:wsbpel:2.0:sublang:xpath1.0"><!
```

```
[CDATA[ns:opAgendauserOperationReturn]]></bpel:query>
                    </bpel:from>
                    <bpel:to part="parameters" variable="exitRequest">
                      <bpel:query
queryLanguage="urn:oasis:names:tc:wsbpel:2.0:sublang:xpath1.0"><!
[CDATA[ns:str]]></bpel:query>
                    </bpel:to>
                  </bpel:copy>

                    </bpel:assign>
                  </bpel:sequence>
                  <bpel:condition><!
[CDATA[contains($agendaSSUserInteractionResponse.parameters/ns:opAgendauserOperationRetur
n, 'end-GUI-Agenda')]]></bpel:condition>
                </bpel:repeatUntil>
              </bpel:sequence>
            </bpel:else>
          </bpel:if>
        </bpel:sequence>
      </bpel:elseif><bpel:elseif>
        <bpel:condition><!
[CDATA[contains($featureSelectionWOResponse.parameters/ns:opSelectFeatureuserOperationRetur
n, '@email&agenda@')]]></bpel:condition>
        <bpel:sequence name="Sequence15">
          <bpel:assign validate="no" name="Assign32">
            <bpel:copy>
              <bpel:from>
                <bpel:literal xml:space="preserve"><impl:mailReviewer
xmlns:impl="http://code" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <impl:str></impl:str>
</impl:mailReviewer>
</bpel:literal>
              </bpel:from>
              <bpel:to variable="mailSSRequest" part="parameters"></bpel:to>
            </bpel:copy>
            <bpel:copy>
              <bpel:from part="parameters" variable="featureSelectionWOResponse">
                <bpel:query
queryLanguage="urn:oasis:names:tc:wsbpel:2.0:sublang:xpath1.0"><!
[CDATA[ns:opSelectFeatureuserOperationReturn]]></bpel:query>
              </bpel:from>
              <bpel:to part="parameters" variable="mailSSRequest">
                <bpel:query
queryLanguage="urn:oasis:names:tc:wsbpel:2.0:sublang:xpath1.0"><!
[CDATA[ns:str]]></bpel:query>
              </bpel:to>
            </bpel:copy>
          </bpel:assign>
          <bpel:invoke name="Mail SS (review)" partnerLink="mailSS"
operation="mailReviewer" portType="ns:MailServices" inputVariable="mailSSRequest"
outputVariable="mailSSResponse"></bpel:invoke>
          <bpel:assign validate="no" name="Assign33">
            <bpel:copy>
```

```
              <bpel:from>
                <bpel:literal xml:space="preserve"><impl:mailComposer
xmlns:impl="http://code" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <impl:str></impl:str>
</impl:mailComposer>
</bpel:literal>
              </bpel:from>
              <bpel:to variable="mailSSRequest1" part="parameters"></bpel:to>
            </bpel:copy>
            <bpel:copy>
              <bpel:from part="parameters" variable="mailSSResponse">
                <bpel:query
queryLanguage="urn:oasis:names:tc:wsbpel:2.0:sublang:xpath1.0"><!
[CDATA[ns:mailReviewerReturn]]></bpel:query>
              </bpel:from>
              <bpel:to part="parameters" variable="mailSSRequest1">
                <bpel:query
queryLanguage="urn:oasis:names:tc:wsbpel:2.0:sublang:xpath1.0"><!
[CDATA[ns:str]]></bpel:query>
              </bpel:to>
            </bpel:copy>
          </bpel:assign>
          <bpel:invoke name="Mail SS (composer)" partnerLink="mailSS"
operation="mailComposer" portType="ns:MailServices" inputVariable="mailSSRequest1"
outputVariable="mailSSResponse1"></bpel:invoke>
          <bpel:assign validate="no" name="Assign34">
            <bpel:copy>
              <bpel:from>
                <bpel:literal xml:space="preserve"><impl:accountManager
xmlns:impl="http://code" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <impl:str></impl:str>
</impl:accountManager>
</bpel:literal>
              </bpel:from>
              <bpel:to variable="mailSSRequest2" part="parameters"></bpel:to>
            </bpel:copy>
            <bpel:copy>
              <bpel:from part="parameters" variable="mailSSResponse1">
                <bpel:query
queryLanguage="urn:oasis:names:tc:wsbpel:2.0:sublang:xpath1.0"><!
[CDATA[ns:mailComposerReturn]]></bpel:query>
              </bpel:from>
              <bpel:to part="parameters" variable="mailSSRequest2">
                <bpel:query
queryLanguage="urn:oasis:names:tc:wsbpel:2.0:sublang:xpath1.0"><!
[CDATA[ns:str]]></bpel:query>
              </bpel:to>
            </bpel:copy>
          </bpel:assign>
          <bpel:invoke name="Mail SS (management)" partnerLink="mailSS"
operation="accountManager" portType="ns:MailServices" inputVariable="mailSSRequest2"
outputVariable="mailSSResponse2"></bpel:invoke>
          <bpel:assign validate="no" name="Assign35">
```

```
                  <bpel:copy>
                    <bpel:from>
                      <bpel:literal xml:space="preserve"><impl:initiateCalendar
xmlns:impl="http://code" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <impl:str></impl:str>
</impl:initiateCalendar>
</bpel:literal>
                    </bpel:from>
                    <bpel:to variable="agendaSSRequest" part="parameters"></bpel:to>
                  </bpel:copy>
                  <bpel:copy>
                    <bpel:from part="parameters" variable="mailSSResponse2">
                      <bpel:query
queryLanguage="urn:oasis:names:tc:wsbpel:2.0:sublang:xpath1.0"><!
[CDATA[ns:accountManagerReturn]]></bpel:query>
                    </bpel:from>
                    <bpel:to part="parameters" variable="agendaSSRequest">
                      <bpel:query
queryLanguage="urn:oasis:names:tc:wsbpel:2.0:sublang:xpath1.0"><!
[CDATA[ns:str]]></bpel:query>
                    </bpel:to>
                  </bpel:copy>
                </bpel:assign>
                <bpel:invoke name="Agenda SS (calendar)" partnerLink="agendaSS"
operation="initiateCalendar" portType="ns:AgendaServices" inputVariable="agendaSSRequest"
outputVariable="agendaSSResponse"></bpel:invoke>
                <bpel:assign validate="no" name="Assign36">
                  <bpel:copy>
                    <bpel:from>
                      <bpel:literal xml:space="preserve"><impl:initiateScheduler
xmlns:impl="http://code" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <impl:str></impl:str>
</impl:initiateScheduler>
</bpel:literal>
                    </bpel:from>
                    <bpel:to variable="agendaSSRequest1" part="parameters"></bpel:to>
                  </bpel:copy>
                  <bpel:copy>
                    <bpel:from part="parameters" variable="agendaSSResponse">
                      <bpel:query
queryLanguage="urn:oasis:names:tc:wsbpel:2.0:sublang:xpath1.0"><!
[CDATA[ns:initiateCalendarReturn]]></bpel:query>
                    </bpel:from>
                    <bpel:to part="parameters" variable="agendaSSRequest1">
                      <bpel:query
queryLanguage="urn:oasis:names:tc:wsbpel:2.0:sublang:xpath1.0"><!
[CDATA[ns:str]]></bpel:query>
                    </bpel:to>
                  </bpel:copy>

                </bpel:assign>
                <bpel:invoke name="Agenda SS (scheduler)" partnerLink="agendaSS"
operation="initiateScheduler" portType="ns:AgendaServices" inputVariable="agendaSSRequest1"
```

```
outputVariable="agendaSSResponse1"></bpel:invoke>
              <bpel:assign validate="no" name="Assign37">
                 <bpel:copy>
                    <bpel:from>
                       <bpel:literal xml:space="preserve"><impl:getLatestDates
xmlns:impl="http://code" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <impl:str></impl:str>
</impl:getLatestDates>
</bpel:literal>
                    </bpel:from>
                    <bpel:to variable="agendaSSRequest2" part="parameters"></bpel:to>
                 </bpel:copy>
                 <bpel:copy>
                    <bpel:from part="parameters" variable="agendaSSResponse1">
                       <bpel:query
queryLanguage="urn:oasis:names:tc:wsbpel:2.0:sublang:xpath1.0"><!
[CDATA[ns:initiateSchedulerReturn]]></bpel:query>
                    </bpel:from>
                    <bpel:to part="parameters" variable="agendaSSRequest2">
                       <bpel:query
queryLanguage="urn:oasis:names:tc:wsbpel:2.0:sublang:xpath1.0"><!
[CDATA[ns:str]]></bpel:query>
                    </bpel:to>
                 </bpel:copy>
              </bpel:assign>
              <bpel:invoke name="Agenda SS (dates)" partnerLink="agendaSS"
operation="getLatestDates" portType="ns:AgendaServices" inputVariable="agendaSSRequest2"
outputVariable="agendaSSResponse2"></bpel:invoke>
              <bpel:assign validate="no" name="Assign38">
                 <bpel:copy>
                    <bpel:from>
                       <bpel:literal xml:space="preserve"><impl:getLatestAnnotations
xmlns:impl="http://code" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <impl:str></impl:str>
</impl:getLatestAnnotations>
</bpel:literal>
                    </bpel:from>
                    <bpel:to variable="agendaSSRequest3" part="parameters"></bpel:to>
                 </bpel:copy>
                 <bpel:copy>
                    <bpel:from part="parameters" variable="agendaSSResponse2">
                       <bpel:query
queryLanguage="urn:oasis:names:tc:wsbpel:2.0:sublang:xpath1.0"><!
[CDATA[ns:getLatestDatesReturn]]></bpel:query>
                    </bpel:from>
                    <bpel:to part="parameters" variable="agendaSSRequest3">
                       <bpel:query
queryLanguage="urn:oasis:names:tc:wsbpel:2.0:sublang:xpath1.0"><!
[CDATA[ns:str]]></bpel:query>
                    </bpel:to>
                 </bpel:copy>
              </bpel:assign>
              <bpel:invoke name="Agenda SS (annotations)" partnerLink="agendaSS"
```

```
operation="getLatestAnnotations" portType="ns:AgendaServices"
inputVariable="agendaSSRequest3" outputVariable="agendaSSResponse3"></bpel:invoke>
               <bpel:assign validate="no" name="Assign39">
                  <bpel:copy>
                     <bpel:from>
                        <bpel:literal xml:space="preserve"><impl:initiateCrossReferencing
xmlns:impl="http://code" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <impl:str></impl:str>
</impl:initiateCrossReferencing>
</bpel:literal>
                     </bpel:from>
                     <bpel:to variable="extraSSRequest" part="parameters"></bpel:to>
                  </bpel:copy>
                  <bpel:copy>
                     <bpel:from part="parameters" variable="agendaSSResponse3">
                        <bpel:query
queryLanguage="urn:oasis:names:tc:wsbpel:2.0:sublang:xpath1.0"><!
[CDATA[ns:getLatestAnnotationsReturn]]></bpel:query>
                     </bpel:from>
                     <bpel:to part="parameters" variable="extraSSRequest">
                        <bpel:query
queryLanguage="urn:oasis:names:tc:wsbpel:2.0:sublang:xpath1.0"><!
[CDATA[ns:str]]></bpel:query>
                     </bpel:to>
                  </bpel:copy>
               </bpel:assign>
               <bpel:invoke name="Extra SS (cross-ref)" partnerLink="extraSS"
operation="initiateCrossReferencing" portType="ns:ExtraServices" inputVariable="extraSSRequest"
outputVariable="extraSSResponse"></bpel:invoke>
               <bpel:assign validate="no" name="Assign40">
                  <bpel:copy>
                     <bpel:from>
                        <bpel:literal xml:space="preserve"><impl:opMailandAgendauserOperation
xmlns:impl="http://code" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <impl:str></impl:str>
</impl:opMailandAgendauserOperation>
</bpel:literal>
                     </bpel:from>
                     <bpel:to variable="mailandagendaUserInteractionRequest"
part="parameters"></bpel:to>
                  </bpel:copy>
                  <bpel:copy>
                     <bpel:from part="parameters" variable="extraSSResponse">
                        <bpel:query
queryLanguage="urn:oasis:names:tc:wsbpel:2.0:sublang:xpath1.0"><!
[CDATA[ns:initiateCrossReferencingReturn]]></bpel:query>
                     </bpel:from>
                     <bpel:to part="parameters" variable="mailandagendaUserInteractionRequest">
                        <bpel:query
queryLanguage="urn:oasis:names:tc:wsbpel:2.0:sublang:xpath1.0"><!
[CDATA[ns:str]]></bpel:query>
                     </bpel:to>
                  </bpel:copy>
               </bpel:assign>
               <bpel:repeatUntil name="Mail &amp; Agenda">
                  <bpel:sequence name="Sequence16">
                     <bpel:invoke name="Mail and Agenda"
partnerLink="mailandagendaUserInteraction" operation="opMailandAgendauserOperation"
portType="ns:GUIMailandAgendaServices" inputVariable="mailandagendaUserInteractionRequest"
outputVariable="mailandagendaUserInteractionResponse"></bpel:invoke>

                     <bpel:assign validate="no" name="Assign41"><bpel:copy>
                     <bpel:from>
                        <bpel:literal xml:space="preserve"><impl:closeEverything
xmlns:impl="http://code" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <impl:str></impl:str>
</impl:closeEverything>
</bpel:literal>
                     </bpel:from>
                     <bpel:to variable="exitRequest" part="parameters"></bpel:to>
                  </bpel:copy>
                  <bpel:copy>
                     <bpel:from part="parameters"
variable="mailandagendaUserInteractionResponse">
                        <bpel:query
queryLanguage="urn:oasis:names:tc:wsbpel:2.0:sublang:xpath1.0"><!
[CDATA[ns:opMailandAgendauserOperationReturn]]></bpel:query>
                     </bpel:from>
                     <bpel:to part="parameters" variable="exitRequest">
                        <bpel:query
queryLanguage="urn:oasis:names:tc:wsbpel:2.0:sublang:xpath1.0"><!
[CDATA[ns:str]]></bpel:query>
                     </bpel:to>
                  </bpel:copy>

                     </bpel:assign>
                  </bpel:sequence>
                  <bpel:condition><!
[CDATA[contains($mailandagendaUserInteractionResponse.parameters/ns:opMailandAgendauserO
perationReturn, 'end-GUI-Mail-and-Agenda-Services')]]></bpel:condition>
               </bpel:repeatUntil>
            </bpel:sequence>
         </bpel:elseif></bpel:if>
      <bpel:invoke name="Exit" partnerLink="exit" operation="closeEverything"
portType="ns:Shutdown" inputVariable="exitRequest"
outputVariable="exitResponse"></bpel:invoke>
      <bpel:assign validate="no" name="Assign12">
         <bpel:copy>
            <bpel:from>
               <bpel:literal xml:space="preserve"><tns:WebOrganizerv01Response
xmlns:tns="http://weborganizerv01" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <tns:result></tns:result>
</tns:WebOrganizerv01Response>
</bpel:literal>
            </bpel:from>
            <bpel:to variable="output" part="payload"></bpel:to>
            </bpel:copy>
            <bpel:copy>
                <bpel:from part="parameters" variable="exitResponse">
                    <bpel:query queryLanguage="urn:oasis:names:tc:wsbpel:2.0:sublang:xpath1.0"><!
[CDATA[ns:closeEverythingReturn]]></bpel:query>
                </bpel:from>
                <bpel:to part="payload" variable="output">
                    <bpel:query queryLanguage="urn:oasis:names:tc:wsbpel:2.0:sublang:xpath1.0"><!
[CDATA[tns:result]]></bpel:query>
                </bpel:to>
            </bpel:copy>
        </bpel:assign>
        <bpel:reply name="replyOutput"
               partnerLink="client"
               portType="tns:WebOrganizerv01"
               operation="process"
               variable="output"
               />
    </bpel:sequence>
</bpel:process>
```

Atc-SBS Scenario Specification

ATC SBS

```
<!-- ATC BPEL Process [Generated by the Eclipse BPEL Designer] -->
<bpel:process name="ATC"
         targetNamespace="atc"
         suppressJoinFailure="yes"
         xmlns:tns="atc"
         xmlns:bpel="http://docs.oasis-open.org/wsbpel/2.0/process/executable"
         xmlns:ns="http://code" xmlns:ns0="http://www.example.org/RadarService/">

    <!-- Import the client WSDL -->
    <bpel:import namespace="http://code" location="GuiAccessService.wsdl"
importType="http://schemas.xmlsoap.org/wsdl/"></bpel:import>
    <bpel:import namespace="http://www.example.org/RadarService/" location="RadarService.wsdl"
importType="http://schemas.xmlsoap.org/wsdl/"></bpel:import>
    <bpel:import namespace="http://code" location="ResourcesService.wsdl"
importType="http://schemas.xmlsoap.org/wsdl/"></bpel:import>
    <bpel:import namespace="http://code" location="SchedulerService.wsdl"
importType="http://schemas.xmlsoap.org/wsdl/"></bpel:import>
    <bpel:import namespace="http://code" location="StatusService.wsdl"
importType="http://schemas.xmlsoap.org/wsdl/"></bpel:import>
    <bpel:import location="ATCArtifacts.wsdl" namespace="atc"
                importType="http://schemas.xmlsoap.org/wsdl/" />

    <!-- ================================================================
-->
    <!-- PARTNERLINKS                                                    -->
    <!-- List of services participating in this BPEL process             -->
    <!-- ================================================================
-->
    <bpel:partnerLinks>
        <!-- The 'client' role represents the requester of this service. -->
        <bpel:partnerLink name="client"
                partnerLinkType="tns:ATC"
                myRole="ATCProvider"
                />
        <bpel:partnerLink name="StatusPL" partnerLinkType="tns:StatusPL"
partnerRole="StatusProvider"></bpel:partnerLink>
        <bpel:partnerLink name="SchedulerPL" partnerLinkType="tns:SchedulerPL"
partnerRole="SchedulerProvider"></bpel:partnerLink>
        <bpel:partnerLink name="ResourcesPL" partnerLinkType="tns:ResourcesPL"
partnerRole="ResourcesProvider"></bpel:partnerLink>
        <bpel:partnerLink name="RadarPL" partnerLinkType="tns:RadarPL"
partnerRole="RadarProvider"></bpel:partnerLink>
        <bpel:partnerLink name="GuiAccessPL" partnerLinkType="tns:GuiAccessPL"
partnerRole="GuiAccessProvider"></bpel:partnerLink>
    </bpel:partnerLinks>

    <!-- ================================================================
-->
    <!-- VARIABLES                                                       -->
    <!-- List of messages and XML documents used within this BPEL process  -->
    <!-- ================================================================
-->
   <bpel:variables>
      <!-- Reference to the message passed as input during initiation -->
      <bpel:variable name="input"
             messageType="tns:ATCRequestMessage"/>

      <!--
        Reference to the message that will be returned to the requester
        -->
      <bpel:variable name="output"
             messageType="tns:ATCResponseMessage"/>
      <bpel:variable name="StatusPLResponse"
messageType="ns:hangarsStatResponse"></bpel:variable>
      <bpel:variable name="StatusPLRequest"
messageType="ns:hangarsStatRequest"></bpel:variable>
      <bpel:variable name="StatusPLResponse1"
messageType="ns:controlTowersStatResponse"></bpel:variable>
      <bpel:variable name="StatusPLRequest1"
messageType="ns:controlTowersStatRequest"></bpel:variable>
      <bpel:variable name="GuiAccessPLResponse"
messageType="ns:generalLayoutResponse"></bpel:variable>
      <bpel:variable name="GuiAccessPLRequest"
messageType="ns:generalLayoutRequest"></bpel:variable>
      <bpel:variable name="GuiAccessPLResponse1"
messageType="ns:oploginuserOperationResponse"></bpel:variable>
      <bpel:variable name="GuiAccessPLRequest1"
messageType="ns:oploginuserOperationRequest"></bpel:variable>
      <bpel:variable name="GuiAccessPLResponse2"
messageType="ns:opselectionuserOperationResponse"></bpel:variable>
      <bpel:variable name="GuiAccessPLRequest2"
messageType="ns:opselectionuserOperationRequest"></bpel:variable>
      <bpel:variable name="RadarPLResponse"
messageType="ns0:opinDestinyuserOperationResponse"></bpel:variable>
      <bpel:variable name="RadarPLRequest"
messageType="ns0:opinDestinyuserOperationRequest"></bpel:variable>
      <bpel:variable name="RadarPLResponse1"
messageType="ns0:opinDepartureuserOperationResponse"></bpel:variable>
      <bpel:variable name="RadarPLRequest1"
messageType="ns0:opinDepartureuserOperationRequest"></bpel:variable>
      <bpel:variable name="GuiAccessPLResponse3"
messageType="ns:showResultsResponse"></bpel:variable>
      <bpel:variable name="GuiAccessPLRequest3"
messageType="ns:showResultsRequest"></bpel:variable>
      <bpel:variable name="RadarPLResponse2"
messageType="ns0:getPositioningResponse"></bpel:variable>
      <bpel:variable name="RadarPLRequest2"
messageType="ns0:getPositioningRequest"></bpel:variable>
      <bpel:variable name="RadarPLResponse3"
messageType="ns0:opreqCoordinatesuserOperationResponse"></bpel:variable>
      <bpel:variable name="RadarPLRequest3"
messageType="ns0:opreqCoordinatesuserOperationRequest"></bpel:variable>
      <bpel:variable name="RadarPLResponse4"
messageType="ns0:opinCoordinatesuserOperationResponse"></bpel:variable>
```

```
      <bpel:variable name="RadarPLRequest4"
messageType="ns0:opinCoordinatesuserOperationRequest"></bpel:variable>
      <bpel:variable name="RadarPLResponse5"
messageType="ns0:opreqTrajectoryuserOperationResponse"></bpel:variable>
      <bpel:variable name="RadarPLRequest5"
messageType="ns0:opreqTrajectoryuserOperationRequest"></bpel:variable>
      <bpel:variable name="GuiAccessPLResponse4"
messageType="ns:exitResponse"></bpel:variable>
      <bpel:variable name="GuiAccessPLRequest4"
messageType="ns:exitRequest"></bpel:variable>
      <bpel:variable name="GuiAccessPLResponse5"
messageType="ns:oprequestSpaceuserOperationResponse"></bpel:variable>
      <bpel:variable name="GuiAccessPLRequest5"
messageType="ns:oprequestSpaceuserOperationRequest"></bpel:variable>
      <bpel:variable name="StatusPLResponse2"
messageType="ns:terminalsStatResponse"></bpel:variable>
      <bpel:variable name="StatusPLRequest2"
messageType="ns:terminalsStatRequest"></bpel:variable>
      <bpel:variable name="StatusPLResponse3"
messageType="ns:takeoffsStatResponse"></bpel:variable>
      <bpel:variable name="StatusPLRequest3"
messageType="ns:takeoffsStatRequest"></bpel:variable>
      <bpel:variable name="StatusPLResponse4"
messageType="ns:landingsStatResponse"></bpel:variable>
      <bpel:variable name="StatusPLRequest4"
messageType="ns:landingsStatRequest"></bpel:variable>
      <bpel:variable name="StatusPLResponse5"
messageType="ns:runwaysAnnotationsResponse"></bpel:variable>
      <bpel:variable name="StatusPLRequest5"
messageType="ns:runwaysAnnotationsRequest"></bpel:variable>
      <bpel:variable name="ResourcesPLResponse"
messageType="ns:availableResourcesResponse"></bpel:variable>
      <bpel:variable name="ResourcesPLRequest"
messageType="ns:availableResourcesRequest"></bpel:variable>
      <bpel:variable name="ResourcesPLResponse1"
messageType="ns:opdistributionAvailableResourcesuserOperationResponse"></bpel:variable>
      <bpel:variable name="ResourcesPLRequest1"
messageType="ns:opdistributionAvailableResourcesuserOperationRequest"></bpel:variable>
      <bpel:variable name="SchedulerPLResponse"
messageType="ns:opnormalTaskOrganiseruserOperationResponse"></bpel:variable>
      <bpel:variable name="SchedulerPLRequest"
messageType="ns:opnormalTaskOrganiseruserOperationRequest"></bpel:variable>
      <bpel:variable name="ResourcesPLResponse2"
messageType="ns:updateResourcesResponse"></bpel:variable>
      <bpel:variable name="ResourcesPLRequest2"
messageType="ns:updateResourcesRequest"></bpel:variable>
      <bpel:variable name="SchedulerPLResponse1"
messageType="ns:opemergencyTaskOrganiseruserOperationResponse"></bpel:variable>
      <bpel:variable name="SchedulerPLRequest1"
messageType="ns:opemergencyTaskOrganiseruserOperationRequest"></bpel:variable>
   </bpel:variables>
```

```
    <!-- ================================================================
-->
    <!-- ORCHESTRATION LOGIC                                    -->
    <!-- Set of activities coordinating the flow of messages across the    -->
    <!-- services integrated within this business process               -->
    <!-- ================================================================
-->
    <bpel:sequence name="main">

        <!-- Receive input from requester.
             Note: This maps to operation defined in ATC.wsdl
             -->
        <bpel:receive name="receiveInput" partnerLink="client"
                 portType="tns:ATC"
                 operation="process" variable="input"
                 createInstance="yes"/>

        <!-- Generate reply to synchronous request -->
        <bpel:assign validate="no" name="Assign">
            <bpel:copy>
                <bpel:from>
                    <bpel:literal xml:space="preserve"><impl:generalLayout xmlns:impl="http://code"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <impl:str></impl:str>
</impl:generalLayout>
</bpel:literal>
                </bpel:from>
                <bpel:to variable="GuiAccessPLRequest" part="parameters"></bpel:to>
            </bpel:copy>
            <bpel:copy>
                <bpel:from part="payload" variable="input">
                    <bpel:query queryLanguage="urn:oasis:names:tc:wsbpel:2.0:sublang:xpath1.0"><!
[CDATA[tns:input]]></bpel:query>
                </bpel:from>
                <bpel:to part="parameters" variable="GuiAccessPLRequest">
                    <bpel:query queryLanguage="urn:oasis:names:tc:wsbpel:2.0:sublang:xpath1.0"><!
[CDATA[ns:str]]></bpel:query>
                </bpel:to>
            </bpel:copy>
        </bpel:assign>
        <bpel:invoke name="Layout" partnerLink="GuiAccessPL" operation="generalLayout"
portType="ns:GuiAccess" inputVariable="GuiAccessPLRequest"
outputVariable="GuiAccessPLResponse"></bpel:invoke>
        <bpel:assign validate="no" name="Assign1">
            <bpel:copy>
                <bpel:from>
                    <bpel:literal xml:space="preserve"><impl:oploginuserOperation
xmlns:impl="http://code" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <impl:str></impl:str>
</impl:oploginuserOperation>
</bpel:literal>
                </bpel:from>
                <bpel:to variable="GuiAccessPLRequest1" part="parameters"></bpel:to>
```

```
      </bpel:copy>
      <bpel:copy>
        <bpel:from part="parameters" variable="GuiAccessPLResponse">
          <bpel:query queryLanguage="urn:oasis:names:tc:wsbpel:2.0:sublang:xpath1.0"><!
[CDATA[ns:getLatestAnnotationsReturn]]></bpel:query>
        </bpel:from>
        <bpel:to part="parameters" variable="GuiAccessPLRequest1">
          <bpel:query queryLanguage="urn:oasis:names:tc:wsbpel:2.0:sublang:xpath1.0"><!
[CDATA[ns:str]]></bpel:query>
        </bpel:to>
      </bpel:copy>
    </bpel:assign>
    <bpel:invoke name="Access (UI)" partnerLink="GuiAccessPL"
operation="oploginuserOperation" portType="ns:GuiAccess"
inputVariable="GuiAccessPLRequest1" outputVariable="GuiAccessPLResponse1"></bpel:invoke>
    <bpel:if name="Role">
      <bpel:condition><!
[CDATA[contains($GuiAccessPLResponse.parameters/ns:getLatestAnnotationsReturn,
'pilot')]]></bpel:condition>
      <bpel:sequence name="Pilot">
        <bpel:assign validate="no" name="Assign2">

          <bpel:copy>
            <bpel:from>
              <bpel:literal xml:space="preserve"><impl:opselectionuserOperation
xmlns:impl="http://code" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <impl:str></impl:str>
</impl:opselectionuserOperation>
</bpel:literal>
            </bpel:from>
            <bpel:to variable="GuiAccessPLRequest2" part="parameters"></bpel:to>
          </bpel:copy>
          <bpel:copy>
            <bpel:from part="parameters" variable="GuiAccessPLResponse1">
              <bpel:query queryLanguage="urn:oasis:names:tc:wsbpel:2.0:sublang:xpath1.0"><!
[CDATA[ns:initiateSchedulerReturn]]></bpel:query>
            </bpel:from>
            <bpel:to part="parameters" variable="GuiAccessPLRequest2">
              <bpel:query queryLanguage="urn:oasis:names:tc:wsbpel:2.0:sublang:xpath1.0"><!
[CDATA[ns:str]]></bpel:query>
            </bpel:to>
          </bpel:copy>
        </bpel:assign>
        <bpel:invoke name="Selection (UI)" partnerLink="GuiAccessPL"
operation="opselectionuserOperation" portType="ns:GuiAccess"
inputVariable="GuiAccessPLRequest2" outputVariable="GuiAccessPLResponse2"></bpel:invoke>
        <bpel:if name="If">
          <bpel:sequence>
            <bpel:assign validate="no" name="Assign3">

              <bpel:copy>
                <bpel:from>
                  <bpel:literal xml:space="preserve"><tns:opinDepartureuserOperation
```

