Monitoring the Quality of Service to support the Service Based System lifecycle

PhD thesis

Presented by:

Marc Oriol Hilari

Advisors:

Dr. Xavier Franch

Dr. Jordi Marco

Programa de Doctorat en Computació
Departament de Ciències de la Computació
Universitat Politècnica de Catalunya

January 2015

Acknowledgments

Some years ago, I was asked if I wanted to pursue a PhD. It was a great opportunity and the beginning of this journey, which, thanks to many people, today is reaching 'land in sight'.

First and foremost, I would like to express my deepest thanks to my two research supervisors: **Xavier Franch** and **Jordi Marco**. They gave me the opportunity, the tools, the guidance, the support, their great insights and everything I could ask for. And for all this, I could not have asked for better supervisors.

My colleagues in the office: **David Ameller, Oscar Cabrera, Oscar Hernán, Lidia López, Silverio Martínez, Cristina Palomares, Marc Rodríguez** and all the **GESSI** members, who helped me through all these years, shared many moments, and contributed with great discussions that improved this work.

Colleagues and researchers abroad: **Anna Perini, Itzel Morales, Nauman A. Qureshi, Antonio Ruiz, Carlos Müller, Manuel Resinas, Andreas Metzger, Osama Sammodi, Eric Schmieders, Attila Kertész, Gábor Kecskeméti** and **Attila Marosi,** who made of this journey a great and more enriched experience.

I would like to thank to the **anonymous reviewers**, who provided priceless feedback and great ideas, so I could make of this thesis a much better version.

To my current and former flatmates: **Cristina Gené, Juan Carlos Isaza, Nacho Llatzer** and **Sabela Regueiro,** for keeping the home in Barcelona always worm.

To my family: **mom, dad, brother** and **sister**. Thank you for all your support. *I did it!*

And to **all my friends**, without them, I would have finished a lot earlier.

This work has been possible thanks to the funding of the Spanish projects TIN2007-64753 and TIN2010-19130-c02-01; and the EC 7th Framework Programme FP7/2007–2013 under grant agreement 215483 (S-Cube) and 318249 (RISCOSS).

Abstract

Service Oriented Computing (SOC) has been established in the last recent years as a successful paradigm in Software Engineering. The systems built under this paradigm, known as Service Based System (SBS), are composed of several services, which are usually third-party software run by external service providers.

SBS rely on these service providers to ensure that their services comply with the agreed Quality of Service (QoS). In contrast to other systems, the dynamic behaviour of SBS requires up-to-date QoS information for its proper management in the different stages of its lifecycle, from their initial construction until their decommission.

Providing such QoS information has resulted in different technological solutions built around a monitor. Nonetheless, several research challenges in the field remain still open, ranging from theoretical aspects of quality assurance to architectonical challenges in decentralized monitoring.

Based on the current research challenges for service monitoring, the research gaps in which we aim to contribute are twofold:

- To investigate on the definition and structure of the different quality factors of services, and provide a framework of common understanding for the definition of **what to monitor**.
- To investigate on the different features required to support the activities of the whole SBS lifecycle (**i.e.** **how to monitor**), and develop a monitoring framework that accomplishes such features.

As a result of this thesis, we provide:

What to monitor

- A distribution of the quality models along the time dimension and the identification of their relationships.
- An analysis of the size and definition coverage of the proposed quality models.
- A quantified coverage of the different ISO/IEC 25010 quality factors given by the proposals.
- The identification of the most used quality factors, and provided the most consolidated definitions for them.

How to monitor

- The elicitation of the requirements of the different activities in the SBS lifecycle.
- The definition of the set of features that supports the elicited requirements.
- A modular service-oriented monitoring framework, named SALMon, implementing the defined features. SALMon has been validated by including it in several frameworks supporting the different activities of the SBS lifecycle. Finally, we have conducted a performance evaluation of SALMon over real web services.

Table of contents

List of figures

List of tables

CHAPTER 1 Introduction

"Every new beginning comes from some other beginning's end" - Seneca.

1.1 Context and terminology

The present thesis has been developed within the scope of Service Oriented Computing (SOC) and the Quality of Service (QoS). These two topics are described in the following subsections in order to provide a general context and clarify the concepts and terminology used along the thesis.

1.1.1 Service Oriented Computing

Software Engineering studies principles and methods to improve the quality of software and its development [1]. An emerging paradigm in Software Engineering is Service Oriented Computing (SOC), which is defined by Mike P. Papazoglou as follows:

> Service-Oriented Computing (SOC) is the computing paradigm that utilizes services as fundamental elements for developing applications/solutions.
>
> **Mike P. Papazoglou**, *Service -Oriented Computing: Concepts, Characteristics and Directions* [2]

SOC encompasses many things, including design principles, protocol standards, technologies, etc. and it is primarily characterized by applying an architectural model known as Service Oriented Architecture (SOA) [3]. A system following this architecture is composed of several services, being each one responsible to accomplish a specific and finer-granular functionality. These services can then be combined as building blocks for the composition of larger systems, known as Service Based Systems (SBS) [3].

A specific type of services are web services, which provide a functionality following web standards such as SOAP, WSDL, REST, etc. As defined by the European Network S-Cube:

> "A Web Service is a service provided by a software system that implements a predefined set of standards. It is designed to support interoperable machine-to-machine interaction over a network."
>
> **S-Cube Network,** *CD-IA-1.1.1 Comprehensive overview of the state of the art on service-based systems* [4]

The main benefits of SOA is that it improves the reusability, abstraction, loose coupling and high cohesion of the services that compose them [5]. Due to these characteristics, SOC has been established as a successful paradigm in both industry and academia. In industry, SOC has proved to be a suitable solution for many projects in the last decade [6], whereas in the academia it is acknowledged to be a consolidated approach which progresses along with cloud computing and the Future Internet [7].

1.1.2 Quality of Service

Services in an SBS are usually third-party software run by external service providers and executed through the Internet.