```
xmlns:tns="http://www.example.org/RadarService/"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <in></in>
</tns:opinDepartureuserOperation>
</bpel:literal>
                            </bpel:from>
                            <bpel:to variable="RadarPLRequest1" part="parameters"></bpel:to>
                        </bpel:copy>
                        <bpel:copy>
                            <bpel:from part="parameters" variable="GuiAccessPLResponse2">
                                <bpel:query
queryLanguage="urn:oasis:names:tc:wsbpel:2.0:sublang:xpath1.0"><!
[CDATA[ns:str]]></bpel:query>
                            </bpel:from>
                            <bpel:to part="parameters" variable="RadarPLRequest1">
                                <bpel:query
queryLanguage="urn:oasis:names:tc:wsbpel:2.0:sublang:xpath1.0"><![CDATA[in]]></bpel:query>
                            </bpel:to>
                        </bpel:copy>
                    </bpel:assign>
                    <bpel:invoke name="Destiny (UI)" partnerLink="RadarPL"
operation="opinDestinyuserOperation" portType="ns0:RadarService"
inputVariable="RadarPLRequest" outputVariable="RadarPLResponse"></bpel:invoke>
                    <bpel:assign validate="no" name="Assign4">
                        <bpel:copy>
                            <bpel:from>
                                <bpel:literal xml:space="preserve"><tns:opinDepartureuserOperation
xmlns:tns="http://www.example.org/RadarService/"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <in></in>
</tns:opinDepartureuserOperation>
</bpel:literal>
                            </bpel:from>
                            <bpel:to variable="RadarPLRequest1" part="parameters"></bpel:to>
                        </bpel:copy>
                        <bpel:copy>
                            <bpel:from part="parameters" variable="RadarPLResponse">
                                <bpel:query
queryLanguage="urn:oasis:names:tc:wsbpel:2.0:sublang:xpath1.0"><![CDATA[out]]></bpel:query>
                            </bpel:from>
                            <bpel:to part="parameters" variable="RadarPLRequest1">
                                <bpel:query
queryLanguage="urn:oasis:names:tc:wsbpel:2.0:sublang:xpath1.0"><![CDATA[in]]></bpel:query>
                            </bpel:to>
                        </bpel:copy>
                    </bpel:assign>
                    <bpel:invoke name="Departure (UI)" partnerLink="RadarPL"
operation="opinDepartureuserOperation" portType="ns0:RadarService"
inputVariable="RadarPLRequest1" outputVariable="RadarPLResponse1"></bpel:invoke>
                    <bpel:assign validate="no" name="Assign5">
                        <bpel:copy>
                            <bpel:from>
                                <bpel:literal xml:space="preserve"><impl:showResults
```

```
xmlns:impl="http://code" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <impl:str></impl:str>
</impl:showResults>
</bpel:literal>
                </bpel:from>
                <bpel:to variable="GuiAccessPLRequest3" part="parameters"></bpel:to>
              </bpel:copy>
              <bpel:copy>
                <bpel:from part="parameters" variable="RadarPLResponse1">
                  <bpel:query
queryLanguage="urn:oasis:names:tc:wsbpel:2.0:sublang:xpath1.0"><![CDATA[out]]></bpel:query>
                </bpel:from>
                <bpel:to part="parameters" variable="GuiAccessPLRequest3">
                  <bpel:query
queryLanguage="urn:oasis:names:tc:wsbpel:2.0:sublang:xpath1.0"><!
[CDATA[ns:str]]></bpel:query>
                </bpel:to>
              </bpel:copy>
            </bpel:assign>
            <bpel:invoke name="Results" partnerLink="GuiAccessPL" operation="showResults"
portType="ns:GuiAccess" inputVariable="GuiAccessPLRequest3"
outputVariable="GuiAccessPLResponse3"></bpel:invoke>
            <bpel:assign validate="no" name="Assign6">
              <bpel:copy>
                <bpel:from>
                  <bpel:literal xml:space="preserve"><impl:exit xmlns:impl="http://code"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <impl:str></impl:str>
</impl:exit>
</bpel:literal>
                </bpel:from>
                <bpel:to variable="GuiAccessPLRequest4" part="parameters"></bpel:to>
              </bpel:copy>
              <bpel:copy>
                <bpel:from part="parameters" variable="GuiAccessPLResponse3">
                  <bpel:query
queryLanguage="urn:oasis:names:tc:wsbpel:2.0:sublang:xpath1.0"><!
[CDATA[ns:stopSchedulerReturn]]></bpel:query>
                </bpel:from>
                <bpel:to part="parameters" variable="GuiAccessPLRequest4">
                  <bpel:query
queryLanguage="urn:oasis:names:tc:wsbpel:2.0:sublang:xpath1.0"><!
[CDATA[ns:str]]></bpel:query>
                </bpel:to>
              </bpel:copy>
            </bpel:assign>
          </bpel:sequence>
        <bpel:elseif>
            <bpel:sequence>
              <bpel:assign validate="no" name="Assign7">
                <bpel:copy>
                  <bpel:from>
                    <bpel:literal xml:space="preserve"><tns:getPositioning
```

```
xmlns:tns="http://www.example.org/RadarService/"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <in></in>
</tns:getPositioning>

</bpel:literal>
                        </bpel:from>
                        <bpel:to variable="RadarPLRequest2" part="parameters"></bpel:to>
                      </bpel:copy>
                      <bpel:copy>
                        <bpel:from part="parameters" variable="GuiAccessPLResponse2">
                          <bpel:query
queryLanguage="urn:oasis:names:tc:wsbpel:2.0:sublang:xpath1.0"><!
[CDATA[ns:str]]></bpel:query>
                        </bpel:from>
                        <bpel:to part="parameters" variable="RadarPLRequest2">
                          <bpel:query
queryLanguage="urn:oasis:names:tc:wsbpel:2.0:sublang:xpath1.0"><![CDATA[in]]></bpel:query>
                        </bpel:to>
                      </bpel:copy>
                    </bpel:assign>
                    <bpel:invoke name="Positioning" partnerLink="RadarPL"
operation="getPositioning" portType="ns0:RadarService" inputVariable="RadarPLRequest2"
outputVariable="RadarPLResponse2"></bpel:invoke>
                    <bpel:assign validate="no" name="Assign8">
                      <bpel:copy>
                        <bpel:from>
                          <bpel:literal xml:space="preserve"><tns:opreqCoordinatesuserOperation
xmlns:tns="http://www.example.org/RadarService/"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <in></in>
</tns:opreqCoordinatesuserOperation>
</bpel:literal>
                        </bpel:from>
                        <bpel:to variable="RadarPLRequest3" part="parameters"></bpel:to>
                      </bpel:copy>
                      <bpel:copy>
                        <bpel:from part="parameters" variable="RadarPLResponse2">
                          <bpel:query
queryLanguage="urn:oasis:names:tc:wsbpel:2.0:sublang:xpath1.0"><![CDATA[out]]></bpel:query>
                        </bpel:from>
                        <bpel:to part="parameters" variable="RadarPLRequest3">
                          <bpel:query
queryLanguage="urn:oasis:names:tc:wsbpel:2.0:sublang:xpath1.0"><![CDATA[in]]></bpel:query>
                        </bpel:to>
                      </bpel:copy>
                    </bpel:assign>
                    <bpel:invoke name="Request Coordinates (UI)" partnerLink="RadarPL"
operation="opreqCoordinatesuserOperation" portType="ns0:RadarService"
inputVariable="RadarPLRequest3" outputVariable="RadarPLResponse3"></bpel:invoke>
                    <bpel:assign validate="no" name="Assign9">
                      <bpel:copy>
                        <bpel:from>
                    <bpel:literal xml:space="preserve"><tns:opinCoordinatesuserOperation
xmlns:tns="http://www.example.org/RadarService/"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <in></in>
</tns:opinCoordinatesuserOperation>
</bpel:literal>
                  </bpel:from>
                  <bpel:to variable="RadarPLRequest4" part="parameters"></bpel:to>
                </bpel:copy>
                <bpel:copy>
                  <bpel:from part="parameters" variable="RadarPLResponse3">
                    <bpel:query
queryLanguage="urn:oasis:names:tc:wsbpel:2.0:sublang:xpath1.0"><![CDATA[out]]></bpel:query>
                  </bpel:from>
                  <bpel:to part="parameters" variable="RadarPLRequest4">
                    <bpel:query
queryLanguage="urn:oasis:names:tc:wsbpel:2.0:sublang:xpath1.0"><![CDATA[in]]></bpel:query>
                  </bpel:to>
                </bpel:copy>
              </bpel:assign>
              <bpel:invoke name="Input Coordinates (UI)" partnerLink="RadarPL"
operation="opinCoordinatesuserOperation" portType="ns0:RadarService"
inputVariable="RadarPLRequest4" outputVariable="RadarPLResponse4"></bpel:invoke>
             <bpel:assign validate="no" name="Assign10">
                <bpel:copy>
                  <bpel:from>
                    <bpel:literal xml:space="preserve"><tns:opreqTrajectoryuserOperation
xmlns:tns="http://www.example.org/RadarService/"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <in></in>
</tns:opreqTrajectoryuserOperation>
</bpel:literal>
                  </bpel:from>
                  <bpel:to variable="RadarPLRequest5" part="parameters"></bpel:to>
                </bpel:copy>
                <bpel:copy>
                  <bpel:from part="parameters" variable="RadarPLResponse4">
                    <bpel:query
queryLanguage="urn:oasis:names:tc:wsbpel:2.0:sublang:xpath1.0"><![CDATA[out]]></bpel:query>
                  </bpel:from>
                  <bpel:to part="parameters" variable="RadarPLRequest5">
                    <bpel:query
queryLanguage="urn:oasis:names:tc:wsbpel:2.0:sublang:xpath1.0"><![CDATA[in]]></bpel:query>
                  </bpel:to>
                </bpel:copy>
              </bpel:assign>
              <bpel:invoke name="Request Trajectory (UI)" partnerLink="RadarPL"
operation="opreqTrajectoryuserOperation" inputVariable="RadarPLRequest5"
outputVariable="RadarPLResponse5"></bpel:invoke>
              <bpel:assign validate="no" name="Assign11">
                <bpel:copy>
                  <bpel:from>
                    <bpel:literal xml:space="preserve"><impl:showResults
```

```
xmlns:impl="http://code" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <impl:str></impl:str>
</impl:showResults>
</bpel:literal>
                  </bpel:from>
                  <bpel:to variable="GuiAccessPLRequest3" part="parameters"></bpel:to>
                </bpel:copy>
                <bpel:copy>
                  <bpel:from part="parameters" variable="RadarPLResponse5">
                    <bpel:query
queryLanguage="urn:oasis:names:tc:wsbpel:2.0:sublang:xpath1.0"><![CDATA[out]]></bpel:query>
                  </bpel:from>
                  <bpel:to part="parameters" variable="GuiAccessPLRequest3">
                    <bpel:query
queryLanguage="urn:oasis:names:tc:wsbpel:2.0:sublang:xpath1.0"><!
[CDATA[ns:str]]></bpel:query>
                  </bpel:to>
                </bpel:copy>
              </bpel:assign>
              <bpel:invoke name="Results" partnerLink="GuiAccessPL"
operation="showResults" portType="ns:GuiAccess" inputVariable="GuiAccessPLRequest3"
outputVariable="GuiAccessPLResponse3"></bpel:invoke>
              <bpel:assign validate="no" name="Assign12">
                <bpel:copy>
                  <bpel:from>
                    <bpel:literal xml:space="preserve"><impl:exit xmlns:impl="http://code"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <impl:str></impl:str>
</impl:exit>
</bpel:literal>
                  </bpel:from>
                  <bpel:to variable="GuiAccessPLRequest4" part="parameters"></bpel:to>
                </bpel:copy>
                <bpel:copy>
                  <bpel:from part="parameters" variable="GuiAccessPLResponse3">
                    <bpel:query
queryLanguage="urn:oasis:names:tc:wsbpel:2.0:sublang:xpath1.0"><!
[CDATA[ns:stopSchedulerReturn]]></bpel:query>
                  </bpel:from>
                  <bpel:to part="parameters" variable="GuiAccessPLRequest4">
                    <bpel:query
queryLanguage="urn:oasis:names:tc:wsbpel:2.0:sublang:xpath1.0"><!
[CDATA[ns:str]]></bpel:query>
                  </bpel:to>
                </bpel:copy>
              </bpel:assign>
            </bpel:sequence>
          </bpel:elseif><bpel:elseif>
            <bpel:sequence>
              <bpel:assign validate="no" name="Assign13">
                <bpel:copy>
                  <bpel:from>
                    <bpel:literal xml:space="preserve"><tns:getPositioning
```

```
xmlns:tns="http://www.example.org/RadarService/"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <in></in>
</tns:getPositioning>
</bpel:literal>
                                        </bpel:from>
                                        <bpel:to variable="RadarPLRequest2" part="parameters"></bpel:to>
                                    </bpel:copy>
                                    <bpel:copy>
                                        <bpel:from part="parameters" variable="GuiAccessPLResponse2">
                                            <bpel:query
queryLanguage="urn:oasis:names:tc:wsbpel:2.0:sublang:xpath1.0"><!
[CDATA[ns:str]]></bpel:query>
                                        </bpel:from>
                                        <bpel:to part="parameters" variable="RadarPLRequest2">
                                            <bpel:query
queryLanguage="urn:oasis:names:tc:wsbpel:2.0:sublang:xpath1.0"><![CDATA[in]]></bpel:query>
                                        </bpel:to>
                                    </bpel:copy>
                                </bpel:assign>
                                <bpel:invoke name="Positioning" partnerLink="RadarPL"
operation="getPositioning" portType="ns0:RadarService" inputVariable="RadarPLRequest2"
outputVariable="RadarPLResponse2"></bpel:invoke>
                                <bpel:assign validate="no" name="Assign14">
                                    <bpel:copy>
                                        <bpel:from>
                                            <bpel:literal xml:space="preserve"><impl:oprequestSpaceuserOperation
xmlns:impl="http://code" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <impl:str></impl:str>
</impl:oprequestSpaceuserOperation>
</bpel:literal>
                                        </bpel:from>
                                        <bpel:to variable="GuiAccessPLRequest5" part="parameters"></bpel:to>
                                    </bpel:copy>
                                    <bpel:copy>
                                        <bpel:from part="parameters" variable="RadarPLResponse2">
                                            <bpel:query
queryLanguage="urn:oasis:names:tc:wsbpel:2.0:sublang:xpath1.0"><![CDATA[out]]></bpel:query>
                                        </bpel:from>
                                        <bpel:to part="parameters" variable="GuiAccessPLRequest5">
                                            <bpel:query
queryLanguage="urn:oasis:names:tc:wsbpel:2.0:sublang:xpath1.0"><!
[CDATA[ns:str]]></bpel:query>
                                        </bpel:to>
                                    </bpel:copy>
                                </bpel:assign>
                                <bpel:invoke name="Request Space (UI)" partnerLink="GuiAccessPL"
operation="oprequestSpaceuserOperation" inputVariable="GuiAccessPLRequest5"
outputVariable="GuiAccessPLResponse5"></bpel:invoke>
                                <bpel:assign validate="no" name="Assign15">
                                    <bpel:copy>
                                        <bpel:from>
                                            <bpel:literal xml:space="preserve"><impl:showResults
```

```
xmlns:impl="http://code" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <impl:str></impl:str>
</impl:showResults>
</bpel:literal>
                                    </bpel:from>
                                    <bpel:to variable="GuiAccessPLRequest3" part="parameters"></bpel:to>
                                </bpel:copy>
                                <bpel:copy>
                                    <bpel:from part="parameters" variable="GuiAccessPLResponse5">
                                        <bpel:query
queryLanguage="urn:oasis:names:tc:wsbpel:2.0:sublang:xpath1.0"><!
[CDATA[ns:str]]></bpel:query>
                                    </bpel:from>
                                    <bpel:to part="parameters" variable="GuiAccessPLRequest3">
                                        <bpel:query
queryLanguage="urn:oasis:names:tc:wsbpel:2.0:sublang:xpath1.0"><!
[CDATA[ns:str]]></bpel:query>
                                    </bpel:to>
                                </bpel:copy>
                            </bpel:assign>
                            <bpel:invoke name="Results" partnerLink="GuiAccessPL"
operation="showResults" portType="ns:GuiAccess" inputVariable="GuiAccessPLRequest3"
outputVariable="GuiAccessPLResponse3"></bpel:invoke>
                            <bpel:assign validate="no" name="Assign16">
                                <bpel:copy>
                                    <bpel:from>
                                        <bpel:literal xml:space="preserve"><impl:exit xmlns:impl="http://code"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <impl:str></impl:str>
</impl:exit>
</bpel:literal>
                                    </bpel:from>
                                    <bpel:to variable="GuiAccessPLRequest4" part="parameters"></bpel:to>
                                </bpel:copy>
                                <bpel:copy>
                                    <bpel:from part="parameters" variable="GuiAccessPLResponse3">
                                        <bpel:query
queryLanguage="urn:oasis:names:tc:wsbpel:2.0:sublang:xpath1.0"><!
[CDATA[ns:stopSchedulerReturn]]></bpel:query>
                                    </bpel:from>
                                    <bpel:to part="parameters" variable="GuiAccessPLRequest4">
                                        <bpel:query
queryLanguage="urn:oasis:names:tc:wsbpel:2.0:sublang:xpath1.0"><!
[CDATA[ns:str]]></bpel:query>
                                    </bpel:to>
                                </bpel:copy>
                            </bpel:assign>
                        </bpel:sequence>
                    </bpel:elseif></bpel:if>
            </bpel:sequence>
        <bpel:elseif>
                <bpel:condition><!
[CDATA[contains($GuiAccessPLResponse.parameters/ns:getLatestAnnotationsReturn,
'maintener')]]></bpel:condition>
        <bpel:sequence name="Maintenance">
          <bpel:assign validate="no" name="Assign17">

            <bpel:copy>
              <bpel:from>
                <bpel:literal xml:space="preserve"><impl:hangarsStat xmlns:impl="http://code"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <impl:str></impl:str>
</impl:hangarsStat>
</bpel:literal>
              </bpel:from>
              <bpel:to variable="StatusPLRequest" part="parameters"></bpel:to>
            </bpel:copy>
            <bpel:copy>
              <bpel:from part="parameters" variable="GuiAccessPLResponse1">
                <bpel:query queryLanguage="urn:oasis:names:tc:wsbpel:2.0:sublang:xpath1.0">
                  <![CDATA[ns:initiateSchedulerReturn]]>
                </bpel:query>
              </bpel:from>
              <bpel:to part="parameters" variable="StatusPLRequest">
                <bpel:query
queryLanguage="urn:oasis:names:tc:wsbpel:2.0:sublang:xpath1.0"><!
[CDATA[ns:str]]></bpel:query>
              </bpel:to>
            </bpel:copy>
          </bpel:assign>
          <bpel:invoke name="Hangars" partnerLink="StatusPL" operation="hangarsStat"
portType="ns:Status" inputVariable="StatusPLRequest"
outputVariable="StatusPLResponse"></bpel:invoke>
          <bpel:assign validate="no" name="Assign18">
            <bpel:copy>
              <bpel:from>
                <bpel:literal xml:space="preserve"><impl:controlTowersStat
xmlns:impl="http://code" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <impl:str></impl:str>
</impl:controlTowersStat>
</bpel:literal>
              </bpel:from>
              <bpel:to variable="StatusPLRequest1" part="parameters"></bpel:to>
            </bpel:copy>
            <bpel:copy>
              <bpel:from part="parameters" variable="StatusPLResponse">
                <bpel:query
queryLanguage="urn:oasis:names:tc:wsbpel:2.0:sublang:xpath1.0"><!
[CDATA[ns:initiateCalendarReturn]]></bpel:query>
              </bpel:from>
              <bpel:to part="parameters" variable="StatusPLRequest1">
                <bpel:query
queryLanguage="urn:oasis:names:tc:wsbpel:2.0:sublang:xpath1.0"><!
[CDATA[ns:str]]></bpel:query>
              </bpel:to>
            </bpel:copy>
        </bpel:assign>
        <bpel:invoke name="Towers" partnerLink="StatusPL" operation="controlTowersStat"
portType="ns:Status" inputVariable="StatusPLRequest1"
outputVariable="StatusPLResponse1"></bpel:invoke>
        <bpel:assign validate="no" name="Assign20">
          <bpel:copy>
            <bpel:from>
              <bpel:literal xml:space="preserve"><impl:terminalsStat
xmlns:impl="http://code" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <impl:str></impl:str>
</impl:terminalsStat>
</bpel:literal>
            </bpel:from>
            <bpel:to variable="StatusPLRequest2" part="parameters"></bpel:to>
          </bpel:copy>
          <bpel:copy>
            <bpel:from part="parameters" variable="StatusPLResponse1">
              <bpel:query
queryLanguage="urn:oasis:names:tc:wsbpel:2.0:sublang:xpath1.0"><!
[CDATA[ns:stopCalendarReturn]]></bpel:query>
            </bpel:from>
            <bpel:to part="parameters" variable="StatusPLRequest2">
              <bpel:query
queryLanguage="urn:oasis:names:tc:wsbpel:2.0:sublang:xpath1.0"><!
[CDATA[ns:str]]></bpel:query>
            </bpel:to>
          </bpel:copy>
        </bpel:assign>
        <bpel:invoke name="Terminals" partnerLink="StatusPL" operation="terminalsStat"
portType="ns:Status" inputVariable="StatusPLRequest2"
outputVariable="StatusPLResponse2"></bpel:invoke>
        <bpel:assign validate="no" name="Assign21">
          <bpel:copy>
            <bpel:from>
              <bpel:literal xml:space="preserve"><impl:takeoffsStat xmlns:impl="http://code"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <impl:str></impl:str>
</impl:takeoffsStat>
</bpel:literal>
            </bpel:from>
            <bpel:to variable="StatusPLRequest3" part="parameters"></bpel:to>
          </bpel:copy>
          <bpel:copy>
            <bpel:from part="parameters" variable="StatusPLResponse2">
              <bpel:query
queryLanguage="urn:oasis:names:tc:wsbpel:2.0:sublang:xpath1.0"><!
[CDATA[ns:getLatestDatesReturn]]></bpel:query>
            </bpel:from>
            <bpel:to part="parameters" variable="StatusPLRequest3">
              <bpel:query
queryLanguage="urn:oasis:names:tc:wsbpel:2.0:sublang:xpath1.0"><!
[CDATA[ns:str]]></bpel:query>
            </bpel:to>
            </bpel:copy>
          </bpel:assign>
          <bpel:invoke name="Takeoffs" partnerLink="StatusPL" operation="takeoffsStat"
portType="ns:Status" inputVariable="StatusPLRequest3"
outputVariable="StatusPLResponse3"></bpel:invoke>
          <bpel:assign validate="no" name="Assign22">
            <bpel:copy>
              <bpel:from>
                <bpel:literal xml:space="preserve"><impl:landingsStat
xmlns:impl="http://code" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <impl:str></impl:str>
</impl:landingsStat>
</bpel:literal>
              </bpel:from>
              <bpel:to variable="StatusPLRequest4" part="parameters"></bpel:to>
            </bpel:copy>
            <bpel:copy>
              <bpel:from part="parameters" variable="StatusPLResponse3">
                <bpel:query
queryLanguage="urn:oasis:names:tc:wsbpel:2.0:sublang:xpath1.0"><!
[CDATA[ns:initiateSchedulerReturn]]></bpel:query>
              </bpel:from>
              <bpel:to part="parameters" variable="StatusPLRequest4">
                <bpel:query
queryLanguage="urn:oasis:names:tc:wsbpel:2.0:sublang:xpath1.0"><!
[CDATA[ns:str]]></bpel:query>
              </bpel:to>
            </bpel:copy>
          </bpel:assign>
          <bpel:invoke name="Landings" partnerLink="StatusPL" operation="landingsStat"
portType="ns:Status" inputVariable="StatusPLRequest4"
outputVariable="StatusPLResponse4"></bpel:invoke>
          <bpel:assign validate="no" name="Assign23">
            <bpel:copy>
              <bpel:from>
                <bpel:literal xml:space="preserve"><impl:runwaysAnnotations
xmlns:impl="http://code" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <impl:str></impl:str>
</impl:runwaysAnnotations>
</bpel:literal>
              </bpel:from>
              <bpel:to variable="StatusPLRequest5" part="parameters"></bpel:to>
            </bpel:copy>
            <bpel:copy>
              <bpel:from part="parameters" variable="StatusPLResponse4">
                <bpel:query
queryLanguage="urn:oasis:names:tc:wsbpel:2.0:sublang:xpath1.0"><!
[CDATA[ns:stopSchedulerReturn]]></bpel:query>
              </bpel:from>
              <bpel:to part="parameters" variable="StatusPLRequest5">
                <bpel:query
queryLanguage="urn:oasis:names:tc:wsbpel:2.0:sublang:xpath1.0"><!
[CDATA[ns:str]]></bpel:query>
```

```
                  </bpel:to>
               </bpel:copy>
            </bpel:assign>
            <bpel:invoke name="Runways" partnerLink="StatusPL"
operation="runwaysAnnotations" portType="ns:Status" inputVariable="StatusPLRequest5"
outputVariable="StatusPLResponse5"></bpel:invoke>
            <bpel:assign validate="no" name="Assign24">
               <bpel:copy>
                  <bpel:from>
                     <bpel:literal xml:space="preserve"><impl:availableResources
xmlns:impl="http://code" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <impl:str></impl:str>
</impl:availableResources>
</bpel:literal>
                  </bpel:from>
                  <bpel:to variable="ResourcesPLRequest" part="parameters"></bpel:to>
               </bpel:copy>
               <bpel:copy>
                  <bpel:from part="parameters" variable="StatusPLResponse5">
                     <bpel:query
queryLanguage="urn:oasis:names:tc:wsbpel:2.0:sublang:xpath1.0"><!
[CDATA[ns:getLatestAnnotationsReturn]]></bpel:query>
                  </bpel:from>
                  <bpel:to part="parameters" variable="ResourcesPLRequest">
                     <bpel:query
queryLanguage="urn:oasis:names:tc:wsbpel:2.0:sublang:xpath1.0"><!
[CDATA[ns:str]]></bpel:query>
                  </bpel:to>
               </bpel:copy>
            </bpel:assign>
            <bpel:invoke name="Available Resources" partnerLink="ResourcesPL"
operation="availableResources" portType="ns:Resources" inputVariable="ResourcesPLRequest"
outputVariable="ResourcesPLResponse"></bpel:invoke>
            <bpel:assign validate="no" name="Assign25">

               <bpel:copy>
                  <bpel:from>
                     <bpel:literal
xml:space="preserve"><impl:opdistributionAvailableResourcesuserOperation
xmlns:impl="http://code" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <impl:str></impl:str>
</impl:opdistributionAvailableResourcesuserOperation>
</bpel:literal>
                  </bpel:from>
                  <bpel:to variable="ResourcesPLRequest1" part="parameters"></bpel:to>
               </bpel:copy>
               <bpel:copy>
                  <bpel:from part="parameters" variable="ResourcesPLResponse">
                     <bpel:query
queryLanguage="urn:oasis:names:tc:wsbpel:2.0:sublang:xpath1.0"><!
[CDATA[ns:stopCalendarReturn]]></bpel:query>
                  </bpel:from>
                  <bpel:to part="parameters" variable="ResourcesPLRequest1">
```

```
                     <bpel:query
queryLanguage="urn:oasis:names:tc:wsbpel:2.0:sublang:xpath1.0"><!
[CDATA[ns:str]]></bpel:query>
                  </bpel:to>
               </bpel:copy>
            </bpel:assign>
            <bpel:invoke name="Distribution AR (UI)" partnerLink="ResourcesPL"
operation="opdistributionAvailableResourcesuserOperation" portType="ns:Resources"
inputVariable="ResourcesPLRequest1" outputVariable="ResourcesPLResponse1"></bpel:invoke>
            <bpel:assign validate="no" name="Assign26">
               <bpel:copy>
                  <bpel:from>
                     <bpel:literal xml:space="preserve"><impl:opnormalTaskOrganiseruserOperation
xmlns:impl="http://code" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <impl:str></impl:str>
</impl:opnormalTaskOrganiseruserOperation>
</bpel:literal>
                  </bpel:from>
                  <bpel:to variable="SchedulerPLRequest" part="parameters"></bpel:to>
               </bpel:copy>
               <bpel:copy>
                  <bpel:from part="parameters" variable="ResourcesPLResponse1">
                     <bpel:query
queryLanguage="urn:oasis:names:tc:wsbpel:2.0:sublang:xpath1.0"><!
[CDATA[ns:getLatestDatesReturn]]></bpel:query>
                  </bpel:from>
                  <bpel:to part="parameters" variable="SchedulerPLRequest">
                     <bpel:query
queryLanguage="urn:oasis:names:tc:wsbpel:2.0:sublang:xpath1.0"><!
[CDATA[ns:str]]></bpel:query>
                  </bpel:to>
               </bpel:copy>
            </bpel:assign>
            <bpel:invoke name="Scheduler (UI)" partnerLink="SchedulerPL"
operation="opnormalTaskOrganiseruserOperation" inputVariable="SchedulerPLRequest"
outputVariable="SchedulerPLResponse"></bpel:invoke>
            <bpel:assign validate="no" name="Assign27">
               <bpel:copy>
                  <bpel:from>
                     <bpel:literal xml:space="preserve"><impl:updateResources
xmlns:impl="http://code" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <impl:str></impl:str>
</impl:updateResources>
</bpel:literal>
                  </bpel:from>
                  <bpel:to variable="ResourcesPLRequest2" part="parameters"></bpel:to>
               </bpel:copy>
               <bpel:copy>
                  <bpel:from part="parameters" variable="SchedulerPLResponse">
                     <bpel:query
queryLanguage="urn:oasis:names:tc:wsbpel:2.0:sublang:xpath1.0"><!
[CDATA[ns:initiateCalendarReturn]]></bpel:query>
                  </bpel:from>
```

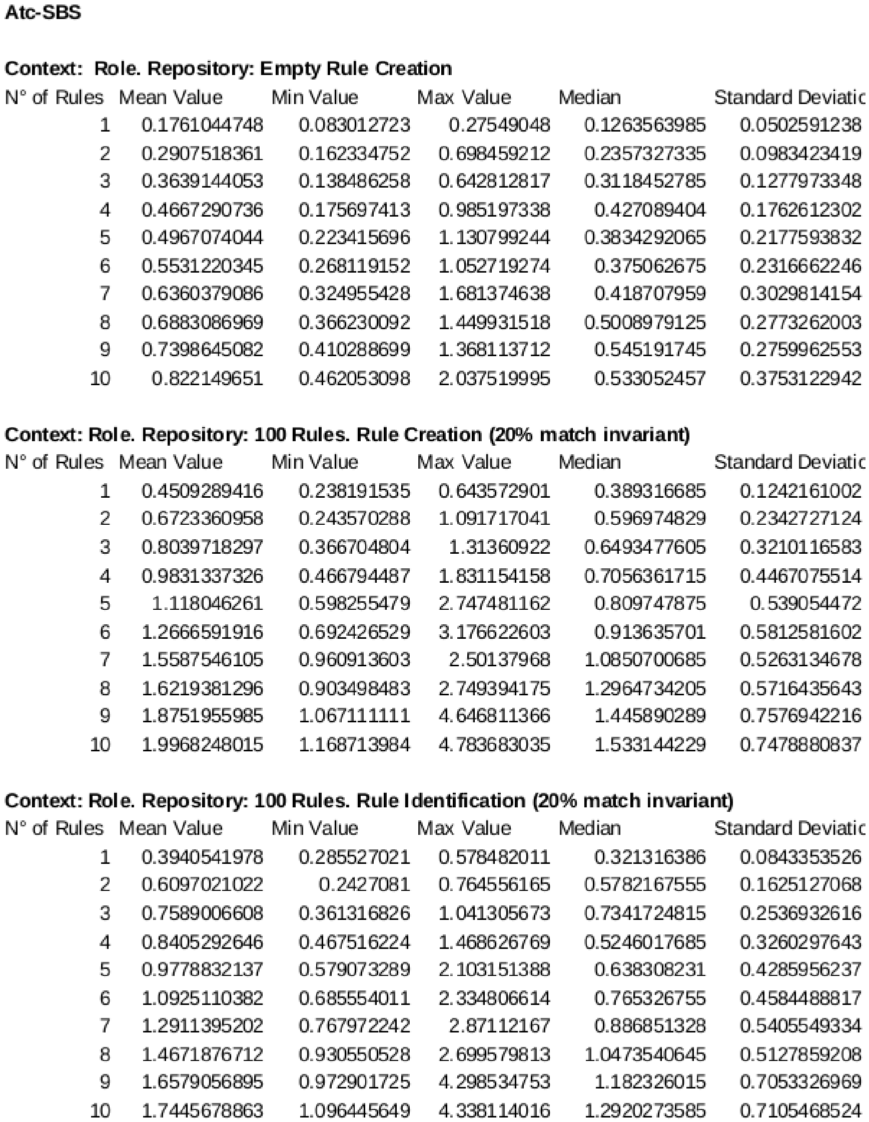
```
              <bpel:to part="parameters" variable="ResourcesPLRequest2">
                <bpel:query
queryLanguage="urn:oasis:names:tc:wsbpel:2.0:sublang:xpath1.0"><!
[CDATA[ns:str]]></bpel:query>
              </bpel:to>
            </bpel:copy>
          </bpel:assign>
          <bpel:invoke name="Update Resources" partnerLink="ResourcesPL"
operation="updateResources" portType="ns:Resources" inputVariable="ResourcesPLRequest2"
outputVariable="ResourcesPLResponse2"></bpel:invoke>
          <bpel:assign validate="no" name="Assign28">
            <bpel:copy>
              <bpel:from>
                <bpel:literal xml:space="preserve"><impl:exit xmlns:impl="http://code"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <impl:str></impl:str>
</impl:exit>
</bpel:literal>
              </bpel:from>
              <bpel:to variable="GuiAccessPLRequest4" part="parameters"></bpel:to>
            </bpel:copy>
            <bpel:copy>
              <bpel:from part="parameters" variable="ResourcesPLResponse2">
                <bpel:query
queryLanguage="urn:oasis:names:tc:wsbpel:2.0:sublang:xpath1.0"><!
[CDATA[ns:initiateSchedulerReturn]]></bpel:query>
              </bpel:from>
              <bpel:to part="parameters" variable="GuiAccessPLRequest4">
                <bpel:query
queryLanguage="urn:oasis:names:tc:wsbpel:2.0:sublang:xpath1.0"><!
[CDATA[ns:str]]></bpel:query>
              </bpel:to>
            </bpel:copy>
          </bpel:assign>
        </bpel:sequence>
      </bpel:elseif><bpel:elseif>
        <bpel:condition><!
[CDATA[contains($GuiAccessPLResponse.parameters/ns:getLatestAnnotationsReturn, 'emergency
operator')]]></bpel:condition>
        <bpel:sequence name="Emergency Operator">
          <bpel:assign validate="no" name="Assign29">
            <bpel:copy>
              <bpel:from>
                <bpel:literal xml:space="preserve"><impl:availableResources
xmlns:impl="http://code" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <impl:str></impl:str>
</impl:availableResources>
</bpel:literal>
              </bpel:from>
              <bpel:to variable="ResourcesPLRequest" part="parameters"></bpel:to>
            </bpel:copy>
            <bpel:copy>
              <bpel:from part="parameters" variable="GuiAccessPLResponse1">
              <bpel:query
queryLanguage="urn:oasis:names:tc:wsbpel:2.0:sublang:xpath1.0"><!
[CDATA[ns:initiateSchedulerReturn]]></bpel:query>
            </bpel:from>
            <bpel:to part="parameters" variable="ResourcesPLRequest">
              <bpel:query
queryLanguage="urn:oasis:names:tc:wsbpel:2.0:sublang:xpath1.0"><!
[CDATA[ns:str]]></bpel:query>
            </bpel:to>
          </bpel:copy>
        </bpel:assign>
        <bpel:invoke name="Available Resources" partnerLink="ResourcesPL"
operation="availableResources" portType="ns:Resources" inputVariable="ResourcesPLRequest"
outputVariable="ResourcesPLResponse"></bpel:invoke>
        <bpel:assign validate="no" name="Assign30">
          <bpel:copy>
            <bpel:from>
              <bpel:literal
xml:space="preserve"><impl:opemergencyTaskOrganiseruserOperation xmlns:impl="http://code"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <impl:str></impl:str>
</impl:opemergencyTaskOrganiseruserOperation>
</bpel:literal>
            </bpel:from>
            <bpel:to variable="SchedulerPLRequest1" part="parameters"></bpel:to>
          </bpel:copy>
          <bpel:copy>
            <bpel:from part="parameters" variable="ResourcesPLResponse">
              <bpel:query
queryLanguage="urn:oasis:names:tc:wsbpel:2.0:sublang:xpath1.0"><!
[CDATA[ns:stopCalendarReturn]]></bpel:query>
            </bpel:from>
            <bpel:to part="parameters" variable="SchedulerPLRequest1">
              <bpel:query
queryLanguage="urn:oasis:names:tc:wsbpel:2.0:sublang:xpath1.0"><!
[CDATA[ns:str]]></bpel:query>
            </bpel:to>
          </bpel:copy>
        </bpel:assign>
        <bpel:invoke name="Scheduler (UI)" partnerLink="SchedulerPL"
operation="opemergencyTaskOrganiseruserOperation" portType="ns:Scheduler"
inputVariable="SchedulerPLRequest1" outputVariable="SchedulerPLResponse1"></bpel:invoke>
        <bpel:assign validate="no" name="Assign31">
          <bpel:copy>
            <bpel:from>
              <bpel:literal xml:space="preserve"><impl:updateResources
xmlns:impl="http://code" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <impl:str></impl:str>
</impl:updateResources>
</bpel:literal>
            </bpel:from>
            <bpel:to variable="ResourcesPLRequest2" part="parameters"></bpel:to>
          </bpel:copy>
```

```
                  <bpel:copy>
                     <bpel:from part="parameters" variable="SchedulerPLResponse1">
                        <bpel:query
queryLanguage="urn:oasis:names:tc:wsbpel:2.0:sublang:xpath1.0"><!
[CDATA[ns:stopCalendarReturn]]></bpel:query>
                     </bpel:from>
                     <bpel:to part="parameters" variable="ResourcesPLRequest2">
                        <bpel:query
queryLanguage="urn:oasis:names:tc:wsbpel:2.0:sublang:xpath1.0"><!
[CDATA[ns:str]]></bpel:query>
                     </bpel:to>
                  </bpel:copy>
               </bpel:assign>
               <bpel:invoke name="Update Resources" partnerLink="ResourcesPL"
operation="updateResources" portType="ns:Resources" inputVariable="ResourcesPLRequest2"
outputVariable="ResourcesPLResponse2"></bpel:invoke>
               <bpel:assign validate="no" name="Assign32">
                  <bpel:copy>
                     <bpel:from>
                        <bpel:literal xml:space="preserve"><impl:availableResources
xmlns:impl="http://code" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
   <impl:str></impl:str>
</impl:availableResources>
</bpel:literal>
                     </bpel:from>
                     <bpel:to variable="ResourcesPLRequest" part="parameters"></bpel:to>
                  </bpel:copy>
                  <bpel:copy>
                     <bpel:from part="parameters" variable="GuiAccessPLResponse1">
                        <bpel:query
queryLanguage="urn:oasis:names:tc:wsbpel:2.0:sublang:xpath1.0"><!
[CDATA[ns:initiateSchedulerReturn]]></bpel:query>
                     </bpel:from>
                     <bpel:to part="parameters" variable="ResourcesPLRequest">
                        <bpel:query
queryLanguage="urn:oasis:names:tc:wsbpel:2.0:sublang:xpath1.0"><!
[CDATA[ns:str]]></bpel:query>
                     </bpel:to>
                  </bpel:copy>
               </bpel:assign>
            </bpel:sequence>
         </bpel:elseif></bpel:if>
      <bpel:invoke name="exit" partnerLink="GuiAccessPL" operation="exit"
portType="ns:GuiAccess" inputVariable="GuiAccessPLRequest4"
outputVariable="GuiAccessPLResponse4"></bpel:invoke>
      <bpel:assign validate="no" name="Assign19">
         <bpel:copy>
            <bpel:from>
               <bpel:literal xml:space="preserve"><tns:output xmlns:tns="atc"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
</tns:output>
</bpel:literal>
            </bpel:from>
```