In contrast to other systems, SBS rely on these service providers to ensure that their services comply with the agreed Quality of Service (QoS), which is defined by ISO/IEC as follows:

> [Quality of Service (QoS) is] the totality of features and characteristics of a product or service that bear on its ability to satisfy stated or implied needs.
>
> **ISO/IEC,** *ISO 8402:1994 - Quality management and quality assurance – Vocabulary* [8]

Delivering an adequate QoS is a critical and significant challenge because of the dynamic and unpredictable nature of the Internet [10]. QoS comprises many characteristics (e.g. performance, security, reliability, etc.), and these characteristics are usually structured in the form of a quality model. Stephen T. Albin defines a quality model as follows:

> A quality model is a specification of the required characteristics that a software system must exhibit.
>
> **Stephen T. Albin,** *The Art of Software Architecture: Design Methods and Techniques* [9]

As the ISO/IEC 25010 software quality standard states, quality models are useful for specifying requirements, establishing measures and performing quality evaluations [11]. There exist many proposals of quality models for software systems.

They differ on the terminology that they use, the set of quality characteristics they define, and the structure of the quality model. The most widely adopted one is the ISO/IEC series of quality standards, especially the 9126 [12] and its replacement, 25010 [11]. Both of them include a concrete proposal of quality model that classifies software quality into a structured set of high-level characteristics and sub-characteristics. Figure 1 shows the 25010 proposal.

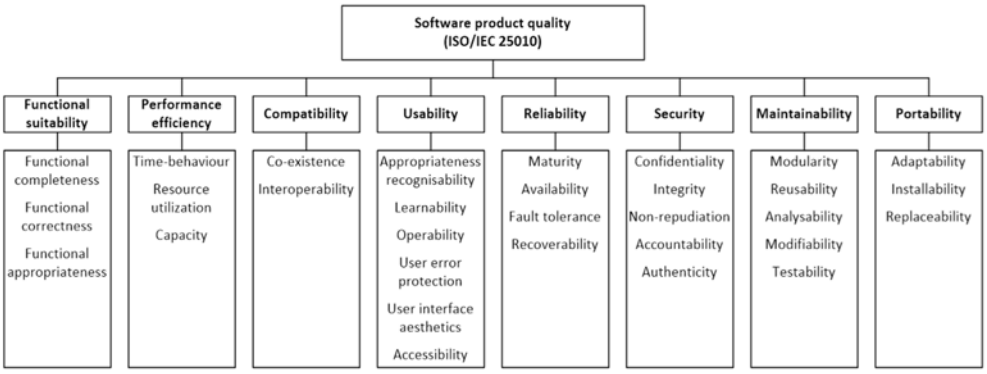


Figure 1 – ISO/IEC 25010 quality model for software products.

These characteristics and subcharacteristics can be in turn decomposed into finer-granular quality attributes and metrics. As defined by Standard Glossary of Software Engineering Terminology (IEEE-610), a quality attribute is "a feature or characteristic that affects an item's quality" [13]. In turn, a quality metric is defined as "a quantitative measurement of the degree to which an item possesses a given quality attribute" [14]. An example of a quality attribute is *response time* and a quality metric is the *average response time during a time interval*. Figure 2 makes explicit the relationship between the different concepts.

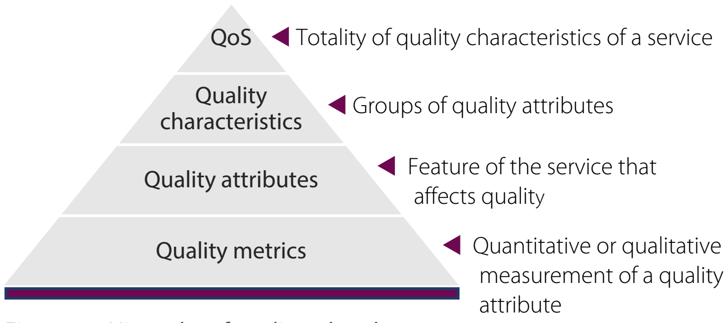

QoS
Totality of quality characteristics of a service
Quality characteristics
Groups of quality attributes
Quality attributes
Feature of the service that affects quality
Quality metrics
Quantitative or qualitative measurement of a quality attribute

Figure 2 – Hierarchy of quality related concepts.

1.2 Motivation

The adoption of a SOC approach impacts on the system's development lifecycle with additional stages and activities that do not usually occur in other paradigms. Figure 3 presents a lifecycle based on the SOC lifecycle presented in [7] complemented with the SOC lifecycle as presented in [15].

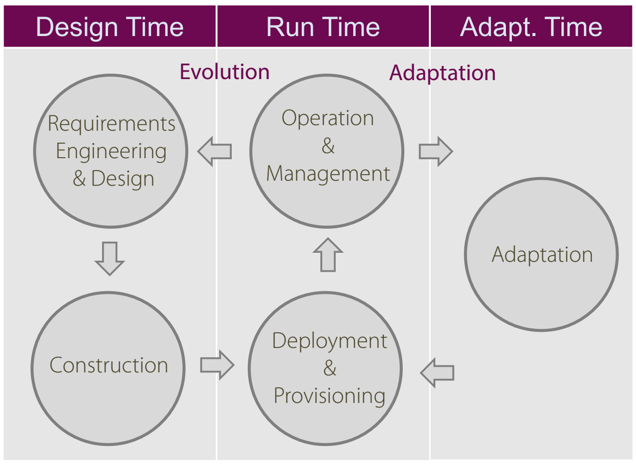


Figure 3 – Lifecycle of an SBS.

In contrast to other traditional software systems, the dynamic behaviour of SBS requires up-to-date QoS information for its proper management in the different stages of the lifecycle, from their initial construction until their decommission:

- **Requirements Engineering & Design.** The activities in this phase define the functional and non-functional requirements of the system. Such non-functional requirements are usually expressed in the form of a Service Level Agreement (SLA), which defines the conditions to be met with respect to the QoS of the system. This phase, instead of requiring QoS information, defines the required QoS information for the following phases.
- **Construction.** During the construction of the SBS, the composition of the services of the system is defined. One of the activities of this stage is *service selection*, which requires QoS information in order to select the most suitable web service from a set of candidates to fulfil a certain task.
- **Deployment & Provisioning.** The activities in this stage include the *deployment of the SBS*, which comprise the deployment of internal web services, their components and resources. Such deployment requires QoS information to assess and allocate the correct resources for the SBS.
- **Operation & Management.** The activities in this stage deal with the execution of the SBS. One of the activities is *quality assessment*, which requires QoS information to assure at execution time that the QoS expressed in the SLA is met.
- **Adaptation.** This phase comprise all the activities to perform an adaptation of the SBS. The activity that requires QoS information in this phase is the *identification of an adaptation need*. This activity consists in identifying if a service fails or does not perform as expected to trigger an adaptation strategy.