```
        <bpel:to variable="output" part="payload"></bpel:to>
      </bpel:copy>
      <bpel:copy>
        <bpel:from part="parameters" variable="GuiAccessPLResponse4">
          <bpel:query queryLanguage="urn:oasis:names:tc:wsbpel:2.0:sublang:xpath1.0"><!
[CDATA[ns:str]]></bpel:query>
        </bpel:from>
        <bpel:to part="payload" variable="output"></bpel:to>
      </bpel:copy>
    </bpel:assign>
    <bpel:reply name="replyOutput"
        partnerLink="client"
        portType="tns:ATC"
        operation="process"
        variable="output"
        />
  </bpel:sequence>
</bpel:process>
```

Appendix B

Atc-SBS Results



Atc-SBS

Context: Role. Repository: Empty Rule Creation

N° of Rules	Mean Value	Min Value	Max Value	Median	Standard Deviatic
1	0.1761044748	0.083012723	0.27549048	0.1263563985	0.0502591238
2	0.2907518361	0.162334752	0.698459212	0.2357327335	0.0983423419
3	0.3639144053	0.138486258	0.642812817	0.3118452785	0.1277973348
4	0.4667290736	0.175697413	0.985197338	0.427089404	0.1762612302
5	0.4967074044	0.223415696	1.130799244	0.3834292065	0.2177593832
6	0.5531220345	0.268119152	1.052719274	0.375062675	0.2316662246
7	0.6360379086	0.324955428	1.681374638	0.418707959	0.3029814154
8	0.6883086969	0.366230092	1.449931518	0.5008979125	0.2773262003
9	0.7398645082	0.410288699	1.368113712	0.545191745	0.2759962553
10	0.822149651	0.462053098	2.037519995	0.533052457	0.3753122942

Context: Role. Repository: 100 Rules. Rule Creation (20% match invariant)

N° of Rules	Mean Value	Min Value	Max Value	Median	Standard Deviatic
1	0.4509289416	0.238191535	0.643572901	0.389316685	0.1242161002
2	0.6723360958	0.243570288	1.091717041	0.596974829	0.2342727124
3	0.8039718297	0.366704804	1.31360922	0.6493477605	0.3210116583
4	0.9831337326	0.466794487	1.831154158	0.7056361715	0.4467075514
5	1.118046261	0.598255479	2.747481162	0.809747875	0.539054472
6	1.2666591916	0.692426529	3.176622603	0.913635701	0.5812581602
7	1.5587546105	0.960913603	2.50137968	1.0850700685	0.5263134678
8	1.6219381296	0.903498483	2.749394175	1.2964734205	0.5716435643
9	1.8751955985	1.067111111	4.646811366	1.445890289	0.7576942216
10	1.9968248015	1.168713984	4.783683035	1.533144229	0.7478880837

Context: Role. Repository: 100 Rules. Rule Identification (20% match invariant)

N° of Rules	Mean Value	Min Value	Max Value	Median	Standard Deviatic
1	0.3940541978	0.285527021	0.578482011	0.321316386	0.0843353526
2	0.6097021022	0.2427081	0.764556165	0.5782167555	0.1625127068
3	0.7589006608	0.361316826	1.041305673	0.7341724815	0.2536932616
4	0.8405292646	0.467516224	1.468626769	0.5246017685	0.3260297643
5	0.9778832137	0.579073289	2.103151388	0.638308231	0.4285956237
6	1.0925110382	0.685554011	2.334806614	0.765326755	0.4584488817
7	1.2911395202	0.767972242	2.87112167	0.886851328	0.5405549334
8	1.4671876712	0.930550528	2.699579813	1.0473540645	0.5127859208
9	1.6579056895	0.972901725	4.298534753	1.182326015	0.7053326969
10	1.7445678863	1.096445649	4.338114016	1.2920273585	0.7105468524

Context: Role. Repository: 100 Rules. Rule Modification (20% match invariant)

N° of Rules	Mean Value	Min Value	Max Value	Median	Standard Deviatic
1	0.4591491068	0.259641539	0.572036151	0.367840596	0.1084809341
2	0.6664191833	0.243647738	0.964107937	0.594460437	0.2430250346
3	0.8104550586	0.362419065	1.450739788	0.565359883	0.3748944488
4	0.9592881655	0.469452992	1.777426099	0.689470148	0.3912280824
5	1.0782159449	0.599913859	2.084402213	0.756002598	0.4478056412
6	1.2414936282	0.689646147	2.448141346	0.918316847	0.4855433438
7	1.3042513343	0.794411493	2.34449311	0.8973505995	0.4588218723
8	1.4656994295	0.940579242	2.632960935	1.1482042185	0.4824311126
9	1.6153591902	0.980181418	2.834018771	1.206063212	0.5190243869
10	1.6577643363	1.034170132	2.628268579	1.2416724205	0.5349810277

Context: Role. Repository: 100 Rules. Rule Removal (20% match invariant)

N° of Rules	Mean Value	Min Value	Max Value	Median	Standard Deviatic
1	0.4569482124	0.247614796	0.62327779	0.3671122555	0.1324643241
2	0.4727687135	0.252914984	0.539475446	0.4096829275	0.1058109676
3	0.4856586908	0.256420526	0.622796659	0.4205145065	0.125995987
4	0.4568347086	0.215589565	0.534472505	0.3909258275	0.1044892698
5	0.4624633041	0.22073289	0.619575297	0.3829977955	0.1169325762
6	0.4542313906	0.263721245	0.669734536	0.3677898545	0.120114941
7	0.462523346	0.252077657	0.585403353	0.3820567205	0.117756143
8	0.4544361883	0.243467903	0.638351284	0.358741148	0.134156171
9	0.44887172	0.218787391	0.611085597	0.3598190105	0.1303527395
10	0.4664755034	0.227451059	0.547765396	0.407657528	0.1093873804

Context: Role. Repository: 200 Rules. Rule Creation (20% match invariant)

N° of Rules	Mean Value	Min Value	Max Value	Median	Standard Deviatic
1	0.628038856	0.205367662	0.895787656	0.5333897205	0.182113048
2	0.8442600814	0.380723907	1.3532892	0.6805054605	0.3466393911
3	1.0486652871	0.563895313	1.934431186	0.748576999	0.4416310056
4	1.2779272274	0.73886581	2.357765951	0.968761842	0.5211604332
5	1.4923175467	0.893639616	4.331060607	1.095174747	0.734453577
6	1.7772730992	1.079846118	4.475218052	1.3486386745	0.7139926559
7	1.9475090558	1.28383107	3.706509535	1.5262145135	0.6167652802
8	2.1656512953	1.511532019	4.091828855	1.674954404	0.6748214686
9	2.3213611545	1.710342288	5.103639433	1.839539197	0.737528422
10	2.579760117	1.880762282	6.169097276	2.037320109	0.9083939108

Context: Role. Repository: 200 Rules. Rule Identification (20% match invariant)

N° of Rules	Mean Value	Min Value	Max Value	Median	Standard Deviatic
1	0.5064938465	0.203911964	0.570915343	0.4949637085	0.1528820605
2	0.7119906148	0.378589203	1.143231556	0.4548232695	0.278456277
3	0.9418386777	0.554367431	1.67191338	0.5931476665	0.3725828667
4	1.1074278019	0.720860033	1.987210315	0.771907203	0.4306631458
5	1.3688413074	0.904705828	3.937763802	0.9479806065	0.6627478569
6	1.6011523111	1.068364208	4.223236977	1.1336911625	0.6899494448
7	1.8301291735	1.245127192	3.673323123	1.3928797425	0.6128570663
8	1.96182417	1.327751806	4.620774477	1.4885184065	0.7325899246
9	2.1312339926	1.520252889	5.013664024	1.6810000775	0.7298954707
10	2.3574486825	1.731374709	5.710989924	1.8939190515	0.7790819714

Context: Role. Repository: 200 Rules. Rule Modification (20% match invariant)

N° of Rules	Mean Value	Min Value	Max Value	Median	Standard Deviatic
1	0.4950524545	0.208143227	0.600814668	0.4470547175	0.149208774
2	0.7291279382	0.377703893	1.181847077	0.4434630265	0.2930857756
3	0.9189424165	0.548801563	1.849857794	0.574071395	0.3940212139
4	1.1043786901	0.717520153	1.914956923	0.764549806	0.4245724062
5	1.349297554	0.882898934	3.757388215	0.9305021985	0.6661745265
6	1.5610482245	1.061711134	4.033367721	1.118646407	0.6509269608
7	1.752134872	1.200251796	3.63939052	1.323846174	0.6119196441
8	1.9937813119	1.376343242	4.5092385	1.5326185685	0.6830772726
9	2.1010682594	1.499169483	4.768279891	1.650947266	0.6963856328
10	2.286283942	1.678506948	5.195217699	1.846721292	0.7124705599

Context: Role. Repository: 200 Rules. Rule Removal (20% match invariant)

N° of Rules	Mean Value	Min Value	Max Value	Median	Standard Deviatic
1	0.5951667357	0.24006286	0.829857063	0.5422723825	0.1683232138
2	0.6202760186	0.20303497	0.872322243	0.528183156	0.1785050962
3	0.6451347361	0.31371175	0.1683232138	0.5603426575	0.1638271769
4	0.6549595119	0.254115415	0.824719679	0.545515776	0.1635581879
5	0.6192459926	0.215666185	0.81291378	0.5734749865	0.1823477726
6	0.6066737049	0.249508124	0.815075435	0.51674637	0.160080833
7	0.6146774183	0.231219487	0.85419019	0.5301099405	0.1793511272
8	0.5994066236	0.250631943	0.839465477	0.550341735	0.1798025911
9	0.6188485813	0.412536747	0.78502287	0.500608982	0.1373110949
10	0.5706217346	0.213104545	0.855193593	0.4890716195	0.1926232079

Context: Role. Repository: 300 Rules. Rule Creation (20% match invariant)

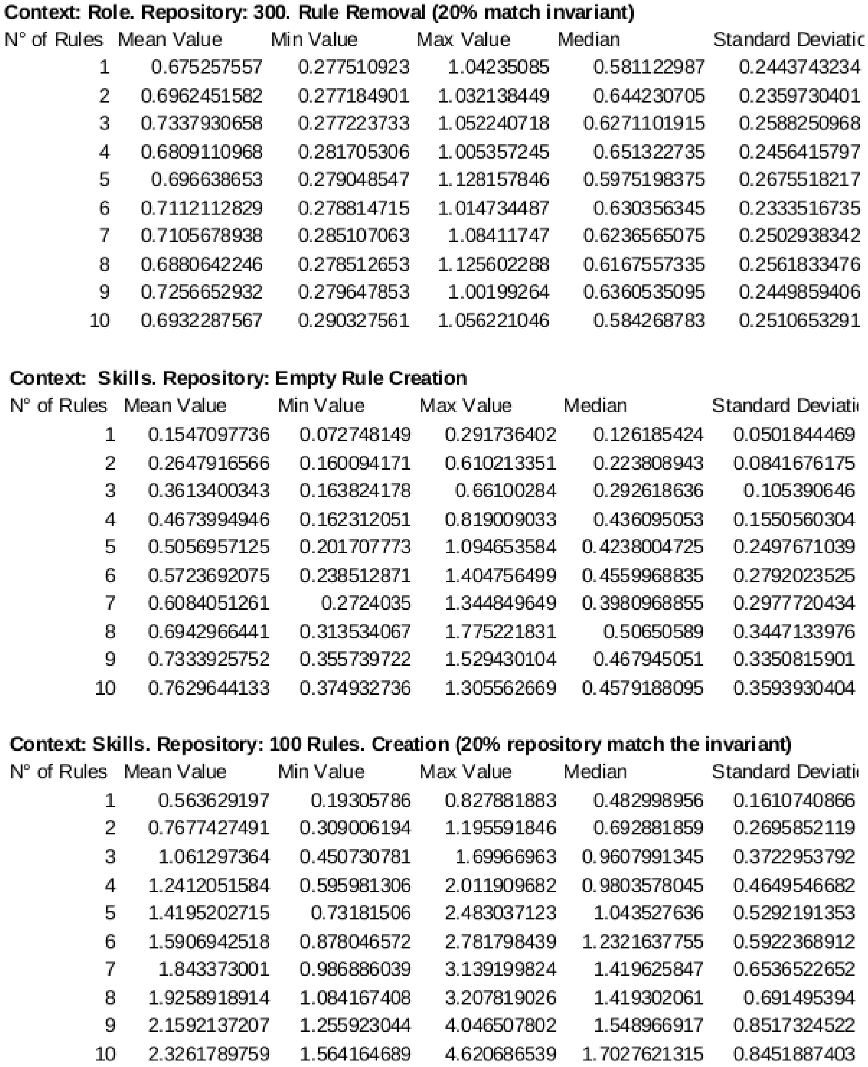
N° of Rules	Mean Value	Min Value	Max Value	Median	Standard Deviatic
1	0.6841921199	0.2849259	1.055806051	0.621357371	0.2422526148
2	1.0301346471	0.510532452	2.328403769	0.7092437905	0.4735499795
3	1.3559046178	0.745749997	3.673959378	0.955031571	0.7463892127
4	1.6113130121	0.975661921	4.665529046	1.196625215	0.7709944858
5	1.9329500299	1.285643585	3.807883039	1.5202374875	0.6580242359
6	2.1794783542	1.616671159	4.133908928	1.744800893	0.6354222463
7	2.4450776995	1.811013604	4.896426631	1.9824414145	0.6926884825
8	2.6413287778	1.99673764	5.509827048	2.1734128335	0.7440226966
9	2.9340149623	2.286320837	5.301241526	2.4665864265	0.7078534639
10	3.1751752463	2.485478989	7.018909179	2.623812917	0.9028692401

Context: Role. Repository: 300. Rule Identification (20% match invariant)

N° of Rules	Mean Value	Min Value	Max Value	Median	Standard Deviatic
1	0.6661293008	0.272887076	1.431746665	0.590313057	0.2468363495
2	0.884207822	0.519617337	1.588112986	0.5438012045	0.3661752063
3	1.164080807	0.746456866	2.616562483	0.7933185105	0.514455089
4	1.4714075491	1.01675073	4.739589022	1.064143492	0.7412598415
5	1.7083000335	1.171242797	3.179119635	1.2924453225	0.5923384712
6	1.9557281091	1.393565341	3.898861445	1.5465703085	0.6274570068
7	2.1874789159	1.593065428	4.338617012	1.763736891	0.6548445398
8	2.4517750776	1.848623068	4.853908865	2.0234129265	0.6915276161
9	2.6806127496	2.054894894	5.680319122	2.238178718	0.7680568315
10	2.9471448169	2.303910714	6.025217242	2.3781581115	0.7844291861

Context: Role. Repository: 300. Rule Modification (20% match invariant)

N° of Rules	Mean Value	Min Value	Max Value	Median	Standard Deviatic
1	0.6320240585	0.279176145	0.822161461	0.5937692885	0.1691067557
2	0.9013416373	0.538209499	1.578732601	0.567429452	0.3759292659
3	1.1860309004	0.742059095	2.37147866	0.7911353145	0.510142209
4	1.4521623772	0.989289772	3.127399715	1.0366999175	0.6173638747
5	1.7634169712	1.183727554	3.668414186	1.325223552	0.6049279402
6	1.9841330762	1.37349135	4.946564151	1.5268685705	0.7419911228
7	2.2321209627	1.615760635	5.022770071	1.763411291	0.7283119829
8	2.3925694413	1.828083514	4.189370522	1.993453218	0.5887013166
9	2.691611152	2.077437781	5.527849809	2.245041634	0.7522087964
10	3.067794368	2.379381828	6.769431508	2.4428853345	0.8894631465



Context: Role. Repository: 300. Rule Removal (20% match invariant)

N° of Rules	Mean Value	Min Value	Max Value	Median	Standard Deviatic
1	0.675257557	0.277510923	1.04235085	0.581122987	0.2443743234
2	0.6962451582	0.277184901	1.032138449	0.644230705	0.2359730401
3	0.7337930658	0.277223733	1.052240718	0.6271101915	0.2588250968
4	0.6809110968	0.281705306	1.005357245	0.651322735	0.2456415797
5	0.696638653	0.279048547	1.128157846	0.5975198375	0.2675518217
6	0.7112112829	0.278814715	1.014734487	0.630356345	0.2333516735
7	0.7105678938	0.285107063	1.08411747	0.6236565075	0.2502938342
8	0.6880642246	0.278512653	1.125602288	0.6167557335	0.2561833476
9	0.7256652932	0.279647853	1.00199264	0.6360535095	0.2449859406
10	0.6932287567	0.290327561	1.056221046	0.584268783	0.2510653291

Context: Skills. Repository: Empty Rule Creation

N° of Rules	Mean Value	Min Value	Max Value	Median	Standard Deviati
1	0.1547097736	0.072748149	0.291736402	0.126185424	0.0501844469
2	0.2647916566	0.160094171	0.610213351	0.223808943	0.0841676175
3	0.3613400343	0.163824178	0.66100284	0.292618636	0.105390646
4	0.4673994946	0.162312051	0.819009033	0.436095053	0.1550560304
5	0.5056957125	0.201707773	1.094653584	0.4238004725	0.2497671039
6	0.5723692075	0.238512871	1.404756499	0.4559968835	0.2792023525
7	0.6084051261	0.2724035	1.344849649	0.3980968855	0.2977720434
8	0.6942966441	0.313534067	1.775221831	0.50650589	0.3447133976
9	0.7333925752	0.355739722	1.529430104	0.467945051	0.3350815901
10	0.7629644133	0.374932736	1.305562669	0.4579188095	0.3593930404

Context: Skills. Repository: 100 Rules. Creation (20% repository match the invariant)

N° of Rules	Mean Value	Min Value	Max Value	Median	Standard Deviati
1	0.563629197	0.19305786	0.827881883	0.482998956	0.1610740866
2	0.7677427491	0.309006194	1.195591846	0.692881859	0.2695852119
3	1.061297364	0.450730781	1.69966963	0.9607991345	0.3722953792
4	1.2412051584	0.595981306	2.011909682	0.9803578045	0.4649546682
5	1.4195202715	0.73181506	2.483037123	1.043527636	0.5292191353
6	1.5906942518	0.878046572	2.781798439	1.2321637755	0.5922368912
7	1.843373001	0.986886039	3.139199824	1.419625847	0.6536522652
8	1.9258918914	1.084167408	3.207819026	1.419302061	0.691495394
9	2.1592137207	1.255923044	4.046507802	1.548966917	0.8517324522
10	2.3261789759	1.564164689	4.620686539	1.7027621315	0.8451887403

Context: Skills. Repository: 100 Rules. Identification (20% repository match the invariant)

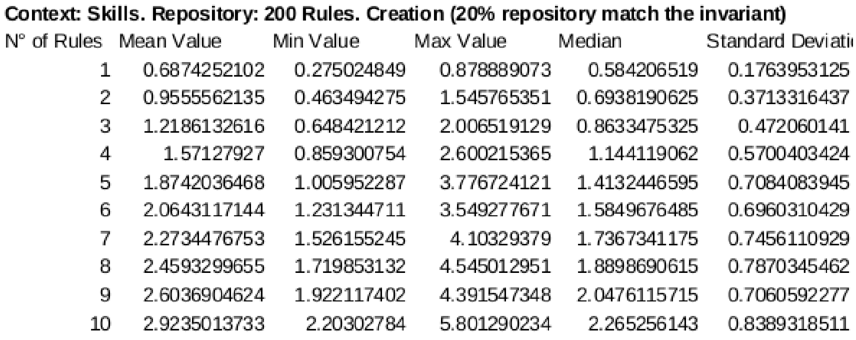
N° of Rules	Mean Value	Min Value	Max Value	Median	Standard Deviati
1	0.4566047825	0.300227214	0.533813274	0.397175681	0.074904258
2	0.6934119224	0.307212388	0.95206604	0.636595684	0.2123995611
3	0.9424859287	0.437056626	1.193067708	0.9314315015	0.2843451549
4	1.1200240998	0.592253456	1.671874268	0.8830075065	0.3873717203
5	1.284781253	0.722947029	2.110933102	0.959946897	0.4589405033
6	1.4582621725	0.809586607	2.398850295	1.1455363865	0.520070314
7	1.5821612135	0.944214766	2.727770618	1.1472681665	0.5810544679
8	1.739547811	1.100628653	2.879340453	1.221709929	0.6340898533
9	1.8822026415	1.215889324	3.309653555	1.3658184475	0.6766251987
10	2.070990994	1.36263082	3.791240355	1.459906282	0.7612602001

Context: Skills. Repository: 100 Rules. Modification (20% repository match the invariant)

N° of Rules	Mean Value	Min Value	Max Value	Median	Standard Deviati
1	0.4541281609	0.318828324	0.535209333	0.3774904235	0.0865357953
2	0.6924656459	0.303955971	0.857877389	0.677785633	0.1893045801
3	0.9146064012	0.447573749	1.274535963	0.8244717105	0.3228038279
4	1.0743408944	0.597680752	1.713564552	0.8570423985	0.366866618
5	1.2690550701	0.699518004	1.950712305	0.988320616	0.4280959148
6	1.3956637265	0.802786509	2.584807544	1.06138829	0.533729205
7	1.5863572791	0.96115686	2.684158714	1.1900232885	0.523352576
8	1.7091152535	1.08360288	3.157410171	1.2165517025	0.615403266
9	1.9142564171	1.230448995	3.327946663	1.375643995	0.6689469957
10	2.0057416358	1.374035484	3.71763232	1.480055529	0.6830700555

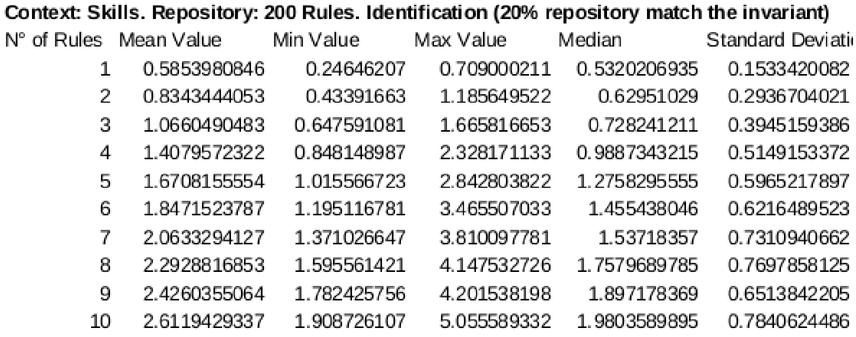
Context: Skills. Repository: 100 Rules. Removal (20% repository match the invariant)

N° of Rules	Mean Value	Min Value	Max Value	Median	Standard Deviati
1	0.4511604642	0.261341542	0.58131143	0.3816665505	0.090635393
2	0.460764101	0.312885109	0.566645241	0.3819855145	0.08949344
3	0.4593497403	0.32478515	0.586326726	0.3825300365	0.084409353
4	0.4585015942	0.313607546	0.525355839	0.378297757	0.08496829
5	0.4601245285	0.308146029	0.518282385	0.3815814495	0.0828228934
6	0.4566653252	0.341456762	0.508736548	0.3797168605	0.0749222546
7	0.4336889004	0.3022487	0.524393431	0.372780577	0.0716965368
8	0.4589318172	0.307662933	0.647689346	0.377305907	0.0935848269
9	0.4498745736	0.287705234	0.498517584	0.381682369	0.0789702111
10	0.4456614886	0.283889106	0.627101393	0.3637067585	0.095141101



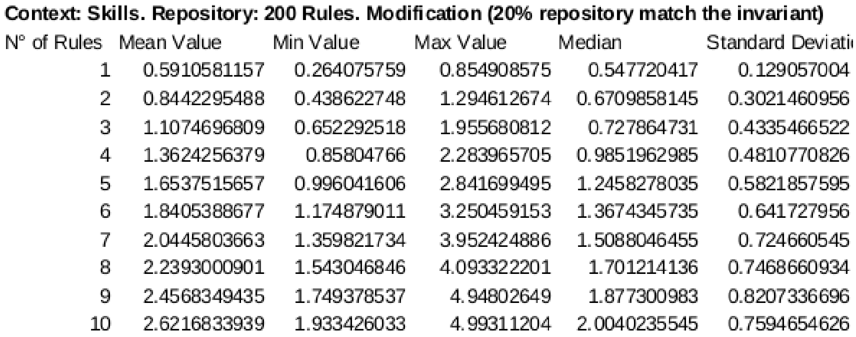
Context: Skills. Repository: 200 Rules. Creation (20% repository match the invariant)

N° of Rules	Mean Value	Min Value	Max Value	Median	Standard Deviati
1	0.6874252102	0.275024849	0.878889073	0.584206519	0.1763953125
2	0.9555562135	0.463494275	1.545765351	0.6938190625	0.3713316437
3	1.2186132616	0.648421212	2.006519129	0.8633475325	0.472060141
4	1.57127927	0.859300754	2.600215365	1.144119062	0.5700403424
5	1.8742036468	1.005952287	3.776724121	1.4132446595	0.7084083945
6	2.0643117144	1.231344711	3.549277671	1.5849676485	0.6960310429
7	2.2734476753	1.526155245	4.10329379	1.7367341175	0.7456110929
8	2.4593299655	1.719853132	4.545012951	1.8898690615	0.7870345462
9	2.6036904624	1.922117402	4.391547348	2.0476115715	0.7060592277
10	2.9235013733	2.20302784	5.801290234	2.265256143	0.8389318511



Context: Skills. Repository: 200 Rules. Identification (20% repository match the invariant)

N° of Rules	Mean Value	Min Value	Max Value	Median	Standard Deviati
1	0.5853980846	0.24646207	0.709000211	0.5320206935	0.1533420082
2	0.8343444053	0.43391663	1.185649522	0.62951029	0.2936704021
3	1.0660490483	0.647591081	1.665816653	0.728241211	0.3945159386
4	1.4079572322	0.848148987	2.328171133	0.9887343215	0.5149153372
5	1.6708155554	1.015566723	2.842803822	1.2758295555	0.5965217897
6	1.8471523787	1.195116781	3.465507033	1.455438046	0.6216489523
7	2.0633294127	1.371026647	3.810097781	1.53718357	0.7310940662
8	2.2928816853	1.595561421	4.147532726	1.7579689785	0.7697858125
9	2.4260355064	1.782425756	4.201538198	1.897178369	0.6513842205
10	2.6119429337	1.908726107	5.055589332	1.9803589895	0.7840624486



Context: Skills. Repository: 200 Rules. Modification (20% repository match the invariant)

N° of Rules	Mean Value	Min Value	Max Value	Median	Standard Deviati
1	0.5910581157	0.264075759	0.854908575	0.547720417	0.129057004
2	0.8442295488	0.438622748	1.294612674	0.6709858145	0.3021460956
3	1.1074696809	0.652292518	1.955680812	0.727864731	0.4335466522
4	1.3624256379	0.85804766	2.283965705	0.9851962985	0.4810770826
5	1.6537515657	0.996041606	2.841699495	1.2458278035	0.5821857595
6	1.8405388677	1.174879011	3.250459153	1.3674345735	0.641727956
7	2.0445803663	1.359821734	3.952424886	1.5088046455	0.724660545
8	2.2393000901	1.543046846	4.093322201	1.701214136	0.7468660934
9	2.4568349435	1.749378537	4.94802649	1.877300983	0.8207336696
10	2.6216833939	1.933426033	4.99311204	2.0040235545	0.7594654626

Context: Skills. Repository: 200 Rules. Rule Removal (20% match the invariant)

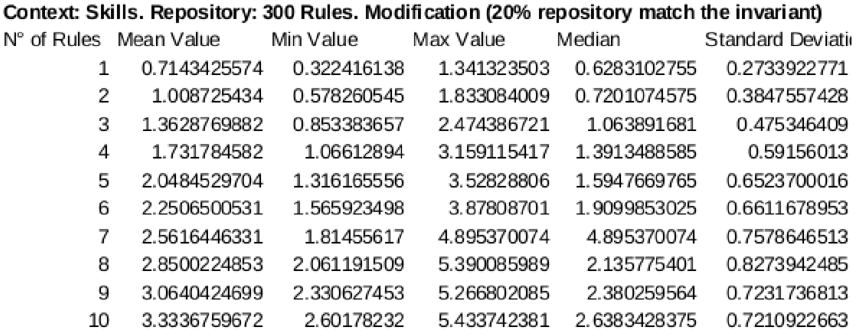
N° of Rules	Mean Value	Min Value	Max Value	Median	Standard Deviati
1	0.599639542	0.332623509	0.772665224	0.5254662985	0.1193938121
2	0.6165136316	0.327583906	0.787654841	0.542231557	0.1223021642
3	0.61448542	0.243067426	0.796960986	0.5427003675	0.1848653968
4	0.5917282197	0.263505575	0.763092294	0.552876	0.166996472
5	0.6207387685	0.288715625	0.755697562	0.537757071	0.1434157031
6	0.5825288911	0.248574978	0.765251853	0.511832651	0.1159458072
7	0.5687285649	0.241198479	0.732589179	0.5272061145	0.1683281816
8	0.5714257635	0.242198468	0.774193078	0.557307321	0.1625101714
9	0.5753584771	0.241660759	0.779731627	0.5353930485	0.1550511982
10	0.5727550387	0.251380499	0.840345684	0.525494619	0.1756204895

Context: Skills. Repository: 300 Rules. Creation (20% repository match the invariant)

N° of Rules	Mean Value	Min Value	Max Value	Median	Standard Deviati
1	0.7608858267	0.317212899	1.721870637	0.6412929335	0.3162730602
2	1.0341482254	0.575030947	1.879327744	0.6874884365	0.4258738363
3	1.3560696066	0.865267574	2.417180494	1.046852569	0.4862642612
4	1.7140572578	1.071370948	2.985168157	1.3526079655	0.5580642003
5	2.0388525432	1.37310778	3.731594777	1.6641291145	0.6303104864
6	2.2734462499	1.562122526	3.91842978	1.848993368	0.6931694458
7	2.5780902725	1.791051156	5.088871913	2.066076173	0.82900456
8	2.8698468837	2.117034718	5.417358941	2.184883951	0.8114295491
9	3.2031692556	2.404743232	5.27204954	2.4476729545	0.7412029709
10	3.4117575081	2.629140062	6.346562441	2.666338365	0.8530579761

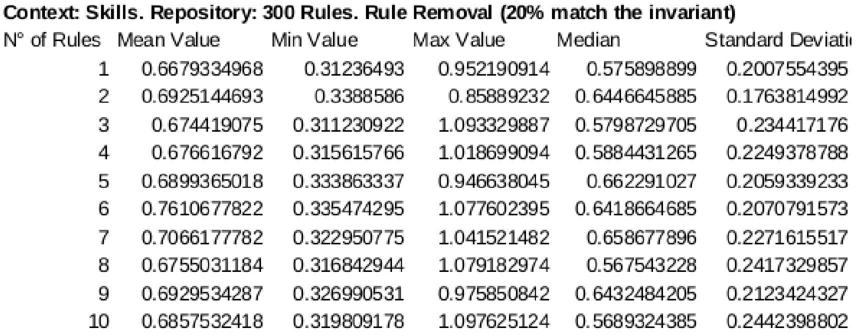
Context: Skills. Repository: 300 Rules. Identification (20% repository match the invariant)

N° of Rules	Mean Value	Min Value	Max Value	Median	Standard Deviati
1	0.7540182567	0.312939867	1.071014484	0.707392644	0.2299872421
2	1.0157392504	0.580893493	1.705476442	0.7182601225	0.3849969757
3	1.4030155462	0.85806938	2.658281547	1.0770559595	0.5246878022
4	1.7263163585	1.069738132	3.106048681	1.3922827765	0.5377423555
5	2.066571906	1.330130671	3.894030308	1.623468563	0.674427896
6	2.3077641121	1.574289585	4.287548579	1.862341925	0.7479151072
7	2.5715508801	1.807587407	5.205414394	2.044872186	0.8071102017
8	2.8948255497	2.092142161	5.626966947	2.1357683485	0.8376457307
9	3.1066083519	2.348225639	5.45420986	2.397578105	0.7706518448
10	3.3545335039	2.591046116	6.017648682	2.6336401105	0.8198416122



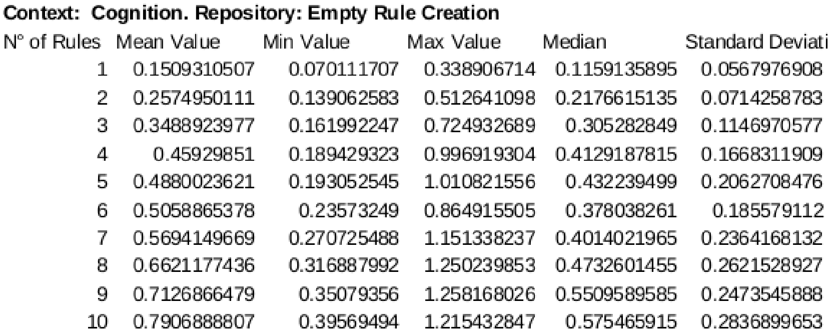
Context: Skills. Repository: 300 Rules. Modification (20% repository match the invariant)

N° of Rules	Mean Value	Min Value	Max Value	Median	Standard Deviati
1	0.7143425574	0.322416138	1.341323503	0.6283102755	0.2733922771
2	1.008725434	0.578260545	1.833084009	0.7201074575	0.3847557428
3	1.3628769882	0.853383657	2.474386721	1.063891681	0.475346409
4	1.731784582	1.06612894	3.159115417	1.3913488585	0.59156013
5	2.0484529704	1.316165556	3.52828806	1.5947669765	0.6523700016
6	2.2506500531	1.565923498	3.87808701	1.9099853025	0.6611678953
7	2.5616446331	1.81455617	4.895370074	4.895370074	0.7578646513
8	2.8500224853	2.061191509	5.390085989	2.135775401	0.8273942485
9	3.0640424699	2.330627453	5.266802085	2.380259564	0.7231736813
10	3.3336759672	2.60178232	5.433742381	2.6383428375	0.7210922663



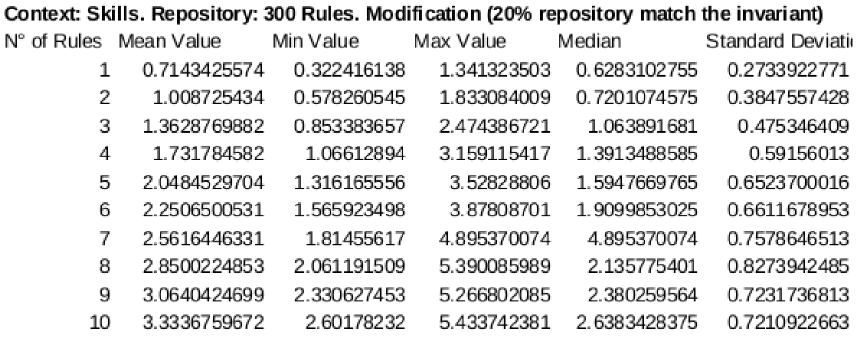
Context: Skills. Repository: 300 Rules. Rule Removal (20% match the invariant)

N° of Rules	Mean Value	Min Value	Max Value	Median	Standard Deviati
1	0.6679334968	0.31236493	0.952190914	0.575898899	0.2007554395
2	0.6925144693	0.3388586	0.85889232	0.6446645885	0.1763814992
3	0.674419075	0.311230922	1.093329887	0.5798729705	0.234417176
4	0.676616792	0.315615766	1.018699094	0.5884431265	0.2249378788
5	0.6899365018	0.333863337	0.946638045	0.662291027	0.2059339233
6	0.7610677822	0.335474295	1.077602395	0.6418664685	0.2070791573
7	0.7066177782	0.322950775	1.041521482	0.658677896	0.2271615517
8	0.6755031184	0.316842944	1.079182974	0.567543228	0.2417329857
9	0.6929534287	0.326990531	0.975850842	0.6432484205	0.2123424327
10	0.6857532418	0.319809178	1.097625124	0.5689324385	0.2442398802



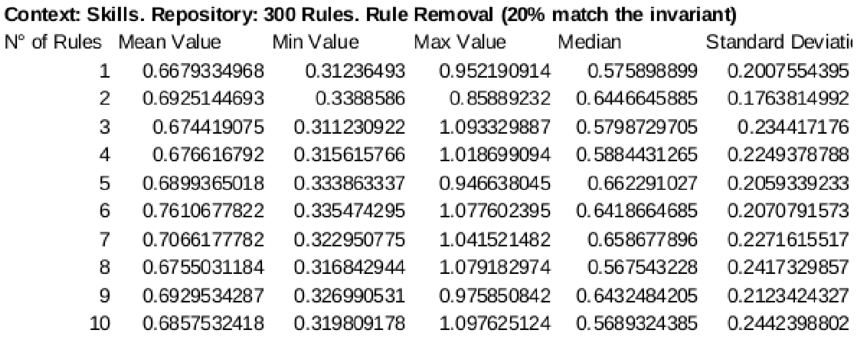
Context: Cognition. Repository: Empty Rule Creation

N° of Rules	Mean Value	Min Value	Max Value	Median	Standard Deviati
1	0.1509310507	0.070111707	0.338906714	0.1159135895	0.0567976908
2	0.2574950111	0.139062583	0.512641098	0.2176615135	0.0714258783
3	0.3488923977	0.161992247	0.724932689	0.305282849	0.1146970577
4	0.45929851	0.189429323	0.996919304	0.4129187815	0.1668311909
5	0.4880023621	0.193052545	1.010821556	0.432239499	0.2062708476
6	0.5058865378	0.23573249	0.864915505	0.378038261	0.185579112
7	0.5694149669	0.270725488	1.151338237	0.4014021965	0.2364168132
8	0.6621177436	0.316887992	1.250239853	0.4732601455	0.2621528927
9	0.7126866479	0.35079356	1.258168026	0.5509589585	0.2473545888
10	0.7906888807	0.39569494	1.215432847	0.575465915	0.2836899653



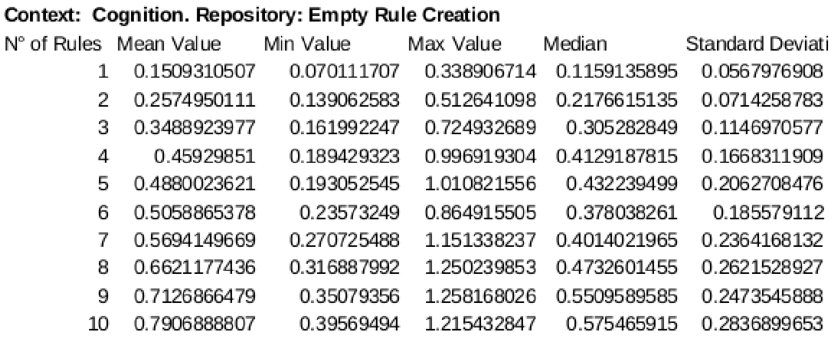
Context: Skills. Repository: 300 Rules. Modification (20% repository match the invariant)

N° of Rules	Mean Value	Min Value	Max Value	Median	Standard Deviati
1	0.7143425574	0.322416138	1.341323503	0.6283102755	0.2733922771
2	1.008725434	0.578260545	1.833084009	0.7201074575	0.3847557428
3	1.3628769882	0.853383657	2.474386721	1.063891681	0.475346409
4	1.731784582	1.06612894	3.159115417	1.3913488585	0.59156013
5	2.0484529704	1.316165556	3.52828806	1.5947669765	0.6523700016
6	2.2506500531	1.565923498	3.87808701	1.9099853025	0.6611678953
7	2.5616446331	1.81455617	4.895370074	4.895370074	0.7578646513
8	2.8500224853	2.061191509	5.390085989	2.135775401	0.8273942485
9	3.0640424699	2.330627453	5.266802085	2.380259564	0.7231736813
10	3.3336759672	2.60178232	5.433742381	2.6383428375	0.7210922663



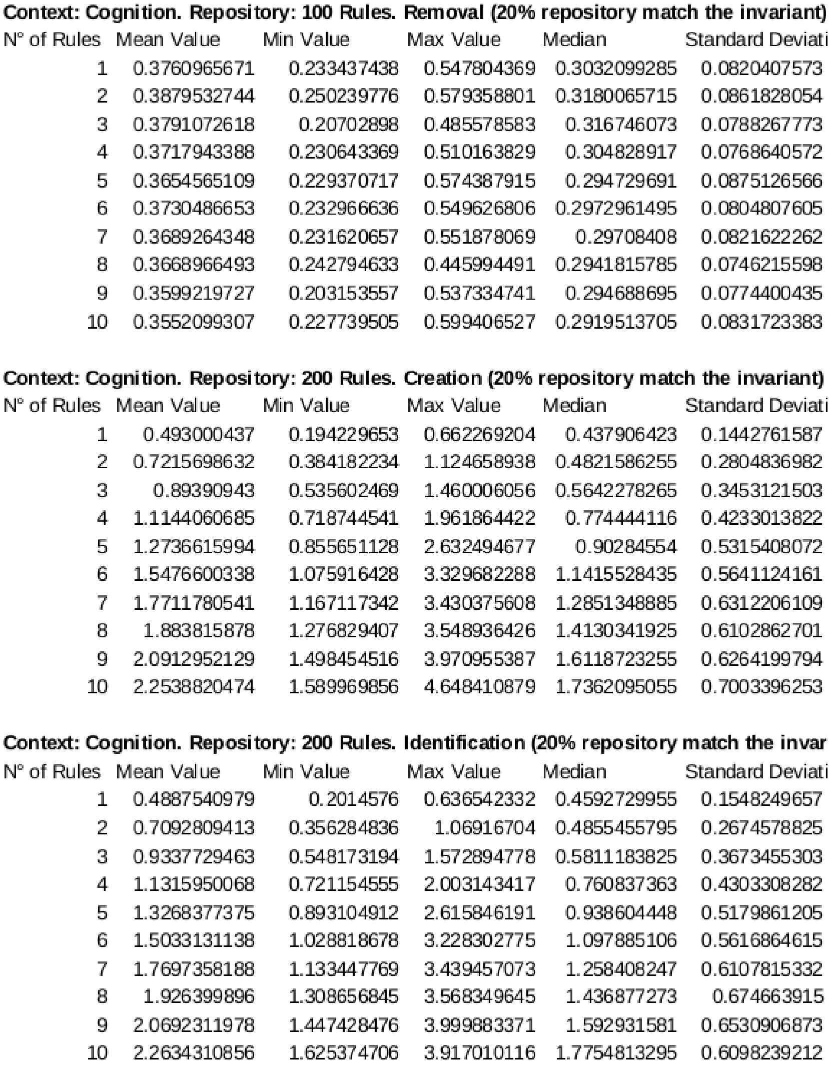
Context: Skills. Repository: 300 Rules. Rule Removal (20% match the invariant)

N° of Rules	Mean Value	Min Value	Max Value	Median	Standard Deviati
1	0.6679334968	0.31236493	0.952190914	0.575898899	0.2007554395
2	0.6925144693	0.3388586	0.85889232	0.6446645885	0.1763814992
3	0.674419075	0.311230922	1.093329887	0.5798729705	0.234417176
4	0.676616792	0.315615766	1.018699094	0.5884431265	0.2249378788
5	0.6899365018	0.333863337	0.946638045	0.662291027	0.2059339233
6	0.7610677822	0.335474295	1.077602395	0.6418664685	0.2070791573
7	0.7066177782	0.322950775	1.041521482	0.658677896	0.2271615517
8	0.6755031184	0.316842944	1.079182974	0.567543228	0.2417329857
9	0.6929534287	0.326990531	0.975850842	0.6432484205	0.2123424327
10	0.6857532418	0.319809178	1.097625124	0.5689324385	0.2442398802



Context: Cognition. Repository: Empty Rule Creation

N° of Rules	Mean Value	Min Value	Max Value	Median	Standard Deviati
1	0.1509310507	0.070111707	0.338906714	0.1159135895	0.0567976908
2	0.2574950111	0.139062583	0.512641098	0.2176615135	0.0714258783
3	0.3488923977	0.161992247	0.724932689	0.305282849	0.1146970577
4	0.45929851	0.189429323	0.996919304	0.4129187815	0.1668311909
5	0.4880023621	0.193052545	1.010821556	0.432239499	0.2062708476
6	0.5058865378	0.23573249	0.864915505	0.378038261	0.185579112
7	0.5694149669	0.270725488	1.151338237	0.4014021965	0.2364168132
8	0.6621177436	0.316887992	1.250239853	0.4732601455	0.2621528927
9	0.7126866479	0.35079356	1.258168026	0.5509589585	0.2473545888
10	0.7906888807	0.39569494	1.215432847	0.575465915	0.2836899653



Context: Cognition. Repository: 100 Rules. Removal (20% repository match the invariant)

N° of Rules	Mean Value	Min Value	Max Value	Median	Standard Deviati
1	0.3760965671	0.233437438	0.547804369	0.3032099285	0.0820407573
2	0.3879532744	0.250239776	0.579358801	0.3180065715	0.0861828054
3	0.3791072618	0.20702898	0.485578583	0.316746073	0.0788267773
4	0.3717943388	0.230643369	0.510163829	0.304828917	0.0768640572
5	0.3654565109	0.229370717	0.574387915	0.294729691	0.0875126566
6	0.3730486653	0.232966636	0.549626806	0.2972961495	0.0804807605
7	0.3689264348	0.231620657	0.551878069	0.29708408	0.0821622262
8	0.3668966493	0.242794633	0.445994491	0.2941815785	0.0746215598
9	0.3599219727	0.203153557	0.537334741	0.294688695	0.0774400435
10	0.3552099307	0.227739505	0.599406527	0.2919513705	0.0831723383

Context: Cognition. Repository: 200 Rules. Creation (20% repository match the invariant)

N° of Rules	Mean Value	Min Value	Max Value	Median	Standard Deviati
1	0.493000437	0.194229653	0.662269204	0.437906423	0.1442761587
2	0.7215698632	0.384182234	1.124658938	0.4821586255	0.2804836982
3	0.89390943	0.535602469	1.460006056	0.5642278265	0.3453121503
4	1.1144060685	0.718744541	1.961864422	0.774444116	0.4233013822
5	1.2736615994	0.855651128	2.632494677	0.90284554	0.5315408072
6	1.5476600338	1.075916428	3.329682288	1.1415528435	0.5641124161
7	1.7711780541	1.167117342	3.430375608	1.2851348885	0.6312206109
8	1.883815878	1.276829407	3.548936426	1.4130341925	0.6102862701
9	2.0912952129	1.498454516	3.970955387	1.6118723255	0.6264199794
10	2.2538820474	1.589969856	4.648410879	1.7362095055	0.7003396253

Context: Cognition. Repository: 200 Rules. Identification (20% repository match the invar

N° of Rules	Mean Value	Min Value	Max Value	Median	Standard Deviati
1	0.4887540979	0.2014576	0.636542332	0.4592729955	0.1548249657
2	0.7092809413	0.356284836	1.06916704	0.4855455795	0.2674578825
3	0.9337729463	0.548173194	1.572894778	0.5811183825	0.3673455303
4	1.1315950068	0.721154555	2.003143417	0.760837363	0.4303308282
5	1.3268377375	0.893104912	2.615846191	0.938604448	0.5179861205
6	1.5033131138	1.028818678	3.228302775	1.097885106	0.5616864615
7	1.7697358188	1.133447769	3.439457073	1.258408247	0.6107815332
8	1.926399896	1.308656845	3.568349645	1.436877273	0.674663915
9	2.0692311978	1.447428476	3.999883371	1.592931581	0.6530906873
10	2.2634310856	1.625374706	3.917010116	1.7754813295	0.6098239212

Context: Cognition. Repository: 200 Rules. Modification (20% repository match the invari

N° of Rules	Mean Value	Min Value	Max Value	Median	Standard Deviati
1	0.4933759601	0.194687124	0.535129079	0.439566023	0.1216824069
2	0.698350188	0.365605647	1.057705171	0.509917217	0.2629571967
3	0.8614414032	0.523743699	1.520526385	0.558791449	0.3403207859
4	1.0423809524	0.687601964	1.84955189	0.731347123	0.3984507786
5	1.2613563868	0.857677501	2.598976936	0.8967752825	0.5232929986
6	1.4884484661	1.018212792	3.167342717	1.0786958305	0.5538153785
7	1.7435392936	1.164604103	3.552941335	1.286618222	0.5975217065
8	1.9142920531	1.288220312	3.678983201	1.4270887405	0.6592058528
9	2.0820546617	1.480782507	3.918467699	1.6164722875	0.6354195123
10	2.241365826	1.582167606	4.414964993	1.731210197	0.6742189028

Context: Cognition. Repository: 200 Rules. Rule Removal (20% match the invariant)

N° of Rules	Mean Value	Min Value	Max Value	Median	Standard Deviati
1	0.4775157485	0.197474691	0.577886338	0.4598289385	0.1640673276
2	0.4866119522	0.199508195	0.549281645	0.4621673275	0.1363930284
3	0.4855624074	0.200719936	0.576239338	0.458814142	0.1235116863
4	0.5065300176	0.202474	0.585355161	0.4509712795	0.0915585855
5	0.5152475039	0.250026974	0.671464019	0.4466967145	0.1167375811
6	0.5164980958	0.200341747	0.685821429	0.451202525	0.1263616138
7	0.5110823102	0.286281726	0.598695889	0.4031702625	0.1106057854
8	0.4954278793	0.196483015	0.600946667	0.479686522	0.1423706519
9	0.5093641516	0.205355994	0.595676655	0.4545761695	0.1287474103
10	0.4954993488	0.210547652	0.584690483	0.4299725565	0.1237188962

Context: Cognition. Repository: 300 Rules. Creation (20% repository match the invariant)

N° of Rules	Mean Value	Min Value	Max Value	Median	Standard Deviati
1	0.6050134166	0.268657211	0.981987867	0.5843640095	0.2065361819
2	0.8820481632	0.499575397	1.716110828	0.5507695475	0.3826035701
3	1.1279656081	0.734902255	2.363848295	0.777519609	0.4616778032
4	1.4305802878	1.000950958	2.681015091	1.0431322325	0.5317048349
5	1.7179968092	1.162035132	3.717941154	1.265218258	0.6107485906
6	1.9901778432	1.364590289	3.829901627	1.5445306595	0.6232577346
7	2.2517296849	1.595165837	4.287347018	1.7929115855	0.6347832289
8	2.393373345	1.76612082	4.937435689	1.928235604	0.6903842116
9	2.6225411442	1.946445283	5.185314557	2.1405213615	0.6993322054
10	2.9718517606	2.324005032	4.965764849	2.523122984	0.6473875707

Context: Cognition. Repository: 300 Rules. Identification (20% repository match the invar

N° of Rules	Mean Value	Min Value	Max Value	Median	Standard Deviati
1	0.5941330846	0.275402059	0.91808407	0.6072887115	0.2153143547
2	0.8830443664	0.502933582	1.592697992	0.556215633	0.3724843703
3	1.1507227445	0.736999724	2.249546538	0.7757792045	0.4512216515
4	1.4066388114	0.978876784	3.011217837	1.02544161	0.5422770453
5	1.6837238917	1.197610745	3.264106001	1.25735179	0.5618066374
6	2.0330395574	1.422529919	4.028986092	1.599017793	0.6066535834
7	2.1408927678	1.492679761	4.358829913	1.6883008335	0.6436048618
8	2.4817463497	1.830291823	4.806014994	2.018842935	0.675376756
9	2.7365785061	2.07343895	5.552044929	2.265278255	0.7312773573
10	2.8714098163	2.206173843	5.203128764	2.4076772605	0.7032396479

Context: Cognition. Repository: 300 Rules. Modification (20% repository match the invari

N° of Rules	Mean Value	Min Value	Max Value	Median	Standard Deviati
1	0.6938537825	0.272934428	1.182324926	0.597125897	0.2807150206
2	1.0026855788	0.497146319	2.015884903	0.696944557	0.4386267831
3	1.26303464	0.737126139	2.497278671	0.9544093905	0.5286232689
4	1.5827097617	0.978795138	2.968610252	1.1997355945	0.5776771231
5	1.8745633367	1.237428309	3.667706064	1.45189584	0.6016296778
6	2.182425349	1.567188886	4.066121886	1.7382606405	0.6312939283
7	2.4104795753	1.756781431	4.446477378	1.9488035805	0.6564696527
8	2.6812664825	2.050230756	4.878407304	2.2360036885	0.6356438268
9	2.8637305573	2.206195777	5.615209859	2.3868538385	0.7194331484
10	3.1150685901	2.433128616	5.066745641	2.6384037745	0.6572296026

Context: Cognition. Repository: 300 Rules. Rule Removal (20% match the invariant)

N° of Rules	Mean Value	Min Value	Max Value	Median	Standard Deviati
1	0.7173025429	0.268082485	1.039663769	0.6403415205	0.2536520762
2	0.6835889093	0.275238005	1.164463305	0.5291021615	0.2714328884
3	0.7497515736	0.34255536	1.102053479	0.680126258	0.2230465012
4	0.7671489667	0.267938477	1.010252972	0.704222753	0.2254737769
5	0.6934640634	0.271175547	1.081118348	0.5439435435	0.2742697851
6	0.690553298	0.269125214	1.167590304	0.5959378985	0.2772430048
7	0.70395933	0.269698336	1.115496183	0.605509017	0.2810093601
8	0.7255000353	0.270543689	1.099522019	0.6581615935	0.2628604462
9	0.7000210553	0.274843539	1.068998226	0.6165860195	0.2457437093
10	0.7882525658	0.280052581	1.143430108	0.7073391125	0.2369039246

Context: Preferences. Repository: Empty Rule Creation

N° of Rules	Mean Value	Min Value	Max Value	Median	Standard Deviat
1	0.1816587525	0.09369753	0.298743369	0.147027461	0.0519366813
2	0.3103521692	0.187225131	0.628291206	0.2538641975	0.0825098612
3	0.4050161947	0.145043867	0.781838316	0.373588573	0.1461253709
4	0.4664824461	0.17246837	0.848731736	0.408544342	0.1773033862
5	0.5755904222	0.222964107	1.221851328	0.4743313045	0.2390706068
6	0.6053042238	0.255758361	1.484680488	0.403837806	0.3192698976
7	0.7036704595	0.295150034	1.607724911	0.586669851	0.2702754439
8	0.805211221	0.33709538	1.54106091	0.6302997745	0.3105883055
9	0.8543787862	0.362891539	1.764407054	0.7049917445	0.3953100441
10	0.8927485902	0.421241136	2.233111201	0.6795153915	0.4235154069

Context: Preferences. Repository: 100 Rules. Creation (20% repository match the invaria

N° of Rules	Mean Value	Min Value	Max Value	Median	Standard Deviat
1	0.5334425327	0.264620733	0.751060105	0.440533986	0.157425858
2	0.8209653944	0.274656155	1.326058506	0.7353017105	0.2078870201
3	0.9193701311	0.351204994	1.581376794	0.8527284105	0.3683551089
4	1.1690448481	0.475694853	2.179033869	1.001704408	0.4866643387
5	1.2782712427	0.601815707	2.592441818	0.9204236115	0.5421689046
6	1.378427005	0.667799026	2.816700014	0.9675085075	0.5839187967
7	1.5125730117	0.811404792	3.045638557	1.076457871	0.5785188446
8	1.6337241124	0.88951661	3.407612239	1.161262058	0.6324561374
9	1.8360553835	1.010826797	3.656433615	1.3888273785	0.7398644678
10	2.0629274786	1.053264315	4.201409624	1.495009911	0.800561507

Context: Preferences. Repository: 100 Rules. Identification (20% repository match the in

N° of Rules	Mean Value	Min Value	Max Value	Median	Standard Deviat
1	0.5722154701	0.259024067	1.851345902	0.4195987835	0.3082089231
2	0.8006374113	0.46205415	1.040126123	0.735392956	0.1453958697
3	0.9712747571	0.358113207	1.793387276	0.863977117	0.4204868627
4	1.1121752259	0.445375279	2.118160003	0.931739539	0.4581825485
5	1.2483394921	0.581476809	2.730369134	0.877795704	0.53115666
6	1.3710119022	0.678317403	2.412688445	0.967728054	0.5237179499
7	1.4950477667	0.808283863	2.422682733	1.0582976035	0.5197659866
8	1.6033181826	0.852829608	2.995372949	1.193303283	0.6011152684
9	1.7531869252	0.960599598	2.953954758	1.2603394995	0.6335600005
10	1.9252648866	1.08067213	3.78301445	1.3502496805	0.7129686486

Context: Preferences. Repository: 100 Rules. Modification (20% repository match the inv

N° of Rules	Mean Value	Min Value	Max Value	Median	Standard Deviat
1	0.5977830258	0.26939578	1.66184806	0.441287925	0.2967119497
2	0.8081953157	0.248561078	1.295399645	0.725931489	0.2079526036
3	0.8934042668	0.358204976	1.35752703	0.7483864115	0.334335089
4	1.0558356522	0.4696945	2.134365686	0.7852780365	0.4501019125
5	1.2297117745	0.562728473	2.170304966	0.953550587	0.5079547136
6	1.3708244297	0.670430927	2.92087597	1.010337522	0.5876676326
7	1.5064856365	0.780328144	3.192292537	1.0737155175	0.6354997865
8	1.6181070827	0.884678976	2.832843262	1.1758473525	0.5649574289
9	1.816327288	1.007755319	3.430787672	1.3914938125	0.6492629333
10	1.9582596149	1.079590917	3.117986653	1.5327403005	0.6123073315

Context: Preferences. Repository: 100 Rules. Removal (20% repository match the invaria

N° of Rules	Mean Value	Min Value	Max Value	Median	Standard Deviat
1	0.4740844276	0.244917876	0.642004125	0.442102067	0.1119524013
2	0.4830972241	0.204712624	0.745787506	0.398700141	0.1394225326
3	0.504796965	0.262397822	0.659100576	0.41447666	0.1239283065
4	0.4841153969	0.263959407	0.68374945	0.409993302	0.1227271035
5	0.4916191454	0.246755609	0.687137099	0.4020458195	0.1411008006
6	0.4850178649	0.26601134	0.720722447	0.399325154	0.1152976142
7	0.5804609716	0.245151843	1.70884992	0.4224207235	0.3110427475
8	0.4875115984	0.254764452	0.642111825	0.3939572535	0.1378857666
9	0.4998156678	0.276312021	0.678430833	0.398463099	0.1310278173
10	0.5115912048	0.276206213	0.699074096	0.4326365795	0.1264685607

Context: Preferences. Repository: 200 Rules. Creation (20% repository match the invaria

N° of Rules	Mean Value	Min Value	Max Value	Median	Standard Deviat
1	0.5943844283	0.20759979	0.859898037	0.498742088	0.2092777346
2	0.8976251462	0.376260416	1.622493476	0.6923214165	0.3987699758
3	1.1136332371	0.578208933	2.313374904	0.786901915	0.5071897886
4	1.3555407526	0.708567409	3.538018349	0.948444455	0.6551003791
5	1.5562056105	0.869893929	2.867693012	1.141659842	0.5783522999
6	1.8535445428	1.074842761	3.702902863	1.376553076	0.705185801
7	2.0790695212	1.303821389	4.164072819	1.6275795955	0.7346923117
8	2.3005381732	1.602118191	4.75293115	1.775430766	0.7254193494
9	2.3913371972	1.666564442	4.771689058	1.839478924	0.7880917623
10	2.4753870611	1.824265505	3.90737244	1.9990898625	0.6122715018

Context: Preferences. Repository: 200 Rules. Identification (20% repository match the in

N° of Rules	Mean Value	Min Value	Max Value	Median	Standard Deviat
1	0.6771485052	0.283457211	0.853511401	0.574165472	0.157727336
2	0.8614752615	0.400932552	1.564009712	0.584156898	0.3803101236
3	1.100268638	0.568496663	1.900264834	0.7634793165	0.4643720022
4	1.3303801533	0.719463349	2.412258848	0.9683612	0.523798634
5	1.5574142398	0.891713532	3.235125574	1.1921213945	0.6102012568
6	1.8590181265	1.112249819	3.951335776	1.375243318	0.6971785128
7	2.026268068	1.206674465	3.994738648	1.600238097	0.7000938723
8	2.2594385201	1.541561328	4.258956118	1.711295618	0.7325899156
9	2.4132075368	1.693894315	4.812621772	1.8483643455	0.804116003
10	2.5203069245	1.83484967	4.143442401	2.03052183	0.6246149184

Context: Preferences. Repository: 200 Rules. Modification (20% repository match the inv

N° of Rules	Mean Value	Min Value	Max Value	Median	Standard Deviat
1	0.6937722073	0.316281635	0.986461339	0.6400912065	0.1670590087
2	0.9447159726	0.378561551	1.924414832	0.8001543705	0.412245231
3	1.1129374204	0.572742523	2.377524892	0.7625944615	0.5027868373
4	1.3588438865	0.725311719	2.508647923	0.9801202725	0.5206192424
5	1.5791749082	0.885802445	3.495332406	1.1478041315	0.6260632378
6	1.8268013811	1.025292741	3.516511906	1.359292963	0.6490087109
7	2.0581670603	1.184059363	3.924990604	1.5995666775	0.7508296039
8	2.1315356672	1.468628374	3.46852677	1.6961732715	0.5718632398
9	2.3890742715	1.665651477	4.822953114	1.8429293285	0.7603980457
10	2.5330477592	1.83305629	5.318955643	1.9991289385	0.7476003624

Context: Preferences. Repository: 200 Rules. Rule Removal (20% match the invariant)

N° of Rules	Mean Value	Min Value	Max Value	Median	Standard Deviat
1	0.6350495569	0.205513636	0.829879272	0.561590548	0.1959090221
2	0.7127963599	0.331139315	0.954038151	0.6167558065	0.1637640535
3	0.6413928247	0.312070338	0.89829021	0.5848780785	0.1681134137
4	0.6381658129	0.211256966	0.823601384	0.5879219015	0.2033940796
5	0.6198292251	0.215965102	0.90515002	0.5951447435	0.1965673899
6	0.7278763604	0.336921681	1.023117397	0.658668751	0.1844209523
7	0.7311734614	0.440104783	0.939370639	0.6419461945	0.1537524146
8	0.6808509356	0.391624735	0.788695407	0.6364007585	0.1508360413
9	0.6780414201	0.241708385	0.858922566	0.6004750665	0.1630320526
10	0.6819026313	0.334838459	0.840908469	0.594012935	0.1616967445

Context: Preferences. Repository: 300 Rules. Creation (20% repository match the invaria

N° of Rules	Mean Value	Min Value	Max Value	Median	Standard Deviat
1	0.8519341419	0.336128219	1.7127516	0.708956461	0.2754489471
2	1.1182829188	0.531920017	2.252017733	0.74600478	0.5052969035
3	1.3615994168	0.748053789	2.93495158	0.981517446	0.5820166933
4	1.7289670935	1.023228513	3.62318937	1.26390175	0.6851251094
5	2.0895614593	1.28288206	3.884395073	1.663388863	0.7099965054
6	2.2300442483	1.593977139	3.691403631	1.7448593135	0.5902262455
7	2.5733930528	1.821278952	5.196628426	1.970280019	0.8289032822
8	2.7406649293	2.025229602	5.486480512	2.1807744905	0.7884407906
9	2.9237300864	2.238869133	4.917541745	2.3576925545	0.6895747524
10	3.1915838782	2.453102174	5.196355066	2.559777121	0.7289127796

Context: Preferences. Repository: 300 Rules. Identification (20% repository match the in

N° of Rules	Mean Value	Min Value	Max Value	Median	Standard Deviat
1	0.8277356837	0.309931384	1.818706413	0.696251316	0.2929729355
2	1.054635645	0.524123843	1.960657346	0.731884406	0.4627417623
3	1.3601690944	0.752008687	2.762017781	0.968527633	0.6039140282
4	1.6653669394	0.979020798	3.48031178	1.2527080435	0.6235864935
5	2.0487655057	1.237508832	3.800469388	1.6165448505	0.7207057311
6	2.3038803613	1.590004496	4.801770465	1.73032241	0.7580459859
7	2.4790691607	1.781976391	4.457281892	1.964527787	0.7294347176
8	2.7604517185	2.078951026	4.66846231	2.2533729675	0.6848867095
9	2.8884844104	2.203959866	5.189170209	2.387879106	0.7086571121
10	3.1765755504	2.425721526	5.299921801	2.529210454	0.7849556781

Context: Preferences. Repository: 300 Rules. Modification (20% repository match the inv

N° of Rules	Mean Value	Min Value	Max Value	Median	Standard Deviat
1	0.7402814672	0.280451373	1.196364216	0.662925573	0.2812736447
2	1.0187756489	0.52807951	1.937274281	0.7371953825	0.4609574376
3	1.353311605	0.744169493	2.681148905	0.9724088855	0.5778156089
4	1.7251792528	1.001458837	3.421152928	1.213770586	0.6185540065
5	2.0301497235	1.279274059	4.017189824	1.567964057	0.7442259133
6	2.2947794429	1.585821772	4.764616783	1.752685557	0.7536378912
7	2.5160342994	1.790920758	4.938072855	1.9670742695	0.7409600088
8	2.7602038405	2.072272322	4.833444181	2.2629609185	0.7125266805
9	2.8369851347	2.20558494	3.857932633	2.3305983245	0.5768777424
10	3.2150982119	2.469291517	5.315999611	2.573596475	0.7542986791

Context: Preferences. Repository: 300 Rules. Rule Removal (20% match the invariant)

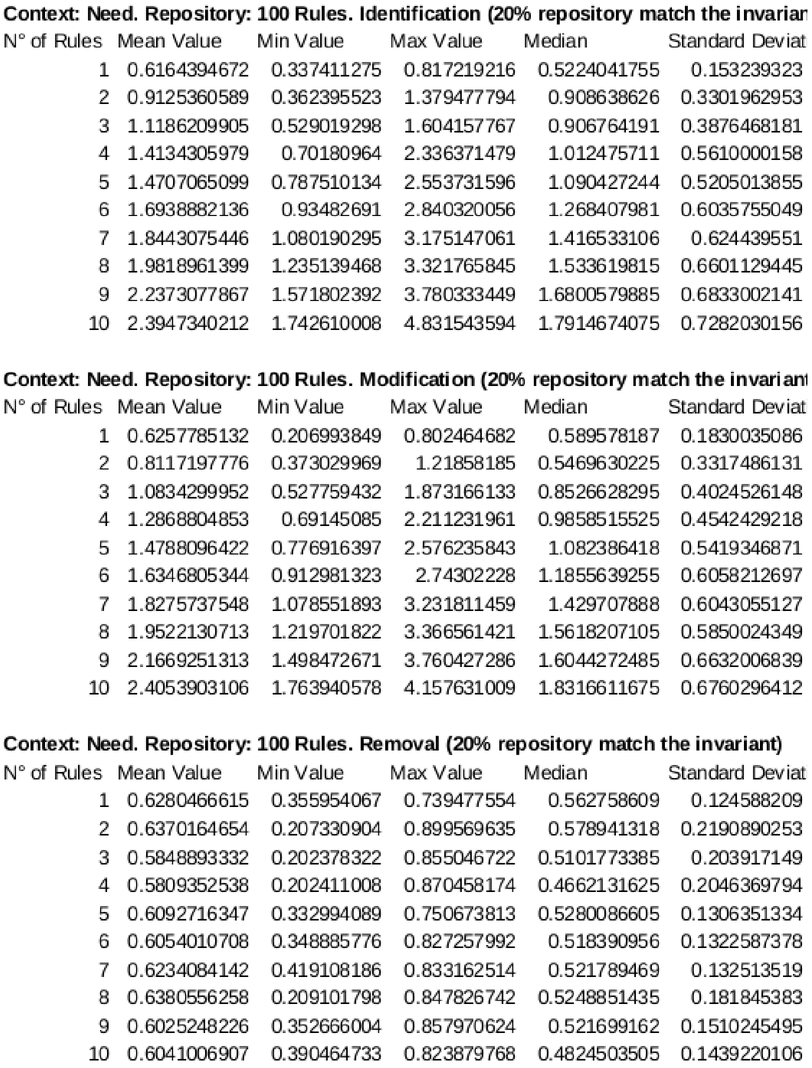
N° of Rules	Mean Value	Min Value	Max Value	Median	Standard Deviat
1	0.7568484724	0.283873947	1.846976561	0.717617859	0.3645882404
2	0.7931839964	0.2807241	1.690762975	0.7306019455	0.3319438323
3	0.7409690233	0.284472419	1.206861853	0.707436783	0.2808646999
4	0.7499576306	0.282684135	1.138837768	0.6582060145	0.2741547124