Providing such QoS information has resulted in different technological solutions built around a **monitor**. As defined by the European Network S-Cube:

> “A monitor is a program or set of programs that observes the execution of the SBS […] and is implemented on the basis of a monitoring specification that describes the properties and events to be observed.”
>
> **S-Cube Network,** *PO-JRA-1.2.1 State of the Art Report, Gap Analysis of Knowledge on Principles, Techniques and Methodologies for Monitoring and Adaptation of SBAs* [16]

Several works addressing the inherent challenges on service monitoring have been proposed. Nonetheless, as pointed out by different studies, like Marconi et al. [17] or Safy et al. [18], several research challenges in the field remain still open, ranging from theoretical aspects of quality assurance to architectonical challenges in decentralized monitoring.

1.3 Objective, research questions and hypotheses

Based on the current research challenges for service monitoring, the research gaps in which we aim to contribute are twofold:

- On the one hand, the classification and terminology of what is required to be monitored (i.e. quality characteristics, attributes and metrics) is not currently following a complete and consistent model along the different monitoring proposals. Current software quality model standards from ISO/IEC [11][12] are not sufficient to solve the problem since (1) they lack of some important quality attributes that are specifically applicable only in the field of SOC (e.g.

service provider reputation) and (2) their scope do not reach to the granularity of quality attributes (e.g. execution time). These weaknesses has led to numerous proposals of quality models specific for web services with a finer granularity. However, an in-depth evaluation of the different proposals is still missing.

- On the other hand, there are several existing monitoring solutions that are specifically designed to support a specific activity (e.g. service selection), but as it will be shown later, none of the proposed solutions are adequate to support the SBS lifecycle as a whole.

The scope of this thesis will be limited to web services; and the objectives are to fill the previously mentioned research gaps by means of:

- To investigate on the definition and structure of the different quality factors of services, and provide a framework of common understanding for the definition of **what to monitor**. The research question to attain this objective is formulated in RQ1 (see Table 1). Since this main research question is very general, we refined it into finer-grained subquestions. First, we want to identify the proposals in the field, find their interrelationships and distribute them along time to find any significant trend (RQ1.1). Second, we want to make explicit the main characteristics of these quality models, in terms of size, structure and completeness (RQ1.2). Third, we want to find which is the scope of these quality models. Namely, what are the quality factors that attract more attention from researchers and also which attract less, since both things together may help to understand priorities of researchers and eventually some research gaps (RQ1.3). Last, we want to uncover the agreed body of knowledge among the different proposals. Particularly, we aim at identifying the most recurrent definitions of quality factors in these proposals, which in some sense could be considered as the starting point of any new future proposal (RQ1.4).

Table 1 – Research question 1 of this thesis.

RQ 1	**In the field of web services, do software quality models proposed so far provide an adequate and structured set of quality factors to define their quality of service?**
RQ 1.1	What is the chronological overview of the research done so far in quality models for web services?
RQ 1.2	What are the characteristics of the proposed quality models?
RQ 1.3	Which quality factors are the most addressed in these quality models? Which are the least addressed?
RQ 1.4	What are the most consolidated quality factors?

For each research question, we define a research hypothesis to evaluate (see Table 2).

Table 2 – Hypotheses for RQ 1.

Hypotheses for RQ1	
H 1.1	It is possible to draw a chronological overview of the proposed quality models for web services, with their relationships and influences.
H 1.2	It is possible to identify the list of characteristics and evaluate the different quality models based on their size and definition.
H 1.3	It is possible to identify which characteristics are the most and least addressed.
H 1.4	It is possible to identify the most consolidated quality factors.

- To investigate on the different features required to support the activities of the whole SBS lifecycle (**i.e. how to monitor**), and develop a monitoring framework that accomplishes such features. The research question to attain this objective is formulated in RQ2 and is decomposed in several subquestions (see Table 3). First, we want to investigate on how the current State of the Art supports the different activities of the SBS lifecycle. Second, we want to explore what are the features required to support the different activities. Finally, we investigate how these features can be bond together in a single framework.

Table 3 – Research question 2 of this thesis.

RQ 2	**In the field of web services, what are the features and strategies required to monitor the QoS of web services during the whole SBS lifecycle?**
RQ 2.1	To which degree the proposed monitoring frameworks support the whole SBS lifecycle.
RQ 2.2	What are the features required when monitoring QoS to support the SBS lifecycle?
RQ 2.3	How these features can be implemented in a single framework?

For each research question, we define a research hypothesis to evaluate (see Table 4).

Table 4 – Hypotheses for RQ 2.

Hypotheses for RQ2	
H 2.1	Current monitoring frameworks do not support the whole SBS lifecycle.
H 2.2	A list of required features to support the SBS lifecycle can be identified.
H 2.3	All the required features can be implemented in a single framework

1.4 Illustrative example

In this section, we describe an illustrative example of an SBS. We will use this example throughout the thesis to illustrate and explain the functionality of the proposed approach.

We have not developed the following SBS from scratch. Instead, the SBS is largely based on a BPEL process described in [19], and consists of an example of an e-commerce SBS to purchase books online. One of the main functionalities of the SBS is described below using the following use case (see Figure 4):

Anne is a user interested on a particular book. She introduces into the SBS the input to retrieve information about the book and its book rating, which will let her decide whether to purchase the book or not. The task of retrieving the book information and rating is accomplished by a web service developed in-house. Once Anne retrieves such information, she decides to purchase the book, and the SBS searches in different bookstores using the web services of those stores (i.e. web services developed from third parties) and retrieves its prices. As these stores can be from different countries with different currencies, another third party web service is used to convert the prices into the local currency of the user.

Finally, Anne selects the bookstore she prefers and purchases the book.