5	0.781703958	0.284293414	1.725714004	0.720941993	0.2944332407
6	0.7029995946	0.282190563	1.11213221	0.63096194	0.2679078926
7	0.6983763256	0.277813121	1.066514742	0.61427026	0.2476783925
8	0.7824814664	0.288742654	1.087039633	0.7530623835	0.2702650912
9	0.7262139482	0.27913536	1.165487806	0.683497181	0.2770457854
10	0.7433092648	0.285049101	1.145797732	0.6062157075	0.2861963826

Context: Need. Repository: Empty Rule Creation

N° of Rules	Mean Value	Min Value	Max Value	Median	Standard Deviat
1	0.3091107771	0.168080172	0.536732015	0.2523525865	0.0892971468
2	0.4572539919	0.217998812	0.953853135	0.387624608	0.1832480582
3	0.6024626599	0.304411196	1.015769955	0.5059074545	0.2061979106
4	0.7296877088	0.403768031	1.375088622	0.6327109755	0.2329472266
5	0.8701897565	0.495591234	1.7895826	0.635143408	0.3393167226
6	0.9979481164	0.543497811	2.12738379	0.799034848	0.3746171531
7	1.0707279233	0.61016104	2.237075468	0.811051255	0.4122348168
8	1.1750331178	0.684247422	2.924101386	0.8410673605	0.4881254234
9	1.3005352097	0.793243678	2.367459854	0.964677319	0.4664849826
10	1.4253370749	0.918629461	2.76816032	1.0660231655	0.4950237331

Context: Need. Repository: 100 Rules. Creation (20% repository match the invariant)

N° of Rules	Mean Value	Min Value	Max Value	Median	Standard Deviat
1	0.5747518077	0.203922231	0.846344778	0.4679870945	0.1981768604
2	0.8502237923	0.364164958	1.359679741	0.7562511245	0.3129389181
3	1.0209391276	0.520641273	1.82143057	0.7332688405	0.3861395443
4	1.3781814039	0.690101582	2.31276813	1.1137158925	0.4991706196
5	1.4560713945	0.764462858	2.396322396	1.1161352625	0.4899989691
6	1.6081298954	0.926129364	2.763280163	1.23284864	0.5661851136
7	1.8145471754	1.105029416	3.157873496	1.5054763605	0.5441629416
8	1.9852645536	1.254206769	3.604864907	1.549746235	0.6070030693
9	2.2108941643	1.544504227	3.5697918	1.6267291735	0.6449250846
10	2.4261220093	1.716997114	4.648058247	1.7915924925	0.7512781429



Context: Need. Repository: 100 Rules. Identification (20% repository match the invarian

N° of Rules	Mean Value	Min Value	Max Value	Median	Standard Deviat
1	0.6164394672	0.337411275	0.817219216	0.5224041755	0.153239323
2	0.9125360589	0.362395523	1.379477794	0.908638626	0.3301962953
3	1.1186209905	0.529019298	1.604157767	0.906764191	0.3876468181
4	1.4134305979	0.70180964	2.336371479	1.012475711	0.5610000158
5	1.4707065099	0.787510134	2.553731596	1.090427244	0.5205013855
6	1.6938882136	0.93482691	2.840320056	1.268407981	0.6035755049
7	1.8443075446	1.080190295	3.175147061	1.416533106	0.624439551
8	1.9818961399	1.235139468	3.321765845	1.533619815	0.6601129445
9	2.2373077867	1.571802392	3.780333449	1.6800579885	0.6833002141
10	2.3947340212	1.742610008	4.831543594	1.7914674075	0.7282030156

Context: Need. Repository: 100 Rules. Modification (20% repository match the invariant

N° of Rules	Mean Value	Min Value	Max Value	Median	Standard Deviat
1	0.6257785132	0.206993849	0.802464682	0.589578187	0.1830035086
2	0.8117197776	0.373029969	1.21858185	0.5469630225	0.3317486131
3	1.0834299952	0.527759432	1.873166133	0.8526628295	0.4024526148
4	1.2868804853	0.69145085	2.211231961	0.9858515525	0.4542429218
5	1.4788096422	0.776916397	2.576235843	1.082386418	0.5419346871
6	1.6346805344	0.912981323	2.74302228	1.1855639255	0.6058212697
7	1.8275737548	1.078551893	3.231811459	1.429707888	0.6043055127
8	1.9522130713	1.219701822	3.366561421	1.5618207105	0.5850024349
9	2.1669251313	1.498472671	3.760427286	1.6044272485	0.6632006839
10	2.4053903106	1.763940578	4.157631009	1.8316611675	0.6760296412

Context: Need. Repository: 100 Rules. Removal (20% repository match the invariant)

N° of Rules	Mean Value	Min Value	Max Value	Median	Standard Deviat
1	0.6280466615	0.355954067	0.739477554	0.562758609	0.124588209
2	0.6370164654	0.207330904	0.899569635	0.578941318	0.2190890253
3	0.5848893332	0.202378322	0.855046722	0.5101773385	0.203917149
4	0.5809352538	0.202411008	0.870458174	0.4662131625	0.2046369794
5	0.6092716347	0.332994089	0.750673813	0.5280086605	0.1306351334
6	0.6054010708	0.348885776	0.827257992	0.518390956	0.1322587378
7	0.6234084142	0.419108186	0.833162514	0.521789469	0.132513519
8	0.6380556258	0.209101798	0.847826742	0.5248851435	0.181845383
9	0.6025248226	0.352666004	0.857970624	0.521699162	0.1510245495
10	0.6041006907	0.390464733	0.823879768	0.4824503505	0.1439220106

Context: Need. Repository: 200 Rules. Creation (20% repository match the invariant)

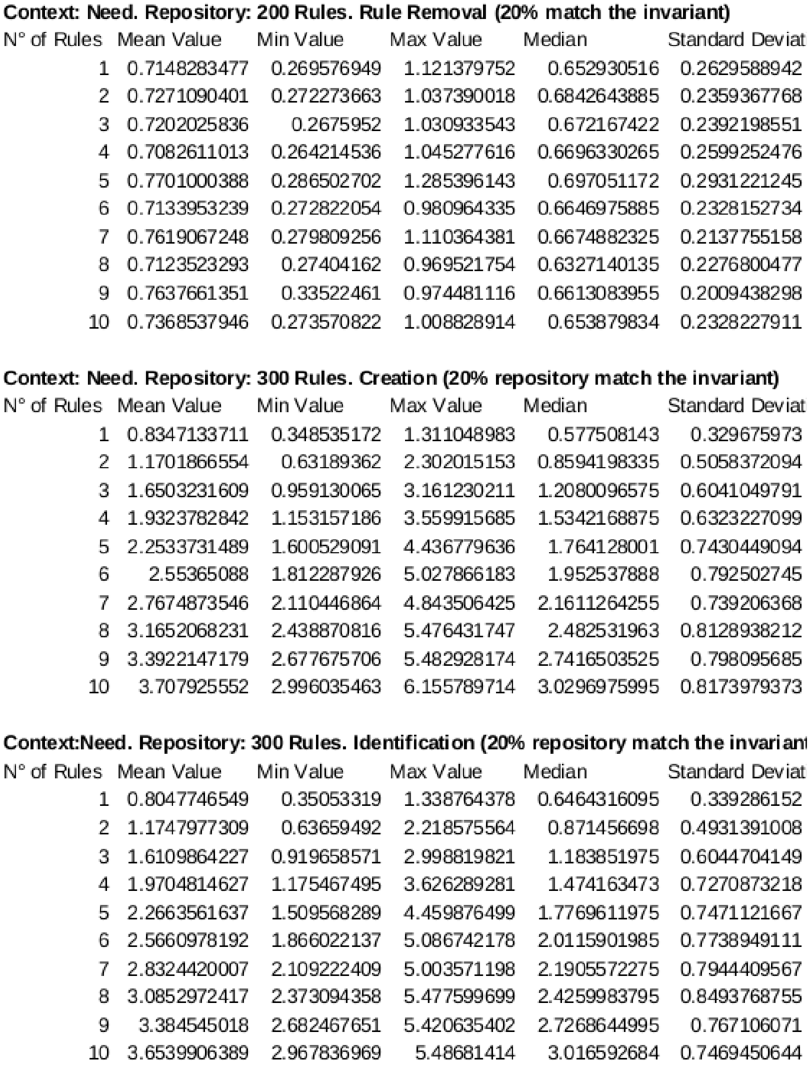
N° of Rules	Mean Value	Min Value	Max Value	Median	Standard Deviat
1	0.7607870146	0.279141151	1.12833615	0.6615878305	0.2628195622
2	0.9883285435	0.492420024	1.849603918	0.692075405	0.4455463944
3	1.3145690407	0.729473278	2.204270877	0.95615235	0.4957471701
4	1.5738365563	0.86926934	2.66028159	1.1626072355	0.5728037311
5	1.8842114937	1.085180386	3.495342594	1.4152520725	0.6897278063
6	2.093961572	1.321447938	4.043205909	1.607582639	0.7430785047
7	2.3445274803	1.63864392	4.38689899	1.7597715805	0.7613964987
8	2.5001335807	1.838105461	4.439023847	1.9180877325	0.7406436468
9	2.7664540154	2.084545938	5.278681527	2.1427306505	0.8225121996
10	2.9614548627	2.290488332	5.239844957	2.3464168155	0.7635095757

Context: Need. Repository: 200 Rules. Identification (20% repository match the invarian

N° of Rules	Mean Value	Min Value	Max Value	Median	Standard Deviat
1	0.682489325	0.263207562	1.164337938	0.60154741	0.2764851194
2	1.0052251967	0.486442516	1.655506616	0.7325893175	0.402678411
3	1.3064510729	0.705053343	2.17078869	0.9540620355	0.4835255096
4	1.5534644027	0.854229297	2.549332283	1.165028104	0.5374416208
5	1.8581599173	1.074579735	3.418706608	1.4454427165	0.6524805069
6	2.1213559248	1.463793333	4.397642115	1.639655611	0.7232730331
7	2.2872782801	1.65935766	3.924161934	1.77981334	0.6521029184
8	2.5570837885	1.856534061	5.457670349	1.8926616325	0.8786126044
9	2.7383070741	2.080631273	4.510690289	2.129816229	0.731477461
10	2.9961611832	2.328254388	5.40127212	2.36402251	0.7947498388

Context: Need. Repository: 200 Rules. Modification (20% repository match the invariant

N° of Rules	Mean Value	Min Value	Max Value	Median	Standard Deviat
1	0.7072118626	0.28284728	1.046688687	0.6734217385	0.2664527026
2	0.965770309	0.495606462	1.906082328	0.692212675	0.4117863556
3	1.2969449741	0.711575411	2.074028619	0.996695007	0.4847403996
4	1.5758090438	0.98193164	2.782508584	1.163186709	0.5326386862
5	1.8800559839	1.094020549	3.637879651	1.44685637	0.7015681921
6	2.1317137512	1.351452093	4.186076806	1.6121069945	0.7935183578
7	2.3293749084	1.648315334	4.418496724	1.7678070305	0.7560803931
8	2.4747557254	1.843358986	4.013691475	1.9153219115	0.6875700602
9	2.7546789209	2.108216554	5.285213712	2.1558942665	0.7760505031
10	3.2822231958	2.318144996	7.097728696	2.4141464475	1.0979800017



Context: Need. Repository: 200 Rules. Rule Removal (20% match the invariant)

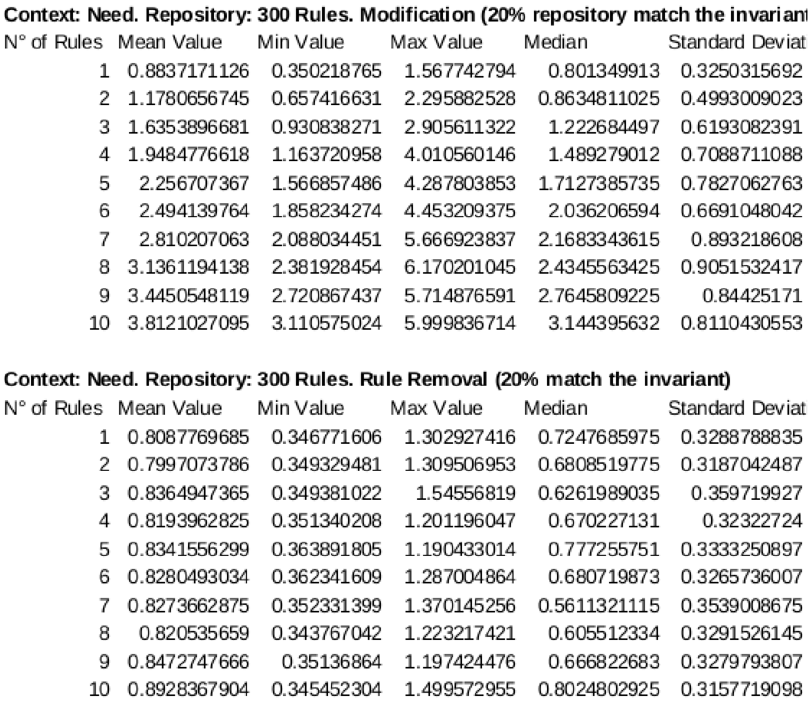
N° of Rules	Mean Value	Min Value	Max Value	Median	Standard Deviat
1	0.7148283477	0.269576949	1.121379752	0.652930516	0.2629588942
2	0.7271090401	0.272273663	1.037390018	0.6842643885	0.2359367768
3	0.7202025836	0.2675952	1.030933543	0.672167422	0.2392198551
4	0.7082611013	0.264214536	1.045277616	0.6696330265	0.2599252476
5	0.7701000388	0.286502702	1.285396143	0.697051172	0.2931221245
6	0.7133953239	0.272822054	0.980964335	0.6646975885	0.2328152734
7	0.7619067248	0.279809256	1.110364381	0.6674882325	0.2137755158
8	0.7123523293	0.27404162	0.969521754	0.6327140135	0.2276800477
9	0.7637661351	0.33522461	0.974481116	0.6613083955	0.2009438298
10	0.7368537946	0.273570822	1.008828914	0.653879834	0.2328227911

Context: Need. Repository: 300 Rules. Creation (20% repository match the invariant)

N° of Rules	Mean Value	Min Value	Max Value	Median	Standard Deviat
1	0.8347133711	0.348535172	1.311048983	0.577508143	0.329675973
2	1.1701866554	0.63189362	2.302015153	0.8594198335	0.5058372094
3	1.6503231609	0.959130065	3.161230211	1.2080096575	0.6041049791
4	1.9323782842	1.153157186	3.559915685	1.5342168875	0.6323227099
5	2.2533731489	1.600529091	4.436779636	1.764128001	0.7430449094
6	2.55365088	1.812287926	5.027866183	1.952537888	0.792502745
7	2.7674873546	2.110446864	4.843506425	2.1611264255	0.739206368
8	3.1652068231	2.438870816	5.476431747	2.482531963	0.8128938212
9	3.3922147179	2.677675706	5.482928174	2.7416503525	0.798095685
10	3.707925552	2.996035463	6.155789714	3.0296975995	0.8173979373

Context:Need. Repository: 300 Rules. Identification (20% repository match the invariant

N° of Rules	Mean Value	Min Value	Max Value	Median	Standard Deviat
1	0.8047746549	0.35053319	1.338764378	0.6464316095	0.339286152
2	1.1747977309	0.63659492	2.218575564	0.871456698	0.4931391008
3	1.6109864227	0.919658571	2.998819821	1.183851975	0.6044704149
4	1.9704814627	1.175467495	3.626289281	1.474163473	0.7270873218
5	2.2663561637	1.509568289	4.459876499	1.7769611975	0.7471121667
6	2.5660978192	1.866022137	5.086742178	2.0115901985	0.7738949111
7	2.8324420007	2.109222409	5.003571198	2.1905572275	0.7944409567
8	3.0852972417	2.373094358	5.477599699	2.4259983795	0.8493768755
9	3.384545018	2.682467651	5.420635402	2.7268644995	0.767106071
10	3.6539906389	2.967836969	5.48681414	3.016592684	0.7469450644



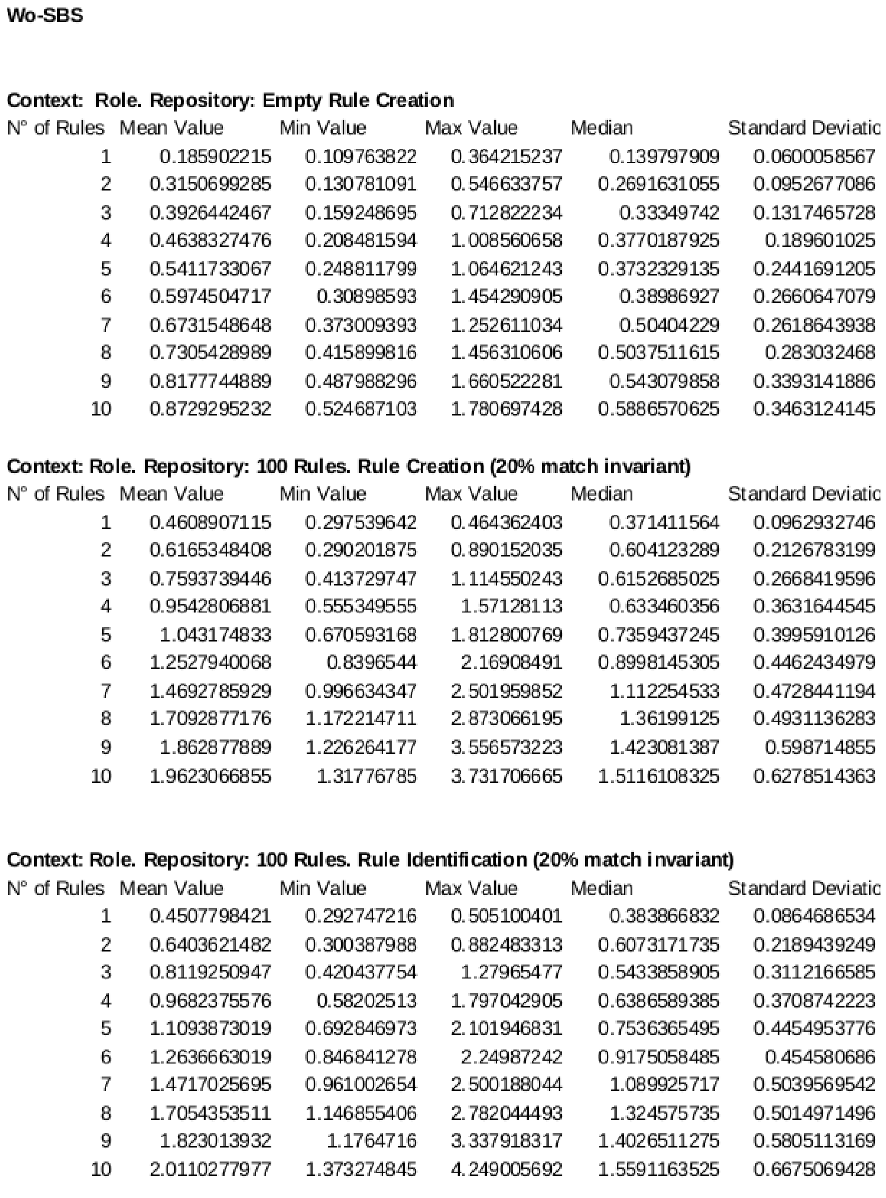
Context: Need. Repository: 300 Rules. Modification (20% repository match the invariant

N° of Rules	Mean Value	Min Value	Max Value	Median	Standard Deviat
1	0.8837171126	0.350218765	1.567742794	0.801349913	0.3250315692
2	1.1780656745	0.657416631	2.295882528	0.8634811025	0.4993009023
3	1.6353896681	0.930838271	2.905611322	1.222684497	0.6193082391
4	1.9484776618	1.163720958	4.010560146	1.489279012	0.7088711088
5	2.256707367	1.566857486	4.287803853	1.7127385735	0.7827062763
6	2.494139764	1.858234274	4.453209375	2.036206594	0.6691048042
7	2.810207063	2.088034451	5.666923837	2.1683343615	0.893218608
8	3.1361194138	2.381928454	6.170201045	2.4345563425	0.9051532417
9	3.4450548119	2.720867437	5.714876591	2.7645809225	0.84425171
10	3.8121027095	3.110575024	5.999836714	3.144395632	0.8110430553

Context: Need. Repository: 300 Rules. Rule Removal (20% match the invariant)

N° of Rules	Mean Value	Min Value	Max Value	Median	Standard Deviat
1	0.8087769685	0.346771606	1.302927416	0.7247685975	0.3288788835
2	0.7997073786	0.349329481	1.309506953	0.6808519775	0.3187042487
3	0.8364947365	0.349381022	1.54556819	0.6261989035	0.359719927
4	0.8193962825	0.351340208	1.201196047	0.670227131	0.32322724
5	0.8341556299	0.363891805	1.190433014	0.777255751	0.3333250897
6	0.8280493034	0.362341609	1.287004864	0.680719873	0.3265736007
7	0.8273662875	0.352331399	1.370145256	0.5611321115	0.3539008675
8	0.820535659	0.343767042	1.223217421	0.605512334	0.3291526145
9	0.8472747666	0.35136864	1.197424476	0.666822683	0.3279793807
10	0.8928367904	0.345452304	1.499572955	0.8024802925	0.3157719098

Wo-SBS Results



Wo-SBS

Context: Role. Repository: Empty Rule Creation

N° of Rules	Mean Value	Min Value	Max Value	Median	Standard Deviatic
1	0.185902215	0.109763822	0.364215237	0.139797909	0.0600058567
2	0.3150699285	0.130781091	0.546633757	0.2691631055	0.0952677086
3	0.3926442467	0.159248695	0.712822234	0.33349742	0.1317465728
4	0.4638327476	0.208481594	1.008560658	0.3770187925	0.189601025
5	0.5411733067	0.248811799	1.064621243	0.3732329135	0.2441691205
6	0.5974504717	0.30898593	1.454290905	0.38986927	0.2660647079
7	0.6731548648	0.373009393	1.252611034	0.50404229	0.2618643938
8	0.7305428989	0.415899816	1.456310606	0.5037511615	0.283032468
9	0.8177744889	0.487988296	1.660522281	0.543079858	0.3393141886
10	0.8729295232	0.524687103	1.780697428	0.5886570625	0.3463124145

Context: Role. Repository: 100 Rules. Rule Creation (20% match invariant)

N° of Rules	Mean Value	Min Value	Max Value	Median	Standard Deviatic
1	0.4608907115	0.297539642	0.464362403	0.371411564	0.0962932746
2	0.6165348408	0.290201875	0.890152035	0.604123289	0.2126783199
3	0.7593739446	0.413729747	1.114550243	0.6152685025	0.2668419596
4	0.9542806881	0.555349555	1.57128113	0.633460356	0.3631644545
5	1.043174833	0.670593168	1.812800769	0.7359437245	0.3995910126
6	1.2527940068	0.8396544	2.16908491	0.8998145305	0.4462434979
7	1.4692785929	0.996634347	2.501959852	1.112254533	0.4728441194
8	1.7092877176	1.172214711	2.873066195	1.36199125	0.4931136283
9	1.862877889	1.226264177	3.556573223	1.423081387	0.598714855
10	1.9623066855	1.31776785	3.731706665	1.5116108325	0.6278514363

Context: Role. Repository: 100 Rules. Rule Identification (20% match invariant)

N° of Rules	Mean Value	Min Value	Max Value	Median	Standard Deviatic
1	0.4507798421	0.292747216	0.505100401	0.383866832	0.0864686534
2	0.6403621482	0.300387988	0.882483313	0.6073171735	0.2189439249
3	0.8119250947	0.420437754	1.27965477	0.5433858905	0.3112166585
4	0.9682375576	0.58202513	1.797042905	0.6386589385	0.3708742223
5	1.1093873019	0.692846973	2.101946831	0.7536365495	0.4454953776
6	1.2636663019	0.846841278	2.24987242	0.9175058485	0.454580686
7	1.4717025695	0.961002654	2.500188044	1.089925717	0.5039569542
8	1.7054353511	1.146855406	2.782044493	1.324575735	0.5014971496
9	1.823013932	1.1764716	3.337918317	1.4026511275	0.5805113169
10	2.0110277977	1.373274845	4.249005692	1.5591163525	0.6675069428

Context: Role. Repository: 100 Rules. Rule Modification (20% match invariant)

N° of Rules	Mean Value	Min Value	Max Value	Median	Standard Deviatic
1	0.4347062148	0.250099754	0.452681936	0.3590493595	0.0859879276
2	0.6401433823	0.285771604	0.817154192	0.646009033	0.1880232737
3	0.813760822	0.437080652	1.26765918	0.527612519	0.2972759482
4	0.939896226	0.54876329	1.673312784	0.6200435805	0.3663390415
5	1.0868843869	0.720007551	1.667534119	0.769634359	0.3733766184
6	1.2609595954	0.846782472	2.265727368	0.9068692665	0.4223054817
7	1.4463016541	0.950360512	2.321012339	1.0443404515	0.4626148382
8	1.6365658287	1.099380369	2.521154054	1.2763105225	0.4758819178
9	1.8615251725	1.251711057	2.867514367	1.432816212	0.5266898035
10	1.9236256301	1.31243184	3.309626313	1.4987774635	0.5245419836

Context: Role. Repository: 100 Rules. Rule Removal (20% match invariant)

N° of Rules	Mean Value	Min Value	Max Value	Median	Standard Deviatic
1	0.4434817744	0.296056851	0.475922674	0.375269822	0.0818251519
2	0.4301687153	0.278997975	0.493310914	0.362213764	0.0862802178
3	0.4331028806	0.263321545	0.49371795	0.354763235	0.0839827606
4	0.4364112063	0.29077399	0.579860119	0.374477301	0.0813518147
5	0.4301411818	0.282997091	0.546307604	0.363980609	0.0821162266
6	0.4357113128	0.278530044	0.499461211	0.3760534415	0.0824882585
7	0.4379944549	0.279436588	0.490975072	0.358133844	0.083530646
8	0.4315873874	0.285494126	0.580213447	0.347573353	0.0885030254
9	0.4228240157	0.288949455	0.482553889	0.3531638365	0.0782786599
10	0.4253573608	0.294806203	0.468840769	0.3531710965	0.0789355118

Context: Role. Repository: 200 Rules. Rule Creation (20% match invariant)

N° of Rules	Mean Value	Min Value	Max Value	Median	Standard Deviatic
1	0.5497961017	0.247985446	0.76332563	0.497080216	0.1710189298
2	0.8089633867	0.465799734	1.414200646	0.504352999	0.3081886621
3	1.0681599707	0.682995791	1.908666385	0.7158884535	0.4155946131
4	1.2819007992	0.897443525	2.367548694	0.940389522	0.4548228987
5	1.5765981202	1.098510776	3.048778475	1.1597180065	0.5883446928
6	1.8068403319	1.318783558	3.398752662	1.3854630535	0.5908774331
7	2.0924674732	1.502530178	4.012965409	1.641569817	0.6359916577
8	2.2639238592	1.729488505	4.026450921	1.8611158935	0.5327331952
9	2.4610067311	1.878772787	4.737651412	2.038927362	0.6449097444
10	2.68752342	2.091877041	5.730983034	2.270267849	0.7373700765

Context: Role. Repository: 200 Rules. Rule Identification (20% match invariant)

N° of Rules	Mean Value	Min Value	Max Value	Median	Standard Deviatic
1	0.5988542962	0.31789161	0.753475008	0.5581671735	0.1341373435
2	0.8196363658	0.462583476	1.276766409	0.51835918	0.310470642
3	1.0778527956	0.71935125	1.759708273	0.7430016095	0.3975628491
4	1.3149082546	0.921930505	2.210888688	0.9728561855	0.4563652551
5	1.5842862699	1.1087771	3.239730557	1.1780095425	0.6330598998
6	1.8497246819	1.321454491	3.837829512	1.3937421795	0.6299514187
7	2.0346549522	1.475538752	3.402770355	1.6146820095	0.5088068541
8	2.3661627991	1.708200182	4.924259918	1.89475205	0.6997913794
9	2.5243012418	1.904116297	4.853979758	2.0839394825	0.6822401244
10	2.7079462318	2.104371232	5.46469408	2.284529129	0.707699149

Context: Role. Repository: 200 Rules. Rule Modification (20% match invariant)

N° of Rules	Mean Value	Min Value	Max Value	Median	Standard Deviatic
1	0.5474242306	0.248152994	0.774836171	0.4925208365	0.171873924
2	0.7983704172	0.467788607	1.270077991	0.5197281725	0.2927094256
3	1.0516507722	0.681597356	1.697633344	0.710727987	0.3859133953
4	1.2974202665	0.916157288	2.162444051	0.9632383405	0.43876186
5	1.5120974522	1.088357473	3.012363166	1.1577249085	0.5247937981
6	1.848583481	1.328189501	3.611623514	1.4052585185	0.6081406978
7	2.0903501512	1.552594153	4.071193131	1.6688032045	0.6120536793
8	2.2510293386	1.685700748	4.220094192	1.8200236575	0.6210765559
9	2.5387182616	1.9464508	5.077117768	2.0992756135	0.6581715995
10	2.6984436832	2.083274626	5.325730805	2.2756913055	0.6823032162

Context: Role. Repository: 200 Rules. Rule Removal (20% match invariant)

N° of Rules	Mean Value	Min Value	Max Value	Median	Standard Deviatic
1	0.5670157026	0.255733982	0.702790913	0.5402605975	0.1582275805
2	0.5635500443	0.247644559	0.712276195	0.5457913965	0.1806652622
3	0.5704598498	0.259336123	0.723659695	0.508089123	0.1530037111
4	0.6088427325	0.276095654	0.682836693	0.52560905	0.1351270635
5	0.5682804334	0.247446408	0.71887647	0.5284514185	0.169470288
6	0.5764400089	0.243311944	0.668900842	0.5210486605	0.1401826454
7	0.5832231199	0.26125438	0.718985421	0.5478277975	0.1556853939
8	0.5557235826	0.248874525	0.703997978	0.5037291415	0.1764792152
9	0.583459252	0.250387147	0.739498092	0.5404743845	0.1633623919
10	0.5644132789	0.245370521	0.714245371	0.490690331	0.1552443603

Context: Role. Repository: 300 Rules. Rule Creation (20% match invariant)

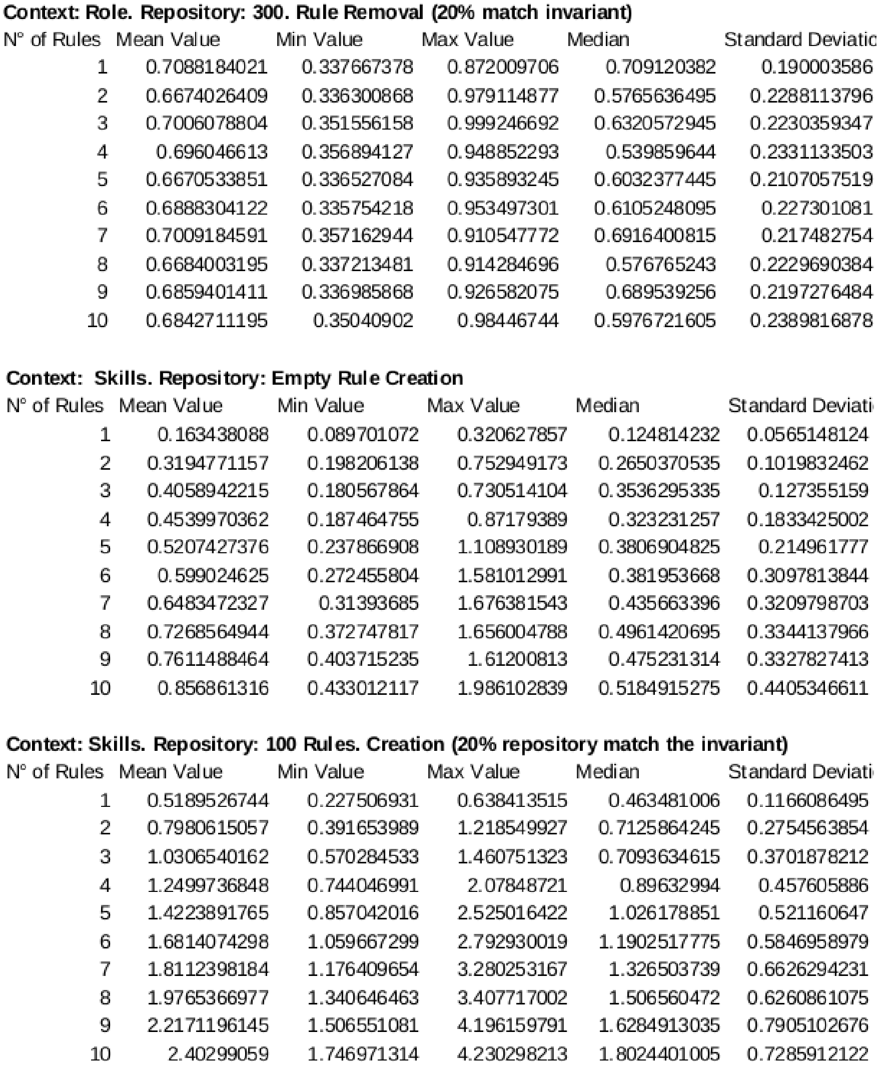
N° of Rules	Mean Value	Min Value	Max Value	Median	Standard Deviatic
1	0.6884674467	0.356016087	0.979478187	0.6283954805	0.2199878153
2	0.9976936342	0.642852697	1.61575917	0.6835247385	0.3714795184
3	1.3155573317	0.93423773	2.322537174	0.9734112485	0.4657055353
4	1.7054452852	1.289304246	3.518017829	1.334548304	0.5752025769
5	1.9883052697	1.473393087	3.667077219	1.5966664825	0.5770248347
6	2.3494406705	1.788706511	4.492595882	1.934233256	0.6521526213
7	2.6161207791	2.047715286	4.69286772	2.1893238315	0.6335624738
8	2.9453802078	2.315719899	5.904612731	2.489884993	0.7649321035
9	3.2505590441	2.623374314	6.168400183	2.784582874	0.7344215569
10	3.5611026176	2.920607086	5.845737155	3.011705713	0.6783897899

Context: Role. Repository: 300. Rule Identification (20% match invariant)

N° of Rules	Mean Value	Min Value	Max Value	Median	Standard Deviatic
1	0.6899909873	0.353313224	0.930832895	0.6411724225	0.2166065649
2	1.0197875387	0.666540276	1.72299543	0.695307483	0.374301666
3	1.3207167225	0.932320721	2.150238657	0.9743564475	0.4546934663
4	1.6876865184	1.246722018	3.388137348	1.3004427805	0.5781860667
5	2.0579332095	1.559075983	3.943860389	1.6705050965	0.5914050254
6	2.4239306197	1.86266417	4.511607866	1.9844686255	0.6274043219
7	2.6574218623	2.046918328	4.829269284	2.2218602645	0.6566806655
8	2.9513712226	2.33020323	5.158331866	2.509638133	0.6633550276
9	3.3587341951	2.738530717	6.320739111	2.90379545	0.7318170155
10	3.5382483634	2.918745745	5.879851412	3.0424531635	0.672748569

Context: Role. Repository: 300. Rule Modification (20% match invariant)

N° of Rules	Mean Value	Min Value	Max Value	Median	Standard Deviatic
1	0.6827109085	0.346725027	0.925382898	0.6161303395	0.2207298739
2	1.0166183917	0.645947156	1.79319619	0.6760760205	0.3964981611
3	1.3872439486	0.948593809	2.731644438	0.988455902	0.5342794297
4	1.7099948533	1.261420743	3.703376738	1.3117695325	0.6452134401
5	1.9887787501	1.48894954	4.158019186	1.605726118	0.5917151548
6	2.3116731813	1.776830491	4.524548269	1.8774509635	0.6275690949
7	2.6268642281	2.039284325	4.585618123	2.233363337	0.5852777606
8	2.9418174493	2.356654846	5.144377368	2.529001347	0.6252949118
9	3.1335392647	2.522197616	5.840952614	2.6803887935	0.6882431041
10	3.5453005535	2.925616313	5.857965528	3.0525491735	0.6722165833



Context: Role. Repository: 300. Rule Removal (20% match invariant)

N° of Rules	Mean Value	Min Value	Max Value	Median	Standard Deviatic
1	0.7088184021	0.337667378	0.872009706	0.709120382	0.190003586
2	0.6674026409	0.336300868	0.979114877	0.5765636495	0.2288113796
3	0.7006078804	0.351556158	0.999246692	0.6320572945	0.2230359347
4	0.696046613	0.356894127	0.948852293	0.539859644	0.2331133503
5	0.6670533851	0.336527084	0.935893245	0.6032377445	0.2107057519
6	0.6888304122	0.335754218	0.953497301	0.6105248095	0.227301081
7	0.7009184591	0.357162944	0.910547772	0.6916400815	0.217482754
8	0.6684003195	0.337213481	0.914284696	0.576765243	0.2229690384