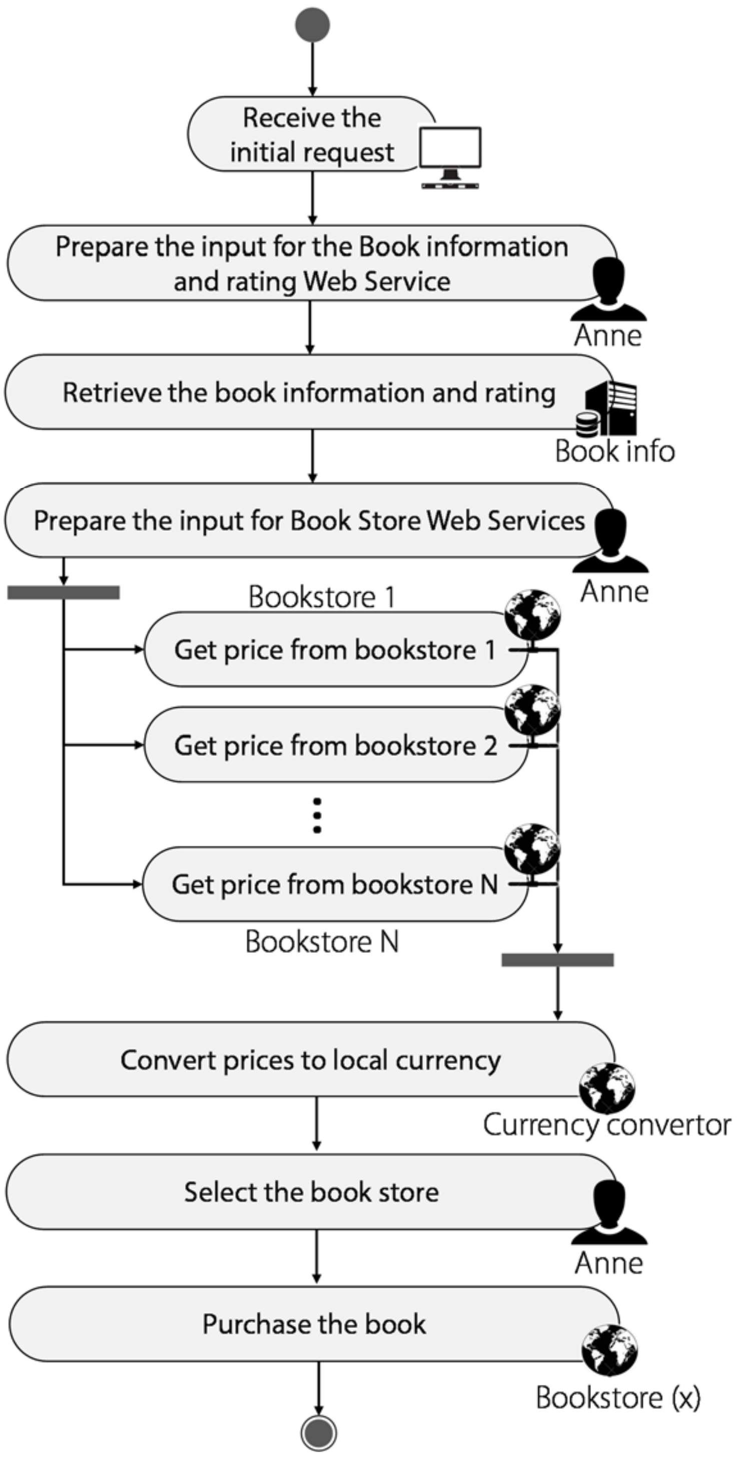

Receive the initial request
Prepare the input for the Book information and rating Web Service
Anne
Retrieve the book information and rating
Book info
Prepare the input for Book Store Web Services
Anne
Bookstore 1
Get price from bookstore 1
Get price from bookstore 2
Get price from bookstore N
Bookstore N
Convert prices to local currency
Currency convertor
Select the book store
Anne
Purchase the book
Bookstore (x)

Figure 4 – BPEL example of e-commerce SBS.

As shown, the SBS consists of one internal web service to retrieve the information and ratings of books (*BookInfo*), several external web services of bookstores (*BookStore(x)*), and one external web service to convert the currency (*CurrencyConvertor*).

1.5 Methodological approach

Mary Shaw studied and provided in [20] several ways of characterizing software engineering research, in terms of what she describes as research settings, research products, and validation techniques. We describe each of these terms bellow and define the characterizations of the research questions of this thesis.

1.5.1 Research settings

Research settings are the different classes of research problems. Shaw lists five research settings along with a sample question as example (see Table 5).

Table 5 – Shaw's list of research settings.

Research setting	Sample question
Feasibility	Is there an X, and what is it? Is it possible to accomplish X at all?
Characterization	What are the important characteristics of X? What is X like? What, exactly, do we mean by X? What are the varieties of X, and how are they related?
Method/Means	How can we accomplish X? What is a better way to accomplish X? How can I automate doing X?
Generalization	Is X always true of Y? Given X, what will Y be?
Selection	How do I decide between X and Y?

The Shaw's research settings of this thesis are characterization, and method/means.

- In RQ1, we address the characterization of the different quality factors of web services
- In RQ2, we define the means on how to monitor those quality factors.

1.5.2 Research products

Research products are the tangible results of the research project. Shaw lists five research products along with a short description of how to achieve it (see Table 6).

Table 6 – Shaw's list of research products.

Research product	Research approach or method
Qualitative or descriptive model	Organize and report interesting observations about the world. Create and defend generalizations from real examples. Structure a problem area; formulate the right questions. Do a careful analysis of a system or its development.
Technique	Invent new ways to do some tasks, including procedures and implementation techniques. Develop a technique to choose among alternatives.
System	Embody result in a system, using the system development as both source of insight and carrier of results.
Empirical predictive model	Develop predictive models from observed data.
Analytic model	Develop structural (quantitative or symbolic) models that permit formal analysis.

The research products of this thesis include qualitative or descriptive model, technique, and system.

- In RQ1, we provide qualitative/descriptive models about the characterization of the different quality factors of web services.
- In RQ2, we provide techniques on how to monitor those quality factors and a system implementing these techniques.

1.5.3 Research validation

The last characterization is the research validation. Shaw provides a list of five validation techniques (see Table 7). The validation techniques used in this thesis are persuasion, implementation, evaluation, and experience. We do not use the analysis technique as it consists of identifying consequences from rigorous formulae in the form of derivation and proof, and it is not applicable in this thesis.

Table 7 – Shaw's list of validation techniques.

Technique	Character of validation
Persuasion	A technique, design or example.
Implementation	Of a system or technique.
Evaluation	With respect to a descriptive model, a qualitative model, an empirical quantitative model.
Analysis	Of an analytic formal model, an empirical predictive model.
Experience	Expressed in a qualitative or descriptive model, as decision criteria or an empirical predictive model.

Persuasion is used all along the thesis, using examples that illustrate the behaviour of the proposed ideas or processes. For the implementation technique, a tool has been developed that demonstrates the approach and the features proposed in RQ2. Evaluations are conducted in both RQ1 and RQ2. The former by measuring the quality models and their quality factors through defined criteria, whereas the latter by measuring the performance of the monitor and its suitability to support the different activities of the SBS lifecycle. Finally, experience is used in RQ2, by integrating the monitor in different frameworks that required QoS information for different activities of the SBS lifecycle.

- In RQ1, the techniques included are persuasion and evaluation.
- In RQ2, the techniques included are persuasion, implementation, evaluation and experience.

1.6 Contributions of this thesis

1.6.1 Contributions of RQ1

The structure and definition of the quality factors of web services have been studied by analysing in depth the different quality models proposed for web services through a rigorous systematic mapping methodology. The need of such study was identified after the applicant finalized his Master Thesis, which included a more generic Systematic Literature Review on QoS in SBS [21]. As a result of this Master Thesis, it was identified that current monitoring approaches do not follow a consistent terminology and definition of quality factors. At the same time, it was identified that the proposed quality models for web services were not aligned, leading to the need of studying in depth the proposed quality models and the definition of their quality factors. The main goal and contribution of this systematic mapping is to provide a reference for prospective researchers and practitioners in the field, especially to help avoiding the definition of new proposals that do not align with current research.

Furthermore, other contributions of the systematic mapping include:

- A distribution of the quality models along the time dimension and the identification of their relationships.
- An analysis of the size and definition coverage of the proposed quality models.
- A quantified coverage of the different ISO/IEC 25010 quality factors given by the proposals.
- The identification of the most used quality factors, and provided the most consolidated definitions for them.

Results of such study were published in the SCI-indexed journal *Information and Software Technology* (I.F.: 1,522) [22].

1.6.2 Contributions of RQ2

The different features to monitor the quality factors along the SBS lifecycle were studied and developed iteratively, resulting in the proposed monitoring framework, named SALMon.

A preliminary version of SALMon was developed by the applicant in his Master Thesis [22], following the ideas contrived by Ameller and Franch in [23]. At that time, SALMon was able to check simple SLAs[1] with the sole purpose of supporting self-adaptation. Preliminary results of SALMon for self-adaptation were published in [24].

With the results of the Master Thesis as a starting point, the applicant has developed a monitoring framework able to support the whole SBS lifecycle, which is the contribution of RQ2.

During the development of this thesis, the applicant has actively participated in the European Network of Excellence in Software Services and Systems (S-Cube) as an associate member in the GESSI research group. Under the S-Cube umbrella, we initiated several collaborations to study and integrate SALMon in the different activities of the SBS lifecycle. The collaborations in which SALMon has been used include web service selection, SBS deployment in the cloud, quality assurance with SLAs, and several SBS adaptation frameworks. In each collaboration, the applicant has enhanced SALMon to meet the requirements of each activity.

- **Web service selection:** The enhancements of SALMon started with web service selection, which were introduced in a collaboration within the UPC with O. Cabrera. Results of such collaboration were a framework named WeSSQoS. The initial results were published in [25] and later refined in [26] and [27].

[1] In fact, SALMon is an acronym of SLA Monitoring (swapping the A and L).

- **SBS deployment:** SBS deployment was a collaboration with the research centre MTA SZTAKI (Hungary). SALMon was included in a framework named FCM to support dynamic deployment of SBS. Preliminary results of such collaboration were published in [28] and improved later in [29] in the *International Conference on Parallel, Distributed, and Network-Based Processing* (CORE-B). Final results were presented in the SCI-indexed journal the *Journal of Grid Computing*. (I.F.: 1,603) [30].

- **Quality assurance:** Quality assurance was a collaboration with Universidad de Sevilla (Spain), combining SALMon with the ADA platform [31]. The result of such collaboration was the SALMonADA framework, which is aimed at ensuring the compliance of SLAs at runtime. The initial results of such collaboration were published in [32][33], and the final results were published in the SCI-indexed journal *IEEE Transactions on Services Computing* (I.F.: 2,460) [30].

- **SBS adaptation:** The initial collaboration of SALMon, which started with the Master Thesis and previous to the S-Cube network, was with Johannes Kepler Universität (Austria), with a framework named MAESoS. The development of MAESoS was refined during this thesis with results published in [34] and [35]. In the S-Cube network, we initiated several collaborations for this activity. With Universität Duisburg-Essen (Germany) and MTA SZTAKI (Hungary), we contributed in a framework named PROTEUS to support proactive adaptation of SBS. Preliminary results were published in [29]. With Universität Duisburg-Essen (Germany) SALMon was introduced in the PROSA Framework. The results of such collaboration were published in [36] in the *IEEE International Computers, Software, and Applications Conference* (CORE B).

The applicant has also done a research stay in Fondazione Bruno Kessler (Italy) for a total period of 4 months. During this stay, he collaborated in another SBS framework, in which SALMon was combined with the CARE framework to support adaptive requirements. Results of such collaboration were published in [37] in the *Requirements Engineering: Foundation for Software Quality* (CORE-B).

Co-authorship statement: In all the aforementioned collaborative papers, the applicant was the contributor to all the aspects related to monitoring, whereas the contributions of the specific research aspects that are out of scope of this thesis (e.g. service selection algorithms, adaptation strategies, etc.) were conducted mainly by the mentioned partners.

Statement

Beyond those collaborations, publications describing the insights of SALMon were published in [38]. And the final development and results of SALMon has been submitted to the journal *Expert Systems With Applications* (I.F.: 1,854).