9	0.6859401411	0.336985868	0.926582075	0.689539256	0.2197276484
10	0.6842711195	0.35040902	0.98446744	0.5976721605	0.2389816878

Context: Skills. Repository: Empty Rule Creation

N° of Rules	Mean Value	Min Value	Max Value	Median	Standard Deviati
1	0.163438088	0.089701072	0.320627857	0.124814232	0.0565148124
2	0.3194771157	0.198206138	0.752949173	0.2650370535	0.1019832462
3	0.4058942215	0.180567864	0.730514104	0.3536295335	0.127355159
4	0.4539970362	0.187464755	0.87179389	0.323231257	0.1833425002
5	0.5207427376	0.237866908	1.108930189	0.3806904825	0.214961777
6	0.599024625	0.272455804	1.581012991	0.381953668	0.3097813844
7	0.6483472327	0.31393685	1.676381543	0.435663396	0.3209798703
8	0.7268564944	0.372747817	1.656004788	0.4961420695	0.3344137966
9	0.7611488464	0.403715235	1.61200813	0.475231314	0.3327827413
10	0.856861316	0.433012117	1.986102839	0.5184915275	0.4405346611

Context: Skills. Repository: 100 Rules. Creation (20% repository match the invariant)

N° of Rules	Mean Value	Min Value	Max Value	Median	Standard Deviati
1	0.5189526744	0.227506931	0.638413515	0.463481006	0.1166086495
2	0.7980615057	0.391653989	1.218549927	0.7125864245	0.2754563854
3	1.0306540162	0.570284533	1.460751323	0.7093634615	0.3701878212
4	1.2499736848	0.744046991	2.07848721	0.89632994	0.457605886
5	1.4223891765	0.857042016	2.525016422	1.026178851	0.521160647
6	1.6814074298	1.059667299	2.792930019	1.1902517775	0.5846958979
7	1.8112398184	1.176409654	3.280253167	1.326503739	0.6626294231
8	1.9765366977	1.340646463	3.407717002	1.506560472	0.6260861075
9	2.2171196145	1.506551081	4.196159791	1.6284913035	0.7905102676
10	2.40299059	1.746971314	4.230298213	1.8024401005	0.7285912122

Context: Skills. Repository: 100 Rules. Identification (20% repository match the invariant)

N° of Rules	Mean Value	Min Value	Max Value	Median	Standard Deviati
1	0.5216085644	0.202090505	0.683519742	0.461980287	0.1481894684
2	0.8286902646	0.388103131	1.087080001	0.792127892	0.223135589
3	1.0535721842	0.567671906	1.48604058	0.747823203	0.3828395801
4	1.3089420713	0.741577403	2.113045031	0.9327171785	0.4690616642
5	1.5092694862	0.851499063	2.448474768	1.062965512	0.5617334702
6	1.6420353152	1.013373694	2.638516533	1.146043817	0.5972543253
7	1.8531333671	1.190483229	3.166527388	1.335215289	0.6630072284
8	1.9992780395	1.351779793	3.30718501	1.501421689	0.6298090381
9	2.1893404272	1.517931712	3.75841069	1.638773199	0.7095450877
10	2.3406497692	1.666548655	4.039939228	1.726792391	0.6995576454

Context: Skills. Repository: 100 Rules. Modification (20% repository match the invariant)

N° of Rules	Mean Value	Min Value	Max Value	Median	Standard Deviati
1	0.4982431005	0.300474872	0.575382316	0.4389130765	0.0806284199
2	0.7723465442	0.393611436	1.157716775	0.6619393065	0.2656740229
3	1.0722446342	0.575991819	1.830134821	0.7316137415	0.4369534536
4	1.2943645421	0.746689433	2.075327245	0.9887320515	0.4571492371
5	1.4677627511	0.852295187	2.469133331	1.0466804445	0.5217417334
6	1.6918276363	1.060416201	2.799221398	1.1960398045	0.5878659281
7	1.833978342	1.176402742	3.400671409	1.3343312375	0.6695836517
8	2.0262605581	1.402512706	3.69716973	1.571211745	0.6365084756
9	2.1602471329	1.51498127	3.699985453	1.6923961155	0.6864932351
10	2.3661560939	1.733682196	4.317829868	1.7934001945	0.6786159638

Context: Skills. Repository: 100 Rules. Removal (20% repository match the invariant)

N° of Rules	Mean Value	Min Value	Max Value	Median	Standard Deviati
1	0.55235985	0.370146786	0.643202674	0.4756960005	0.0913545385
2	0.5564621668	0.415560601	0.603828329	0.4839724015	0.0826580565
3	0.5321220819	0.355183784	0.614433581	0.464729279	0.0852489897
4	0.551312902	0.319565013	0.629017766	0.4734211985	0.1002743212
5	0.5458016786	0.341341314	0.624619854	0.4707040965	0.0959055748
6	0.5312308576	0.360896383	0.672850786	0.451858196	0.092452285
7	0.5358919363	0.370858058	0.610793106	0.475440836	0.0913455715
8	0.5210380586	0.373466764	0.670673553	0.442152978	0.0951978441
9	0.5320110515	0.314427414	0.678022956	0.433833799	0.1095055886
10	0.5465428717	0.370668643	0.71098825	0.481685449	0.0943776747

Context: Skills. Repository: 200 Rules. Creation (20% repository match the invariant)

N° of Rules	Mean Value	Min Value	Max Value	Median	Standard Deviati
1	0.6512383359	0.300652201	1.012105385	0.531362367	0.2281685812
2	0.9556608281	0.561262091	1.48796324	0.703849942	0.3353793605
3	1.2681567072	0.831944975	2.19134598	0.907566733	0.4747501687
4	1.6052268377	1.069108716	2.750726192	1.2394682685	0.5126496884
5	1.8880446951	1.272922975	3.468114904	1.4796158775	0.6413845226
6	2.2014982072	1.540097242	4.118082684	1.6984320405	0.7311902412
7	2.4475523741	1.779092162	4.46176738	1.9266507985	0.7146944162
8	2.7444889028	2.053518054	5.183761847	2.166599083	0.8092007331
9	2.9831182109	2.287296661	4.889997865	2.34194886	0.7424534422
10	3.2317522198	2.54722657	5.73320769	2.559953362	0.7933349665

Context: Skills. Repository: 200 Rules. Identification (20% repository match the invariant)

N° of Rules	Mean Value	Min Value	Max Value	Median	Standard Deviati
1	0.6596518292	0.299967755	0.892794957	0.5669286235	0.206263591
2	0.9692422917	0.568127412	1.613844542	0.6908100895	0.3484078064
3	1.2746969404	0.826763513	2.251668107	0.899347675	0.4696845238
4	1.6917299164	1.114401628	2.647893838	1.3504524525	0.5124918824
5	1.9016590008	1.286059632	3.632030856	1.481349728	0.6378201126
6	2.2669162233	1.53024152	4.113634912	1.6979292205	0.7486879339
7	2.5092024777	1.830583272	4.562156331	1.981433251	0.7799892647
8	2.7082470566	2.067338245	4.429490441	2.2338720225	0.6644758462
9	3.0008310744	2.359379151	4.503331951	2.4187103365	0.666656105
10	3.2706735327	2.563675867	5.724044938	2.588018665	0.8024497724

Context: Skills. Repository: 200 Rules. Modification (20% repository match the invariant)

N° of Rules	Mean Value	Min Value	Max Value	Median	Standard Deviati
1	0.6737953365	0.310915306	0.808013298	0.6433412735	0.1718811228
2	0.9626478244	0.55835117	1.410845335	0.720634413	0.3239271325
3	1.2636256681	0.826772107	2.069148385	0.916364122	0.4592571869
4	1.6185162051	1.066463408	2.822583728	1.2870601415	0.5280148678
5	1.9186507127	1.273381484	3.243881014	1.4731123615	0.5914247334
6	2.1766746599	1.522607589	4.039149672	1.6674944475	0.7122062192
7	2.4160378403	1.765229363	4.473528022	1.926043668	0.7359600518
8	2.7586576251	2.056410874	5.115926962	2.1732552725	0.7565004943
9	3.0297960579	2.305353145	5.574579283	2.4209494075	0.8612931376
10	3.2132529615	2.539562256	5.429033112	2.5659670465	0.73815346

Context: Skills. Repository: 200 Rules. Rule Removal (20% match the invariant)

N° of Rules	Mean Value	Min Value	Max Value	Median	Standard Deviati
1	0.6534054014	0.311399651	1.101228377	0.6553580585	0.1826743476
2	0.6535697948	0.306771271	0.882124053	0.5951231705	0.1939335282
3	0.6518666272	0.300239437	0.854167422	0.5966194075	0.1848432582
4	0.6540654642	0.29977066	0.986811739	0.5094163125	0.2348025195
5	0.6657580828	0.30355208	0.885823957	0.662903431	0.1778257583
6	0.68291197	0.311790206	0.901728352	0.6507987135	0.1993829334
7	0.6490079452	0.304086922	0.915247811	0.611366815	0.1980207268
8	0.6752509262	0.310366352	0.919747756	0.5891685315	0.1407844758
9	0.6600653193	0.302702251	0.895488737	0.5992389805	0.1855788435
10	0.6244869386	0.306525083	0.876372897	0.5735537995	0.1925246612

Context: Skills. Repository: 300 Rules. Creation (20% repository match the invariant)

N° of Rules	Mean Value	Min Value	Max Value	Median	Standard Deviati
1	0.7502201388	0.397190659	1.170623096	0.5570694085	0.2689969896
2	1.1595876755	0.740109479	1.97131025	0.8474872425	0.4273837457
3	1.6109695	1.087772894	2.755913092	1.234388952	0.5415992043
4	1.9829063576	1.352154072	3.49471724	1.6084138585	0.5690529621
5	2.4304547944	1.760177047	4.368549661	1.947514937	0.7415732861
6	2.7095777662	2.052845474	4.850152878	2.2401173025	0.6703878497
7	3.0296385385	2.367774561	4.706732121	2.471734819	0.6572776704
8	3.3977465712	2.706832896	5.95380048	2.7504547895	0.7307958622
9	3.8095366132	3.089261978	5.757061702	3.1402551615	0.7185897009
10	4.1031802506	3.340313523	5.999654631	3.393488201	0.759333654

Context: Skills. Repository: 300 Rules. Identification (20% repository match the invariant)

N° of Rules	Mean Value	Min Value	Max Value	Median	Standard Deviati
1	0.777643014	0.421151673	1.122918636	0.6115206415	0.2463321816
2	1.2504118068	0.756260626	2.022808211	0.9076930805	0.4451007283
3	1.6099141332	1.058633508	2.853785318	1.2186702325	0.5441903485
4	2.0033250739	1.372370536	3.536609717	1.622584197	0.6241514406
5	2.4342192914	1.714037099	4.542975194	1.916825004	0.7847306316
6	2.7028330196	2.019366779	4.998880306	2.2010596835	0.758463254
7	3.0318091795	2.367351113	5.446129707	2.436322595	0.7297425696
8	3.5206761746	2.836718273	5.859258567	2.884928662	0.7123758824
9	3.7560533222	3.052045795	5.555123112	3.1020205165	0.709808472
10	4.0958461289	3.490653597	5.638759508	3.539278113	0.687311721

Context: Skills. Repository: 300 Rules. Modification (20% repository match the invariant)

N° of Rules	Mean Value	Min Value	Max Value	Median	Standard Deviati
1	0.7764655721	0.417184196	1.172183209	0.5709115315	0.2720856703
2	1.188187027	0.750266702	2.279634446	0.859213246	0.4123023451
3	1.5873530472	1.064882061	3.169224383	1.19839894	0.547630749
4	1.9763682167	1.36436959	3.443860843	1.600152297	0.5687451022
5	2.428623819	1.762789384	4.273449157	1.9926711	0.6957982031
6	2.7404984791	2.042957838	4.790048371	2.2176654255	0.7787831554
7	3.0449467675	2.37502695	5.367910968	2.4837551965	0.6899170983
8	3.3815003922	2.69160611	5.548275022	2.760480432	0.6804948633
9	3.8424691253	3.131463508	5.689049776	3.1727309035	0.7038698639
10	4.0881815599	3.357671878	5.979542303	3.3969416425	0.7343215014

Context: Skills. Repository: 300 Rules. Rule Removal (20% match the invariant)

N° of Rules	Mean Value	Min Value	Max Value	Median	Standard Deviati
1	0.7684114419	0.412140466	1.206633819	0.5634823045	0.2809540887
2	0.8005558408	0.40308366	1.065212483	0.776043377	0.2421823942
3	0.7824765468	0.401662812	1.077659526	0.7249121905	0.2360462242
4	0.7709677554	0.41528884	1.111498542	0.5884540245	0.2579875683
5	0.8076151985	0.407655262	1.201787117	0.777479596	0.23677277
6	0.7645469717	0.401266315	1.178452439	0.55354281	0.268554927
7	0.8081991507	0.399050259	1.239694864	0.6856175655	0.2730605065
8	0.7781034975	0.396940633	1.281402612	0.716274079	0.2629997659
9	0.8725537699	0.401542472	1.099056312	0.7839569185	0.2461304254
10	0.7873854468	0.400742086	1.11643778	0.7219777745	0.2421238986

Context: Cognition. Repository: Empty Rule Creation

N° of Rules	Mean Value	Min Value	Max Value	Median	Standard Deviati
1	0.1626217784	0.093438909	0.448432583	0.1252720065	0.0653247595
2	0.3058089386	0.19894925	0.542072424	0.247450217	0.0767676403
3	0.3683289845	0.156631789	0.643772862	0.3190834545	0.1118981133
4	0.4711532083	0.194376324	0.884245336	0.4427252595	0.1766622803
5	0.5021663714	0.22836242	1.340103236	0.347269229	0.2624419197
6	0.561790476	0.273646328	1.445506152	0.371243621	0.2637103776
7	0.6284597729	0.308835922	1.264408978	0.468570199	0.2580004427
8	0.7062091084	0.375885018	1.289994839	0.5205815275	0.2637523801
9	0.776686249	0.392672351	1.701278702	0.590613443	0.3312311227
10	0.8217944966	0.462378068	1.516663392	0.5843676745	0.2988521418

Context: Cognition. Repository: 100 Rules. Creation (20% repository match the invariant)

N° of Rules	Mean Value	Min Value	Max Value	Median	Standard Deviati
1	0.4173288514	0.278052679	0.601017628	0.350546184	0.0863176924
2	0.6280133971	0.279974288	0.81078669	0.6078526125	0.1891300263
3	0.7472204598	0.406320663	1.093971443	0.4735608855	0.2883280013
4	0.9139040464	0.545629021	1.599634912	0.5909048195	0.3499445064
5	1.0587347614	0.666661033	1.808204453	0.755811609	0.4078878976
6	1.2588644054	0.782131514	2.1637292	0.837437291	0.4725466571
7	1.3632095386	0.923889835	2.667061286	1.031457772	0.4908151671
8	1.557246472	1.089876587	2.572065697	1.159762705	0.4680853791
9	1.72071096	1.219601878	3.103186165	1.308895392	0.5214346607
10	1.8721534539	1.31464909	3.641251724	1.397193351	0.6356489979

Context: Cognition. Repository: 100 Rules. Identification (20% repository match the invarian

N° of Rules	Mean Value	Min Value	Max Value	Median	Standard Deviati
1	0.4145964931	0.277403504	0.465876279	0.3464927685	0.073841517
2	0.6336784708	0.280838017	0.777623887	0.60094377	0.156025209
3	0.7587551168	0.412329809	1.254853567	0.4895143125	0.3046408154
4	0.9126376923	0.551768213	1.458435246	0.6108622105	0.3351085516
5	1.0426788586	0.676923366	1.795394293	0.7307884945	0.3889817173
6	1.2603179825	0.819714507	2.252131503	0.878397769	0.4616054984
7	1.3821562016	0.947517003	2.373479549	1.045239414	0.4651524918
8	1.5460203744	1.081651445	2.750435301	1.2032326895	0.4539978689
9	1.7174366606	1.230617173	3.264819428	1.3278553445	0.5287883995
10	1.9063417105	1.364139953	3.803194318	1.4839610905	0.6048244115

Context: Cognition. Repository: 100 Rules. Modification (20% repository match the invariant

N° of Rules	Mean Value	Min Value	Max Value	Median	Standard Deviati
1	0.4291717851	0.270372026	0.546705065	0.3561716855	0.0863775229
2	0.6354283743	0.288674209	0.869227511	0.6100681495	0.1824454914
3	0.7960825339	0.409921545	1.094416198	0.5602649625	0.2962202117
4	0.9405344122	0.538498368	1.652165407	0.6566089495	0.3611633818
5	1.0535954095	0.673921662	2.11240242	0.7069805815	0.4139011576
6	1.2314996834	0.801324667	2.123872109	0.8567988635	0.4094946063
7	1.4063674646	0.967333346	2.269012419	1.0396559175	0.4235912846
8	1.569938894	1.098583065	2.610707618	1.2324384985	0.4520737914
9	1.7165478997	1.236980412	2.783178852	1.357746296	0.4617689095
10	1.8792820906	1.364562564	2.814982338	1.4602986855	0.4836761354

Context: Cognition. Repository: 100 Rules. Removal (20% repository match the invariant)

N° of Rules	Mean Value	Min Value	Max Value	Median	Standard Deviati
1	0.4247614916	0.30873053	0.436816161	0.3520375755	0.0726372875
2	0.4072251517	0.33108407	0.570550156	0.3664236555	0.08934317
3	0.4184676325	0.272487928	0.585039204	0.3458430005	0.0874855594
4	0.4281033695	0.260826537	0.578303435	0.3611375795	0.09418196
5	0.4084395302	0.254904344	0.457533182	0.343479887	0.0867961874
6	0.412173727	0.243377042	0.530930232	0.3448898025	0.090604887
7	0.4089425906	0.269355199	0.498310783	0.3423337515	0.0803735289
8	0.418170248	0.294646476	0.644260771	0.3331217245	0.0919468576
9	0.4015894338	0.281063536	0.442413871	0.3260573845	0.0727763467
10	0.4204522058	0.273953972	0.60881052	0.3432483275	0.0900036508

Context: Cognition. Repository: 200 Rules. Creation (20% repository match the invariant)

N° of Rules	Mean Value	Min Value	Max Value	Median	Standard Deviati
1	0.5362396244	0.23950378	0.659794237	0.5074703985	0.1521815265
2	0.7981262701	0.448202738	1.258986217	0.505922645	0.2865583408
3	1.0436386619	0.695577139	1.953366627	0.7349602555	0.3831953081
4	1.2915218944	0.868047743	2.335168527	0.924640558	0.4764016963
5	1.5621314662	1.133343349	3.094639477	1.1849060905	0.5511234532
6	1.7842753734	1.301067054	3.360063316	1.3536384825	0.5431045404
7	2.0377510642	1.472163747	3.87654708	1.5928070875	0.6080347204
8	2.2376006448	1.672259089	4.118636947	1.794846012	0.573783861
9	2.4428105116	1.842403562	4.289623845	1.989202751	0.6040100301
10	2.6870522388	2.034308202	5.224581421	2.185951056	0.7132529185

Context: Cognition. Repository: 200 Rules. Identification (20% repository match the invarian

N° of Rules	Mean Value	Min Value	Max Value	Median	Standard Deviati
1	0.560497234	0.254768988	0.747081456	0.4660339415	0.183306881
2	0.8012608904	0.448322091	1.104295038	0.500833968	0.2926803129
3	1.0498941454	0.660799672	1.653264358	0.7014197815	0.384817174
4	1.2906904111	0.872580723	2.108417839	0.9206149775	0.4614987148
5	1.5027880685	1.076863965	2.997500242	1.1284987785	0.5340141991
6	1.804747303	1.297754343	3.735566303	1.358473587	0.6221112623
7	2.0430148541	1.463826928	3.93908765	1.6027051385	0.6192000481
8	2.2486338714	1.664081937	4.156084238	1.8107835185	0.5961366109
9	2.5336423303	1.920909568	4.411287773	2.0698707565	0.6449550707
10	2.7095445377	2.056541755	5.544356721	2.201519341	0.7650593694

Context: Cognition. Repository: 200 Rules. Modification (20% repository match the invariant

N° of Rules	Mean Value	Min Value	Max Value	Median	Standard Deviati
1	0.5283210879	0.240162453	0.685535637	0.5260495065	0.1734168553
2	0.7833650822	0.451973605	1.285573397	0.519469306	0.2900372084
3	1.0202839269	0.662780648	1.825048981	0.687788828	0.3793635785
4	1.2567759297	0.866555164	2.142694613	0.902316559	0.4616268805
5	1.5443274017	1.117018373	3.154200057	1.1718180795	0.5415147651
6	1.7972453747	1.29828542	3.549006127	1.355892884	0.5876541655
7	2.0377714841	1.448209298	4.039544832	1.58943857	0.6223068729
8	2.2946281695	1.697696754	4.259505766	1.8451827225	0.6158468137
9	2.4885093422	1.868617236	4.493900938	2.0036087345	0.701052739
10	2.7099236763	2.060629288	5.558582416	2.203826339	0.7580291503

Context: Cognition. Repository: 200 Rules. Rule Removal (20% match the invariant)

N° of Rules	Mean Value	Min Value	Max Value	Median	Standard Deviati
1	0.5524441481	0.240776567	0.680277155	0.5366201595	0.1727448227
2	0.5563043739	0.243479985	0.628696711	0.5117642395	0.1249743748
3	0.5812449507	0.351050926	0.715248495	0.5306338195	0.0996573087
4	0.5946530789	0.24391712	0.706252382	0.4983520925	0.1432112878
5	0.5829624858	0.256737677	0.680694461	0.5401713825	0.1483200151
6	0.5821669766	0.246066347	0.718109755	0.5551124855	0.161732973
7	0.5704193884	0.253561645	0.74311503	0.540122702	0.1486831218
8	0.5887279	0.257745485	0.73008167	0.5160534025	0.1620572866
9	0.5655598319	0.247563326	0.750942492	0.5226845865	0.1502430378
10	0.5587001686	0.242987465	0.734598025	0.5256975395	0.145977593

Context: Cognition. Repository: 300 Rules. Creation (20% repository match the invariant)

N° of Rules	Mean Value	Min Value	Max Value	Median	Standard Deviati
1	0.6577271431	0.334123562	1.003001356	0.6400063525	0.2163143802
2	1.0310797859	0.656473984	1.942148578	0.6835957315	0.374958067
3	1.350098447	0.959057436	2.764757583	0.9922642405	0.4806756245
4	1.7371037593	1.274702952	3.361430043	1.3293976095	0.5727085162
5	1.9881914786	1.53823765	4.015034037	1.58658301	0.5830782863
6	2.3529317443	1.792982822	4.192197831	1.915722732	0.6036496202
7	2.617445417	2.021631871	4.778295479	2.18885785	0.5909479648
8	2.94897424	2.326095308	5.499678761	2.489099767	0.6563888026
9	3.2654034355	2.60213069	5.658841599	2.8237355465	0.665921747
10	3.6144295758	2.939266923	5.364231578	3.132180984	0.6604160533

Context: Cognition. Repository: 300 Rules. Identification (20% repository match the invarian

N° of Rules	Mean Value	Min Value	Max Value	Median	Standard Deviati
1	0.6692748937	0.331145319	0.93299616	0.57710202	0.2179940896
2	1.0737092814	0.651663522	1.893350687	0.673920296	0.4122571909
3	1.3517403517	0.965890494	2.726761554	1.001346574	0.4744671514
4	1.6772173802	1.226352877	3.264795402	1.2752788255	0.5803299525
5	1.9854817431	1.492607825	3.766110691	1.579345739	0.5570805685
6	2.3122784324	1.793179008	4.105823372	1.8996084315	0.5516450286
7	2.7084155467	2.104294057	5.275979928	2.2862174025	0.6435001508
8	2.9872395834	2.345038151	5.481235638	2.513495746	0.6852953507
9	3.2509561004	2.563189491	5.908290295	2.765602109	0.7248972588
10	3.5391097354	2.876239782	5.454517296	3.0829519585	0.6626018633

Context: Cognition. Repository: 300 Rules. Modification (20% repository match the invariant

N° of Rules	Mean Value	Min Value	Max Value	Median	Standard Deviati
1	0.6679989053	0.331087911	0.975426974	0.601134786	0.223528837
2	0.989201586	0.630177272	1.837762259	0.6595805575	0.3813282719
3	1.3360498013	0.954086686	2.427477966	0.9894454135	0.4548561976
4	1.7062557961	1.272220511	3.309607318	1.318906441	0.5481026001
5	2.0372072409	1.559656572	3.782165945	1.654436372	0.5277701707
6	2.323623402	1.780333798	4.167028502	1.9142338205	0.5718703607
7	2.6214877788	2.018214465	4.502771403	2.1974635555	0.6029248308
8	2.941448781	2.306377089	5.392116275	2.477905397	0.6401425056
9	3.3490641141	2.681468153	5.541995247	2.897937441	0.6468666622
10	3.5247295581	2.880876965	5.211161289	3.074627297	0.6248133562

Context: Cognition. Repository: 300 Rules. Rule Removal (20% match the invariant)

N° of Rules	Mean Value	Min Value	Max Value	Median	Standard Deviati
1	0.7007855563	0.352276851	1.135606071	0.6813198795	0.2395537716
2	0.7412584543	0.333605553	1.005052382	0.6637346805	0.2264749676
3	0.6926438399	0.339517823	1.097899533	0.624067347	0.2426881325
4	0.7211936164	0.337813351	1.103992899	0.6152948275	0.2419150345
5	0.6711062532	0.335922187	1.120957281	0.61543109	0.2395593494
6	0.7078520692	0.351791735	1.025524055	0.6984170975	0.2130160052
7	0.6967677617	0.349316629	1.012219779	0.6268188865	0.2107463
8	0.6857019302	0.33745709	0.963447236	0.6481086745	0.2028506501
9	0.6983319388	0.350143266	1.014960422	0.624818875	0.2372892434
10	0.6753538191	0.338102071	1.053582446	0.6321019965	0.238007793

Context: Preferences. Repository: Empty Rule Creation

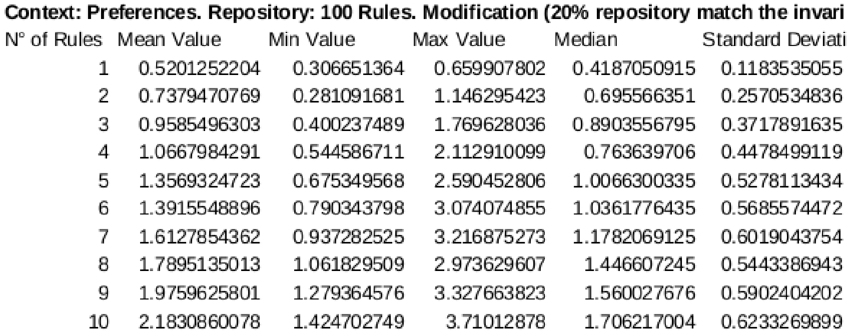