Furthermore, SALMon is currently extended beyond the field of SOC. Particularly, SALMon is extended to monitor different sources of information to assess the health of an Open Source Software (OSS) community. This work is part of the FP7 European Project RISCOSS. Preliminary results of OSS Health monitoring with SALMon have been published in [39] in the *International Conference on Research Challenges in Information Science* (CORE-B).

Finally, SALMon has been also used in frameworks where the applicant has not been involved, yet are worth to mention. SALMon has been used to monitor the accuracy of predictive services [40][41], which was part of the Master thesis of S. Martínez [42].

1.6.3 List of publications

Table 8 summarizes the list of publications by the applicant, highlighting the ones in most important venues.

Table 8 – List of publications.

Ref	Pub. Type	Venue	Title	Year	Remarks	Cit.
Quality models for web services						
[22]	**Journal**	**IST**	**Quality models for web services: A systematic mapping**	**2014**	**I.F.: 1,522**	**2**
SALMon in detail						
[43]	**Journal**	**ESWA**	**Monitoring the SBS' lifecycle with SALMon**	**2014**	**Submitted I.F.: 1,854**	**0**
[38]	Tech. report	LSI-UPC	SALMon: A SOA System for Monitoring Service Level Agreements	2010		3
[24]	Workshop	Mona+	Monitoring adaptable soa-systems using salmon	2008		36
SALMon in web service selection						
[27]	Journal	CyS	Open Framework For Web Service Selection Using Multimodal and Configurable Techniques	2014	-	0
[26]	Tech. report	arXiv	WeSSQoS: a configurable SOA system for quality-aware web service selection	2011	-	4
[25]	National conference	JCIS	WeSSQoS: Un sistema SOA para la selección de servicios web según su calidad	2009	-	2
SALMon in SBS deployment						
[30]	**Journal**	**JGC**	**Enhancing federated cloud management with an integrated service monitoring approach**	**2013**	**I.F.: 1,603**	**8**
[29]	**International conference**	**EuroPDP**	**Integrated monitoring approach for seamless service provisioning in federated clouds**	**2012**	**CORE B**	**13**
[28]	Workshop	S-Cube	A holistic service provisioning solution for federated cloud infrastructures	2012	-	2
SALMon in quality assurance						
[44]	**Journal**	**TSC**	**Comprehensive Explanation of SLA Violations at Runtime**	**2013**	**I.F.: 2,460**	**5**

[32]	Workshop	PESOS	SALMonADA: A Platform for Monitoring and Explaining Violations of WS-Agreement-Compliant Documents	2012	-	8
[33]	National conference	JCIS	SALMonADA: A Platform for Monitoring and Explaining Violations of WS-Agreement-Compliant Documents	2012	-	0
SALMon in SBS adaptation						
[37]	**International Conference**	**REFSQ**	**Requirements monitoring for adaptive service-based applications**	**2012**	**CORE B**	**1**
[36]	**International conference**	**COMPSAC**	**Usage-based online testing for proactive adaptation of service-based applications**	**2011**	**CORE B**	**18**
[45]	Workshop	WoSS	Combining SLA prediction and cross layer adaptation for preventing SLA violations	2011	-	9
[35]	Workshop	COMPSACW	Goal-driven adaptation of service-based systems from runtime monitoring data	2011	-	11
[34]	Tech. report	LSI-UPC	Monitoring and Adaptation of Service-oriented Systems with Goal and Variability Models	2009	-	3
SALMon beyond SOC						
[39]	**International conference**	**RCIS**	**Assessing Open Source Communities' Health using Service Oriented Computing Concepts**	**2014**	**CORE B**	**0**

1.7 Structure of this thesis

This thesis is presented in two parts, which corresponds to the two research questions exposed in Section 1.3. The first part, *What to monitor - Study on quality models*, refers to RQ1, and the second part, *How to monitor - Study on monitoring systems*, refers to RQ2.

- Part I – Chapter 2 describes a Systematic Mapping conducted to study the different quality models for web services. Section 2.1 presents a brief introduction to Systematic Literature studies. Section 2.2 describes the protocol followed for this Systematic Mapping. Section 2.3 presents the results of the Systematic Mapping. Finally, section 2.4 describes the threats to validity and the actions taken to mitigate them.
- Part II – Chapter 3 presents the State of the Art in field of monitoring QoS using a Systematic Literature Review. Section 3.1 describes the protocol and section 3.2 presents the results of the SLR. Chapter 4 describes the proposed monitoring framework. Section 4.1 introduces the features of SALMon, section 4.2 describes the architecture, section 4.3 summarizes the technologies used in SALMon, section 4.4 describes the interaction of the different components of SALMon, section 4.5 presents the performance evaluation and section 4.6 describes the usage of SALMon in different frameworks to support the different activities of the SBS lifecycle.

Chapter 5 provides the conclusions of this thesis. Section 5.1 describes the conclusions of RQ1, whereas section 5.2 describes the conclusions of RQ2. Section 5.3 presents the future work. Finally, chapter 6 presents the bibliography.

Part I

What to monitor

Study on quality models

CHAPTER 2 What to monitor
Study on quality models

"Even though there are no ways of knowing for sure, there are ways of knowing for pretty sure" – Lemony Snicket.

The analysis of QoS-related issues becomes crucial for several web service activities. In this regard, all kind of proposals and approaches for different activities involving the definition and analysis of QoS in web services have emerged. Maybe the first question that may be raised is: what are the facets that compose QoS? In other words, what criteria can make a "good" service? The answer to this question requires determining a shared universe of discourse, a common framework of understanding upon which different actors may communicate effectively, without ambiguities. Quality models are the engineering artefacts that have been proposed to structure and standardize the concepts and definitions of the quality factors of the QoS in web services. However, as it happens in many other software engineering areas, there does not exist a single quality model agreed by the community, instead many ad-hoc proposals have emerged in the last decade. These proposals may diverge in several matters: addressed facets, size, structure, terminology, underlying principles, etc. It is important to identify and relate the existing proposals, assess them with respect to some criteria and conclude which is the most consolidated body of knowledge and on the contrary, which are the most remarkable gaps to bridge.

The aim of this section is to identify and evaluate the different quality models for web services proposed in the literature. We have conducted a systematic mapping to accurately retrieve and analyse the different quality models by defining and conducting a rigorous protocol following the guidelines described by Kitchenham in [46]. As a result of such research, we have been able to evaluate the current state of the art in quality models and identify the strengths and weaknesses of its current status.

2.1 Introduction to systematic literature studies

To conduct any type of literature study in an accurate and objective manner, it is necessary to use a precise and rigorous methodology. For such a purpose, we have followed the principles and guidelines defined by Kitchenham [46].

These guidelines have been derived from other existing studies used by medical researchers and adapted to reflect the specific problems of software engineering research. Since their inception, these guidelines have been widely used by software engineering researchers and when applied properly, they drastically reduce the risk of bias and incompleteness in the review results.

The stages of the methodology, as defined by B. Kitchenham [3], are as follows:

- **Planning the review:** activities performed before conducting the review. These activities include the definition of the protocol that specifies the criteria that will be used to perform the review (e.g. search keywords, bibliographic databases, selection criteria, etc.).

- **Conducting the review:** activities that constitute the execution of the review and follow the protocol as defined in the previous phase. These activities include the identification of research, the selection of primary studies, the study of quality assessment, data extraction and synthesis.
- **Reporting the review:** activities to report the results of the review. It includes the specification of a dissemination mechanism, the format of the report, and its evaluation.

2.2 Planning the review

2.2.1 Identification of the need for a review

As Kitchenham states, prior to undertaking a systematic literature study, we ensured that such a study is necessary by searching for others in the subject and assessing their quality through a quality assessment checklist.

There is no procedure defined in [46] in order to implement this search. But to make this step also systematic, we applied two tactics. First, to increase the number of results, we searched not only other systematic literature studies, but also, all type of reviews and state of the art documents, regardless of the methodology followed for developing them. Second, we followed a procedure analogous to the main search of our systematic mapping. That is, we defined a search protocol to identify other reviews. Such protocol is based on the protocol defined in the main search, which will be explained in the following subsections. Hence, we searched for other reviews once the systematic mapping protocol was defined and before the systematic mapping was conducted. In a glance, we combined automatic and manual searches to the same databases and resources, using the same keywords with the addition of the following terms: “state of the art”, “SLR”, “survey”, “review[2]” and “systematic mapping”. As a result, in the automatic search we found 47 papers fulfilling the search criteria (details are presented in [47]). However, after inspecting them, we found that none of them presented a review on quality models for web services. Similarly, with respect to the manual search, we found 1 paper fulfilling the search criteria, which was also discarded for the same reason.

Therefore, after performing the searches, we did not find any literature study on quality models for web services.

2.2.2 The research questions

The next step of the review is the formulation of the research question, which in turn, will drive the review methodology. To formulate it, we followed the PICO structure [46]. PICO is an acronym which stands for Population, Intervention, Comparison and Outcome. It consists of explicitly identifying these concepts in the research question in order to derive later the keywords to perform the search. In our case, though, the Comparison is more a kind of general analysis of the field, since we do not aim at ranking the proposals found or to compare to some other existing approach. The research question that we aim at, which was described in Section 1.3, is depicted here in Table 9.

[2] We use this term, more general than “systematic literature review”, to make the search more robust with respect to terminological variations.

Table 9 – Research Question 1 and its subquestions.

RQ 1	In the field of web services, do software quality models proposed so far provide an adequate and structured set of quality factors to define their quality of service?
RQ 1.1	What is the chronological overview of the research done so far in quality models for web services?
RQ 1.2	What are the characteristics of the proposed quality models?
RQ 1.3	Which quality factors are the most addressed in these quality models? Which are the least addressed?
RQ 1.4	What are the most consolidated quality factors?

The Population, Intervention and Outcome are identified from the main research question as shown in Table 10.

Table 10 – Research question 1: Population, Intervention and Outcome.

RQ 1	In the field of **web services** (P), do **software quality models** (I) proposed so far provide **an adequate and structured set of quality factors to define their quality of service** (O)**?**

2.2.3 Bibliographic sources

The search process can be either automatic through the usage of bibliographic databases or manual through gathering the works from specific known journals and conferences of the target field. Kitchenham et al. analysed in [48] the advantages and drawbacks of both approaches through a case study.

From their analysis we decided to combine both approaches by performing an automatic search to some selected databases and additional manual searches to the most relevant conferences and journals if they were missing. The selected databases, were: ISI Web of Science (WoS), IEEE Xplore and ACM Digital Library.

With respect to manual searches, we identified the list of journals and conferences depicted in Table 11 and Table 12 respectively. The list of journals was obtained from the top ranked journals in services, software engineering and information systems engineering based on their JCR Impact Factor. Similarly the list of conferences was obtained from the top ranked conferences based on the CORE index[3], selecting those which had a CORE-A or CORE-A* status.

Table 11 – List of journals targeted in the systematic mapping.

Journal title	Acronym	IF *2014*
ACM - Transactions on Software Engineering and Methodology	TOSEM	1.55
ACM - Transactions on the Web	TWEB	1.41
Elsevier - Advances in Engineering Software	AES	1.22
Elsevier - Information and Software Technology	IST	1.52
Elsevier - Journal of Systems and Software	JSS	1.14
IEEE – Computer	-	1.68
IEEE - Internet Computing	-	2.04
IEEE – Software	-	1.62
IEEE Transactions on Services Computing	TSC	2.46
IEEE Transactions on Software Engineering	TSE	2.59
Springer - Automated Software Engineering	ASE	1.40
Springer - Software Quality Journal	SQJ	0.85
Springer – World Wide Web	WWW	1.20

Table 12 – List of conferences targeted in the systematic mapping.