N° of Rules	Mean Value	Min Value	Max Value	Median	Standard Deviati
1	0.1620373511	0.077895801	0.436278383	0.1279853385	0.066420142
2	0.2782970857	0.156528072	0.606807611	0.239225009	0.0893301908
3	0.3563232866	0.156974434	0.600531325	0.297632118	0.1108602904
4	0.4128386291	0.162939503	0.736765828	0.36306827	0.1325643932
5	0.5276398157	0.225637355	1.093687128	0.4867530285	0.1856180854
6	0.6253376617	0.268730262	1.078593232	0.4718007455	0.2261912127
7	0.7199046048	0.351199826	1.14067473	0.596752247	0.2415477873
8	0.7795533751	0.34914901	1.244692434	0.7116024985	0.2924069914
9	0.8018493764	0.400168902	1.350247336	0.6418684265	0.2848857668
10	0.8413883612	0.438012749	1.316928568	0.645507506	0.3005865922

Context: Preferences. Repository: 100 Rules. Creation (20% repository match the invariant)

N° of Rules	Mean Value	Min Value	Max Value	Median	Standard Deviati
1	0.4712110267	0.267817507	0.729061707	0.396662243	0.1090234065
2	0.6557629527	0.283374511	1.09845037	0.624769078	0.2090934833
3	0.8324880911	0.400941278	1.504438309	0.819956086	0.3124805161
4	0.984672618	0.521570581	1.841632239	0.6752849995	0.3965120448
5	1.1542452851	0.662690975	1.993535073	0.8527288975	0.4341368651
6	1.2821983293	0.776939931	2.771577931	0.8639733445	0.5238980125
7	1.3717312743	0.916677039	2.359945912	0.989223663	0.4496120122
8	1.5660276332	1.053710666	3.082060713	1.1364916605	0.5348650679
9	1.7722213526	1.164380613	3.547159171	1.2544802335	0.6683941325
10	1.9854987404	1.292016885	4.351288456	1.450834496	0.7004832278

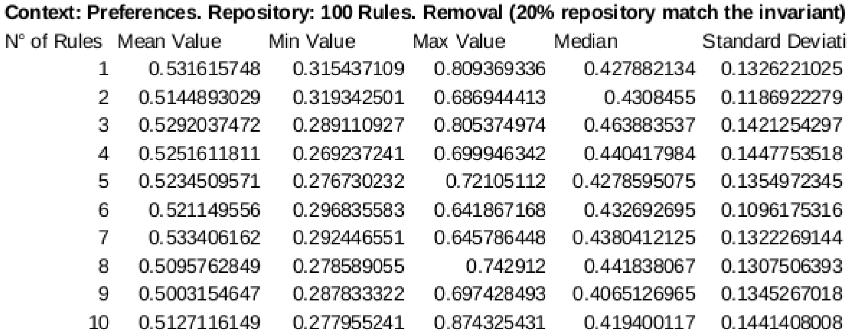
Context: Preferences. Repository: 100 Rules. Identification (20% repository match the invar

N° of Rules	Mean Value	Min Value	Max Value	Median	Standard Deviati
1	0.4437081729	0.276776748	0.613037735	0.3771847	0.0867961133
2	0.703020743	0.278615318	1.13372175	0.6700618875	0.1627392105
3	0.8574169266	0.403283546	1.531908635	0.9246853945	0.3253783579
4	0.9847703907	0.528658913	2.051760145	0.754811732	0.3834082664
5	1.1604671979	0.662586913	1.945308618	0.837584827	0.4378758414
6	1.3008098916	0.80345105	2.411263612	0.9614728605	0.481869578
7	1.3944489235	0.914822544	2.470861342	0.9874145315	0.481277803
8	1.5782287055	1.048845457	2.548181233	1.146257112	0.4918535101
9	1.7980701336	1.216546531	3.184572991	1.320760616	0.5831008363
10	1.9393592623	1.283513221	3.270486491	1.432396185	0.5707422908



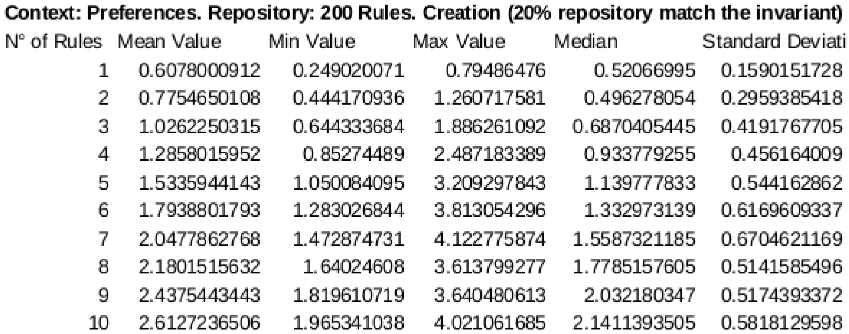
Context: Preferences. Repository: 100 Rules. Modification (20% repository match the invari

N° of Rules	Mean Value	Min Value	Max Value	Median	Standard Deviati
1	0.5201252204	0.306651364	0.659907802	0.4187050915	0.1183535055
2	0.7379470769	0.281091681	1.146295423	0.695566351	0.2570534836
3	0.9585496303	0.400237489	1.769628036	0.8903556795	0.3717891635
4	1.0667984291	0.544586711	2.112910099	0.763639706	0.4478499119
5	1.3569324723	0.675349568	2.590452806	1.0066300335	0.5278113434
6	1.3915548896	0.790343798	3.074074855	1.0361776435	0.5685574472
7	1.6127854362	0.937282525	3.216875273	1.1782069125	0.6019043754
8	1.7895135013	1.061829509	2.973629607	1.446607245	0.5443386943
9	1.9759625801	1.279364576	3.327663823	1.560027676	0.5902404202
10	2.1830860078	1.424702749	3.71012878	1.706217004	0.6233269899



Context: Preferences. Repository: 100 Rules. Removal (20% repository match the invariant)

N° of Rules	Mean Value	Min Value	Max Value	Median	Standard Deviati
1	0.531615748	0.315437109	0.809369336	0.427882134	0.1326221025
2	0.5144893029	0.319342501	0.686944413	0.4308455	0.1186922279
3	0.5292037472	0.289110927	0.805374974	0.463883537	0.1421254297
4	0.5251611811	0.269237241	0.699946342	0.440417984	0.1447753518
5	0.5234509571	0.276730232	0.72105112	0.4278595075	0.1354972345
6	0.521149556	0.296835583	0.641867168	0.432692695	0.1096175316
7	0.533406162	0.292446551	0.645786448	0.4380412125	0.1322269144
8	0.5095762849	0.278589055	0.742912	0.441838067	0.1307506393
9	0.5003154647	0.287833322	0.697428493	0.4065126965	0.1345267018
10	0.5127116149	0.277955241	0.874325431	0.419400117	0.1441408008



Context: Preferences. Repository: 200 Rules. Creation (20% repository match the invariant)

N° of Rules	Mean Value	Min Value	Max Value	Median	Standard Deviati
1	0.6078000912	0.249020071	0.79486476	0.52066995	0.1590151728
2	0.7754650108	0.444170936	1.260717581	0.496278054	0.2959385418
3	1.0262250315	0.644333684	1.886261092	0.6870405445	0.4191767705
4	1.2858015952	0.85274489	2.487183389	0.933779255	0.456164009
5	1.5335944143	1.050084095	3.209297843	1.139777833	0.544162862
6	1.7938801793	1.283026844	3.813054296	1.332973139	0.6169609337
7	2.0477862768	1.472874731	4.122775874	1.5587321185	0.6704621169
8	2.1801515632	1.64024608	3.613799277	1.7785157605	0.5141585496
9	2.4375443443	1.819610719	3.640480613	2.032180347	0.5174393372
10	2.6127236506	1.965341038	4.021061685	2.1411393505	0.5818129598

Context: Preferences. Repository: 200 Rules. Identification (20% repository match the invar

N° of Rules	Mean Value	Min Value	Max Value	Median	Standard Deviati
1	0.568219299	0.241259452	0.883709234	0.524032947	0.1741806192
2	0.8768405608	0.444461341	1.411386462	0.873092137	0.31036752
3	1.0210990959	0.642191792	1.570489445	0.6892305955	0.3654087711
4	1.2895257396	0.853539827	2.568797538	0.9364694005	0.485260308
5	1.5798067433	1.075607728	3.200491699	1.1364942415	0.5706587024
6	1.802504555	1.279671329	3.692385032	1.3335829255	0.6276877246
7	2.1010198711	1.457887139	4.942956714	1.590233192	0.7704150989
8	2.260437676	1.621094205	4.677041055	1.7609834315	0.6825779867
9	2.5264389112	1.835519795	5.335272874	1.9988670645	0.7610912525
10	2.6699057354	1.99925605	4.001622626	2.17684613	0.6182647615

Context: Preferences. Repository: 200 Rules. Modification (20% repository match the invari

N° of Rules	Mean Value	Min Value	Max Value	Median	Standard Deviati
1	0.6523698366	0.238799853	0.988031366	0.5833852515	0.2273127996
2	1.0149397676	0.439872137	1.868562406	0.810263723	0.4238133782
3	1.1636852167	0.652635714	2.25817815	0.849657208	0.4705903073
4	1.4554951206	0.890761317	2.362437433	1.125274517	0.4719419328
5	1.7734776518	1.071520755	3.368024315	1.35240344	0.590541172
6	1.9663412226	1.289568455	5.039552728	1.50584858	0.775220436
7	2.2909869595	1.738577862	4.24425753	1.806917697	0.6108112607
8	2.4951782299	1.86387249	4.669261642	1.974512823	0.6815993136
9	2.689884335	1.988725522	5.114290019	2.188877126	0.7064914053
10	2.904834095	2.202832706	5.022892577	2.397831316	0.7138724415

Context: Preferences. Repository: 200 Rules. Rule Removal (20% match the invariant)

N° of Rules	Mean Value	Min Value	Max Value	Median	Standard Deviati
1	0.6614032389	0.24674213	0.960618531	0.581739166	0.2046545311
2	0.6333863608	0.242116542	0.988762747	0.516355541	0.2512639375
3	0.6809281919	0.245342579	1.116247604	0.570193006	0.2538308828
4	0.7035826549	0.246046233	1.032704295	0.649856698	0.2370921139
5	0.7151823246	0.241225509	1.014094458	0.6326300995	0.217191155
6	0.6784094152	0.235446211	0.967919247	0.617099388	0.2281744104
7	0.6488640191	0.246681577	0.920595071	0.5758868875	0.195990866
8	0.6618912553	0.240935036	0.847325767	0.5817602945	0.1873149026
9	0.6819161431	0.243828774	1.013030357	0.610172425	0.1945442201
10	0.6850897842	0.237429984	0.987827083	0.620934096	0.2217429313

Context: Preferences. Repository: 300 Rules. Creation (20% repository match the invariant)

N° of Rules	Mean Value	Min Value	Max Value	Median	Standard Deviati
1	0.7562102018	0.327219401	1.517061213	0.570767971	0.3013134951
2	1.0185254966	0.615818051	2.149556587	0.6507879945	0.4399392648
3	1.340183588	0.917980981	2.757287638	0.958040823	0.4955985528
4	1.7613471146	1.214101321	3.697162744	1.2741774985	0.6471321725
5	2.0550012522	1.487392852	4.120848599	1.6071642545	0.6105231844
6	2.4323086222	1.764861507	4.777625075	1.879965987	0.7177350465
7	2.6430661375	1.964825695	5.018228294	2.137148027	0.6983308067
8	2.9245800197	2.264565208	4.974892966	2.4368010415	0.6305522362
9	3.249277442	2.561458691	4.875126436	2.7025433535	0.6304400329
10	3.5456280958	2.825444145	5.417465029	2.944259776	0.6842210287

Context: Preferences. Repository: 300 Rules. Identification (20% repository match the invar

N° of Rules	Mean Value	Min Value	Max Value	Median	Standard Deviati
1	0.805118335	0.337106206	1.471331447	0.6961395025	0.2806427194
2	0.9965979436	0.625628929	1.887869259	0.659253806	0.404937161
3	1.3448568511	0.907143294	3.050757706	0.9401377085	0.5652057238
4	1.6462310213	1.224789967	2.744301844	1.2861828325	0.4502888932
5	2.1005350608	1.542292565	4.176182739	1.610371263	0.6242443983
6	2.4235647632	1.807050258	4.568060155	1.937356452	0.6433002954
7	2.5666209018	1.942208571	4.512739712	2.119261884	0.5876051291
8	3.1224885913	2.405686369	5.923407165	2.574238981	0.7380369088
9	3.2860347885	2.60578367	4.826032832	2.7459139175	0.6399815333
10	3.5550960497	2.847730288	5.336894658	2.976722914	0.69592609

Context: Preferences. Repository: 300 Rules. Modification (20% repository match the invari

N° of Rules	Mean Value	Min Value	Max Value	Median	Standard Deviati
1	0.9045337943	0.334176856	1.863384303	0.7622128515	0.322856348
2	1.1688513088	0.630330995	2.321918171	0.775634597	0.5075991547
3	1.448648349	0.921348446	2.758008192	1.126805306	0.577190674
4	1.8728794581	1.273913675	3.515207278	1.434326845	0.5991163674
5	2.1594862657	1.699204553	3.988580881	1.781931419	0.6561780648
6	2.3776860879	1.747932752	4.288230023	1.8563838995	0.6768144913
7	2.68872653	1.98412235	5.168999004	2.163354817	0.7401005829
8	2.9313202608	2.257216929	4.539043405	2.4324995845	0.6168111682
9	3.1784383014	2.521535175	4.18965596	2.692931901	0.5478482735
10	3.7400991702	2.998969492	5.54963924	3.143875522	0.7258279457

Context: Preferences. Repository: 300 Rules. Rule Removal (20% match the invariant)

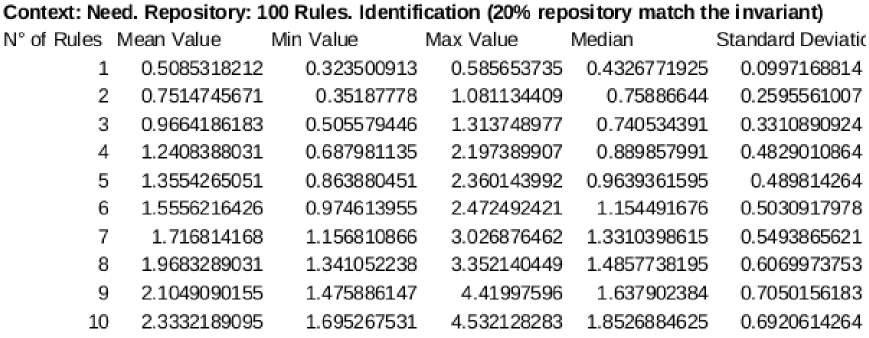
N° of Rules	Mean Value	Min Value	Max Value	Median	Standard Deviati
1	0.8480384521	0.334457131	1.8188271	0.657392356	0.3501103548
2	0.897482149	0.350133213	1.787181945	0.773930223	0.358587222
3	0.7698052822	0.335881122	1.193443453	0.6702487785	0.2882258321
4	0.7971865837	0.351984213	1.299660034	0.6789980755	0.3267108632
5	0.7622953862	0.335999293	1.363195837	0.607499143	0.2995854637
6	0.7665292376	0.333195934	1.124992084	0.574054879	0.2995876942
7	0.7935614017	0.336399971	1.146793179	0.6259151735	0.2961497375
8	0.771218242	0.340753668	1.387763344	0.583448183	0.307031478
9	0.8688873171	0.339315283	1.822321819	0.7687090995	0.3516941224
10	0.7886665941	0.336058028	1.195205402	0.628528252	0.3036216718

Context: Need. Repository: Empty Rule Creation

N° of Rules	Mean Value	Min Value	Max Value	Median	Standard Deviatic
1	0.2667589316	0.136248823	0.617188891	0.22675778	0.0875241227
2	0.407529642	0.160492126	0.530697165	0.3694100715	0.1307814384
3	0.5456099924	0.251659305	0.932260176	0.5000584555	0.1890898397
4	0.6764067549	0.350411875	1.536818479	0.498106567	0.2874666415
5	0.7822815849	0.42084255	1.567476427	0.562268323	0.3127190456
6	0.891911581	0.50553552	1.728041187	0.629978156	0.3557642399
7	1.0865599089	0.626036875	2.242357008	0.8095821425	0.4051587896
8	1.1663420795	0.704154906	2.164433131	0.845849846	0.4236109261
9	1.2897403568	0.774164019	2.760327761	0.908282959	0.4833797641
10	1.3835637643	0.880044806	2.527628409	1.0248119875	0.4859564595

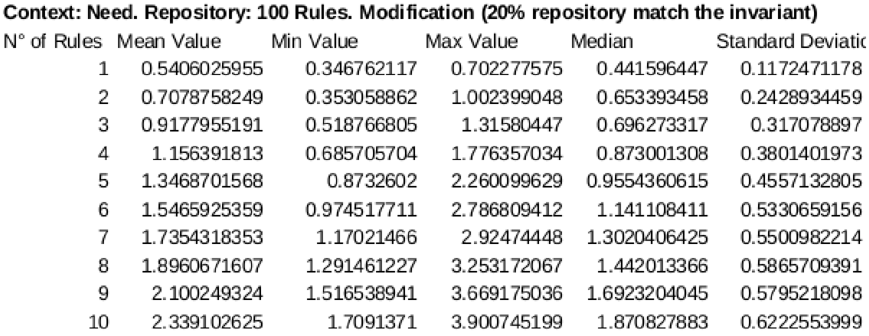
Context: Need. Repository: 100 Rules. Creation (20% repository match the invariant)

N° of Rules	Mean Value	Min Value	Max Value	Median	Standard Deviatic
1	0.4940465516	0.298043349	0.528180853	0.4268367845	0.0912345201
2	0.6830258706	0.351631735	1.012004602	0.445230363	0.25010177
3	0.9631962636	0.52363279	1.35732799	0.828116234	0.298143816
4	1.167346164	0.6766095	1.772689459	0.847801107	0.424642054
5	1.3213439736	0.86739704	2.465368191	0.939328634	0.482940325
6	1.5318473412	0.98652559	2.547364296	1.1371001475	0.496230218
7	1.7403915988	1.179006284	3.106041976	1.318192002	0.5480603936
8	1.9092310951	1.34213925	3.366070577	1.4907506045	0.5446703449
9	2.1189551177	1.504104888	4.217156107	1.66388753	0.6575507923
10	2.3315013846	1.684409238	4.363777126	1.861628776	0.7022496859



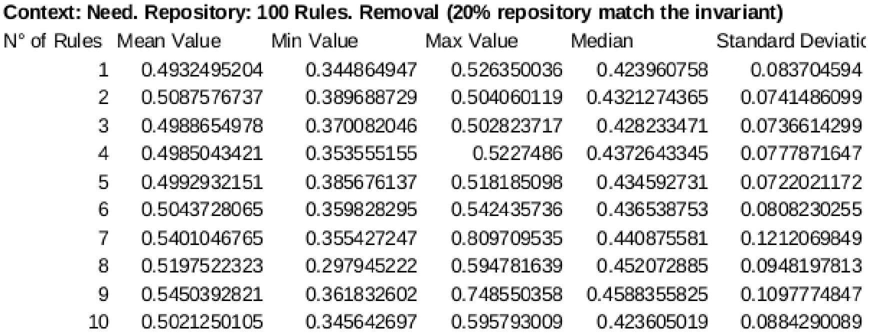
Context: Need. Repository: 100 Rules. Identification (20% repository match the invariant)

N° of Rules	Mean Value	Min Value	Max Value	Median	Standard Deviatic
1	0.5085318212	0.323500913	0.585653735	0.4326771925	0.0997168814
2	0.7514745671	0.35187778	1.081134409	0.75886644	0.2595561007
3	0.9664186183	0.505579446	1.313748977	0.740534391	0.3310890924
4	1.2408388031	0.687981135	2.197389907	0.889857991	0.4829010864
5	1.3554265051	0.863880451	2.360143992	0.9639361595	0.489814264
6	1.5556216426	0.974613955	2.472492421	1.154491676	0.5030917978
7	1.716814168	1.156810866	3.026876462	1.3310398615	0.5493865621
8	1.9683289031	1.341052238	3.352140449	1.4857738195	0.6069973753
9	2.1049090155	1.475886147	4.41997596	1.637902384	0.7050156183
10	2.3332189095	1.695267531	4.532128283	1.8526884625	0.6920614264



Context: Need. Repository: 100 Rules. Modification (20% repository match the invariant)

N° of Rules	Mean Value	Min Value	Max Value	Median	Standard Deviatic
1	0.5406025955	0.346762117	0.702277575	0.441596447	0.1172471178
2	0.7078758249	0.353058862	1.002399048	0.653393458	0.2428934459
3	0.9177955191	0.518766805	1.31580447	0.696273317	0.317078897
4	1.156391813	0.685705704	1.776357034	0.873001308	0.3801401973
5	1.3468701568	0.8732602	2.260099629	0.9554360615	0.4557132805
6	1.5465925359	0.974517711	2.786809412	1.141108411	0.5330659156
7	1.7354318353	1.17021466	2.92474448	1.3020406425	0.5500982214
8	1.8960671607	1.291461227	3.253172067	1.442013366	0.5865709391
9	2.100249324	1.516538941	3.669175036	1.6923204045	0.5795218098
10	2.339102625	1.7091371	3.900745199	1.870827883	0.6222553999



Context: Need. Repository: 100 Rules. Removal (20% repository match the invariant)

N° of Rules	Mean Value	Min Value	Max Value	Median	Standard Deviatic
1	0.4932495204	0.344864947	0.526350036	0.423960758	0.083704594
2	0.5087576737	0.389688729	0.504060119	0.4321274365	0.0741486099
3	0.4988654978	0.370082046	0.502823717	0.428233471	0.0736614299
4	0.4985043421	0.353555155	0.5227486	0.4372643345	0.0777871647
5	0.4992932151	0.385676137	0.518185098	0.434592731	0.0722021172
6	0.5043728065	0.359828295	0.542435736	0.436538753	0.0808230255
7	0.5401046765	0.355427247	0.809709535	0.440875581	0.1212069849
8	0.5197522323	0.297945222	0.594781639	0.452072885	0.0948197813
9	0.5450392821	0.361832602	0.748550358	0.4588355825	0.1097774847
10	0.5021250105	0.345642697	0.595793009	0.423605019	0.0884290089

Context: Need. Repository: 200 Rules. Creation (20% repository match the invariant)

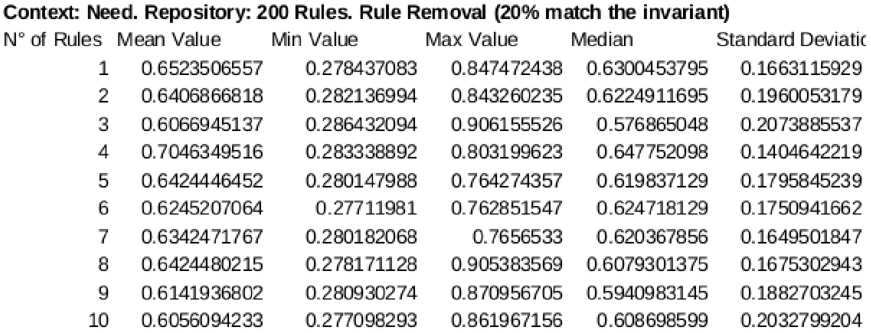
N° of Rules	Mean Value	Min Value	Max Value	Median	Standard Deviatic
1	0.630852045	0.285057899	0.879563316	0.6002470685	0.1907829563
2	0.8784067415	0.512373535	1.413112446	0.5849016145	0.307614966
3	1.1712858112	0.760784313	1.760399143	0.882239599	0.3732476733
4	1.482265289	1.008942551	2.426256864	1.1076333085	0.4732943436
5	1.8531690782	1.231100962	3.750827597	1.350050173	0.6739275682
6	2.0318854661	1.453235075	3.867450114	1.6230339735	0.6121057485
7	2.432326361	1.812413855	4.797836932	1.968531023	0.7111884792
8	2.6281875863	1.985191834	4.975858733	2.092712523	0.7450794992
9	2.8272403082	2.218533095	4.250943224	2.2593207585	0.5859698468
10	2.9808767645	2.364514927	4.905039604	2.4289238215	0.6195941352

Context: Need. Repository: 200 Rules. Identification (20% repository match the invariant)

N° of Rules	Mean Value	Min Value	Max Value	Median	Standard Deviatic
1	0.6571570029	0.282422647	0.843942302	0.626655141	0.164280387
2	0.8892510941	0.521709989	1.273633821	0.5866864055	0.3157441025
3	1.1744058519	0.756521131	1.902839736	0.878081669	0.4047379918
4	1.4877591873	1.011616203	2.442571994	1.1298152185	0.4641194572
5	1.8064502127	1.225893667	3.376256435	1.371167623	0.5836293062
6	1.9816118703	1.43905625	3.510994587	1.6068240605	0.5246321563
7	2.3154025664	1.697102822	4.560487822	1.8506595025	0.6980855102
8	2.6251550243	1.994316457	5.079952416	2.080866777	0.7388409973
9	2.8201869364	2.22091706	4.401026694	2.2681823925	0.5821218025
10	3.0996201005	2.477416226	4.366300842	2.5344346885	0.587937973

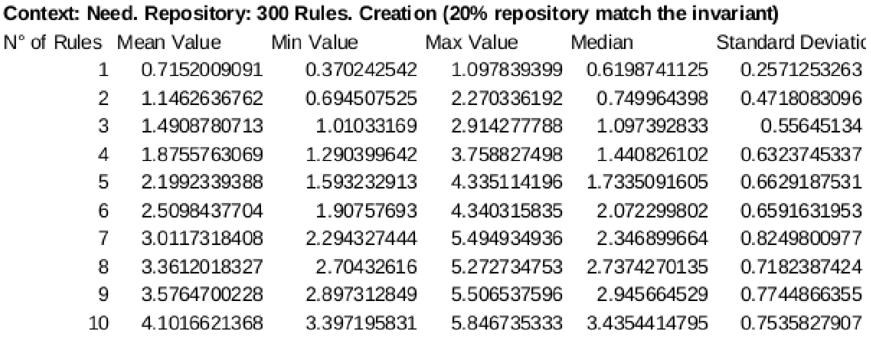
Context: Need. Repository: 200 Rules. Modification (20% repository match the invariant)

N° of Rules	Mean Value	Min Value	Max Value	Median	Standard Deviatic
1	0.6205440951	0.281344717	0.83143157	0.5972548935	0.2025790833
2	0.9242179702	0.513830286	1.370654186	0.735842177	0.3209177706
3	1.1742471349	0.748871141	1.939502437	0.8779977525	0.4024920826
4	1.4677170577	1.017767207	2.436960453	1.103064502	0.460428498
5	1.8036525995	1.233105899	3.290031645	1.364543706	0.5839063919
6	2.0062323523	1.457190751	3.478794476	1.633564924	0.5289984105
7	2.3295404345	1.731472278	4.789929434	1.8799566985	0.7021198164
8	2.6036638811	1.980429005	5.342844998	2.079780328	0.7452900561
9	2.8156707586	2.218685769	4.443259441	2.279142689	0.5830023706
10	3.1415478972	2.524308364	5.189789289	2.606759558	0.6468431132



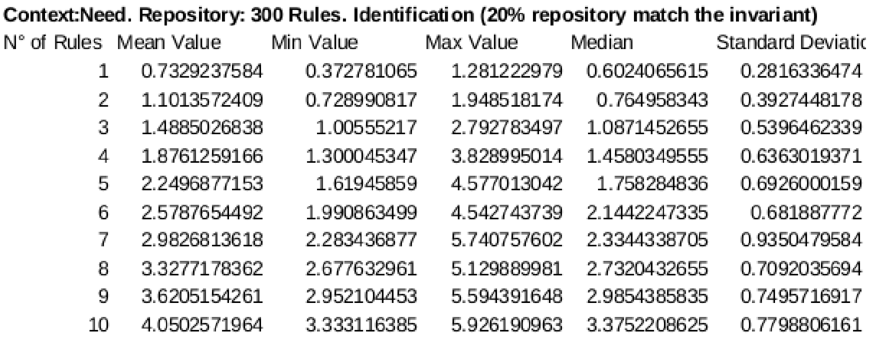
Context: Need. Repository: 200 Rules. Rule Removal (20% match the invariant)

N° of Rules	Mean Value	Min Value	Max Value	Median	Standard Deviatic
1	0.6523506557	0.278437083	0.847472438	0.6300453795	0.1663115929
2	0.6406866818	0.282136994	0.843260235	0.6224911695	0.1960053179
3	0.6066945137	0.286432094	0.906155526	0.576865048	0.2073885537
4	0.7046349516	0.283338892	0.803199623	0.647752098	0.1404642219
5	0.6424446452	0.280147988	0.764274357	0.619837129	0.1795845239
6	0.6245207064	0.27711981	0.762851547	0.624718129	0.1750941662
7	0.6342471767	0.280182068	0.7656533	0.620367856	0.1649501847
8	0.6424480215	0.278171128	0.905383569	0.6079301375	0.1675302943
9	0.6141936802	0.280930274	0.870956705	0.5940983145	0.1882703245
10	0.6056094233	0.277098293	0.861967156	0.608698599	0.2032799204



Context: Need. Repository: 300 Rules. Creation (20% repository match the invariant)

N° of Rules	Mean Value	Min Value	Max Value	Median	Standard Deviatic
1	0.7152009091	0.370242542	1.097839399	0.6198741125	0.2571253263
2	1.1462636762	0.694507525	2.270336192	0.749964398	0.4718083096
3	1.4908780713	1.01033169	2.914277788	1.097392833	0.55645134
4	1.8755763069	1.290399642	3.758827498	1.440826102	0.6323745337
5	2.1992339388	1.593232913	4.335114196	1.7335091605	0.6629187531
6	2.5098437704	1.90757693	4.340315835	2.072299802	0.6591631953
7	3.0117318408	2.294327444	5.494934936	2.346899664	0.8249800977
8	3.3612018327	2.70432616	5.272734753	2.7374270135	0.7182387424
9	3.5764700228	2.897312849	5.506537596	2.945664529	0.7744866355
10	4.1016621368	3.397195831	5.846735333	3.4354414795	0.7535827907



Context:Need. Repository: 300 Rules. Identification (20% repository match the invariant)

N° of Rules	Mean Value	Min Value	Max Value	Median	Standard Deviatic
1	0.7329237584	0.372781065	1.281222979	0.6024065615	0.2816336474
2	1.1013572409	0.728990817	1.948518174	0.764958343	0.3927448178
3	1.4885026838	1.00555217	2.792783497	1.0871452655	0.5396462339
4	1.8761259166	1.300045347	3.828995014	1.4580349555	0.6363019371
5	2.2496877153	1.61945859	4.577013042	1.758284836	0.6926000159
6	2.5787654492	1.990863499	4.542743739	2.1442247335	0.681887772
7	2.9826813618	2.283436877	5.740757602	2.3344338705	0.9350479584
8	3.3277178362	2.677632961	5.129889981	2.7320432655	0.7092035694
9	3.6205154261	2.952104453	5.594391648	2.9854385835	0.7495716917
10	4.0502571964	3.333116385	5.926190963	3.3752208625	0.7798806161

Context: Need. Repository: 300 Rules. Modification (20% repository match the invariant)

N° of Rules	Mean Value	Min Value	Max Value	Median	Standard Deviatic
1	0.7117743021	0.371567644	1.075781424	0.6177227595	0.2491444718
2	1.0912350751	0.687952775	1.965587313	0.7332810275	0.4085152414
3	1.4829510955	1.009474737	2.847857606	1.10354836	0.5173520435
4	1.9307706658	1.297635964	3.501695923	1.454332704	0.6349334719
5	2.2328148341	1.607451692	4.14970905	1.741571849	0.6510013991
6	2.580015676	1.939374763	4.78171079	2.1015827865	0.770283167
7	2.9924519889	2.325158384	5.684414406	2.371080847	0.8459739375
8	3.3698186921	2.713443866	5.569903062	2.7433619505	0.7070548851
9	3.5972743297	2.92769185	5.184205478	2.9740486585	0.6853591118
10	3.9903297409	3.2971512	5.781466276	3.3396016005	0.7554174197

Context: Need. Repository: 300 Rules. Rule Removal (20% match the invariant)

N° of Rules	Mean Value	Min Value	Max Value	Median	Standard Deviatic
1	0.7376770449	0.374585553	1.216609735	0.7055344775	0.263181527
2	0.7625901401	0.373494629	1.048719323	0.7366997935	0.2615570493
3	0.7314085983	0.372342739	1.107063614	0.62501499	0.2490238563
4	0.7137723918	0.369828942	1.166781753	0.7369812505	0.2444838664
5	0.7500532527	0.36727925	1.089189832	0.5735691305	0.2796493122
6	0.7847929934	0.369994746	1.161943359	0.694685582	0.2678839059
7	0.7405061381	0.370833399	1.011694017	0.725172664	0.2542762945
8	0.7162858184	0.37084136	1.041727863	0.529859838	0.2606154579
9	0.7304187959	0.371778634	1.100241309	0.5633247405	0.2648906751
10	0.7593508012	0.370695258	1.101859532	0.5908952885	0.2826437438