Conference name	Acronym	Core status
Automated Software Engineering Conference	ASE	CORE-A
International Conference on Advanced Information Systems Engineering	CAiSE	CORE-A
ACM SIGSOFT International Symposium on the Foundations of Software Engineering	FSE	CORE-A
International Conference on Software Engineering	ICSE	CORE-A*
International Conference on Service Oriented Computing	ICSOC	CORE-A
IEEE International Conference on Web Services	ICWS	CORE-A
IEEE International Conference on Services Computing	SCC	CORE-A
International Conference on Web Information Systems Engineering	WISE	CORE-A
International World Wide Web Conference	WWW	CORE-A*

[3] http://core.edu.au/index.php/categories/conference%20rankings

Through checking the list of editions of each conference and journal from 2001 to 2012, we verified that the journals were indexed with all their editions published in ISI WoS. However, with respect to conferences, we identified that, although all these conferences were indexed (with the exception of FSE), most of them did not have all the editions present in the database. Hence, in the next stage of the systematic mapping, we had to perform a manual search in these missing editions (see Table 13).

Table 13 – Coverage of manual and automatic searches.

	2001	2002	2003	2004	2005	2006	2007	2008	2009	2010	2011	2012
ASE	Auto.	Auto.	Auto.	Auto.	Auto.	Auto.	Auto.	Auto.	Auto.	Manual	Auto.	Auto.
CAiSE	Auto.	Auto.	Manual	Manual	Manual	Manual	Manual	Manual	Manual	Manual	Auto.	Manual
E-Science	-	-	-	-	Auto.	Manual	Auto.	Manual	Auto.	Auto.	Auto.	Auto.
FSE	Manual	Manual	Manual	Manual	Manual	Manual	Manual	Manual	Manual	Manual	Manual	Manual
GRID	Manual	Manual	Auto.	Auto.	Auto.	Auto.	Auto.	Auto.	Auto.	Auto.	Auto.	Auto.
ICSE	Auto.	Auto.	Auto.	Auto.	Auto.	Auto.	Auto.	Auto.	Auto.	Auto.	Auto.	Auto.
ICSOC	-	-	Manual	Manual	Manual	Manual	Auto.	Manual	Manual	Manual	Auto.	Manual
ICWS	-	-	Manual	Auto.	Auto.	Auto.	Auto.	Auto.	Auto.	Auto.	Auto.	Auto.
SCC	-	-	-	Auto.	Auto.	Auto.	Auto.	Auto.	Auto.	Auto.	Auto.	Auto.
WISE	Auto.	Auto.	Auto.	Auto.	Auto.	Auto.	Auto.	Auto.	Auto.	Auto.	Auto.	Manual
WWW	Auto.	Auto.	Auto.	Auto.	Auto.	Auto.	Auto.	Auto.	Auto.	Auto.	Auto.	Auto.

To gather more results, the guidelines recommend also to retrieve documents from the grey literature. To do so, we also performed manual searches in the deliverables, standards and working drafts from international consortiums and networks which are well established in the community (see Table 14).

Table 14 – List of consortiums and networks selected to search in the grey literature.

Organization Name	Acronym	Official webpage
Distributed Management Task Force	DMTF	http://www.dmtf.org/
European Telecommunications Standards Institute	ETSI	http://www.etsi.org/
Internet Engineering Task Force	IETF	http://www.ietf.org/
Networked European Software and Services Initiative	NESSI	http://www.nessi-europe.com
Organization for the Advancement of Structured Information Standards	OASIS	https://www.oasis-open.org
Open Grid Forum	OGF	http://www.gridforum.org/
OpenGroup	-	http://www.opengroup.org/
World Wide Web Consortium	W3C	http://www.w3.org/
Web Services Interoperability Organization	WS-I	http://www.ws-i.org/

2.2.4 Keywords used

The keywords used for the search were retrieved from the PICO terms of the research questions. Particularly we extracted them from the Population and Intervention. In the guidelines [46], it is suggested to extract the keywords also from the Comparison and Outcome, which is the common procedure in the field of medicine. However, as it stated also by Kitchenham in [49] and identified in other SLRs [50] and systematic mappings [51], this is not always applicable. For instance, it is applicable in SLRs when performing a comparison between two already known different approaches [46]. However, in our case we evaluated the different proposed approaches found in the literature. Similarly, the outcome in our research questions is not based on a particular measurement and hence it cannot be included as a keyword.

From each term of the Population and Intervention of the research questions, we identified a set of variants and acronyms:

Table 15 – Keywords related to the Population and Intervention.

PICO	Terms
Population: web service	"web service", webservice, "web services", webservices, WS
Intervention: Quality models	"quality model", "quality models", QM, "quality ontology", "quality ontologies", QO, "quality of service", "quality of services", QoS

To build the query string, the terms inside **Population** and **Intervention** are composed through an OR connector (e.g. **Population**: "web service" OR "webservice" OR "web services" OR "webservices" OR WS). Then **Population** and **Intervention** are composed through an AND connector.

The resulting query string is then:

(“web service” **OR** webservice **OR** “web services” **OR** webservices **OR** WS) **AND** (“quality model” **OR** “quality models” **OR** QM **OR** “quality ontology” **OR** “quality ontologies” **OR** QO **OR** “quality of service” **OR** “quality of services” **OR** QoS)

which can be simplified into the following version:

(webservice? **OR** "web service?" **OR** “WS”) **AND** (“quality model?" **OR** QM **OR** "quality ontology" **OR** "quality ontologies" **OR** QO **OR** "quality of service?” **OR** QoS)

In particular, we simplified singular and plural forms by using the ‘?’ wildcard due to known limitations of the length of the query in ISI Web of Science. Although the resulting terms are not fully equivalent, we considered the probability of the wildcard delivering results other than plurals to be very low. The simplified query string was applied to search into ISI Web of Science. To search into IEEE Xplore and ACM DL, the original query string was used.

2.2.5 Selection criteria

The search was conducted by title, abstract and keywords in ISI Web of Science, IEEE Xplore and ACM DL. For the conference editions not in the aforementioned databases (see Table 13), we conducted manually the same search over title, abstract and keywords.

After retrieving the results, we conducted several steps applying the following selection criteria to filter the candidates:

- **Selection by title**: The objective of this first filter is to quickly identify and remove noise from the results. After this selection, documents whose scope is clearly unrelated to quality models for web services were removed.

- **Selection by abstract:** At this stage, we discarded all those works that although being related to quality models, did not present a quality model as contribution of the paper.
- **Selection by full paper (fast reading):** At this point, we removed the papers which did not accomplish properly the following inclusion criteria: (1) presenting a quality model as one of the contributions of the paper; (2) defining explicitly the quality model.
- **Literature from organizations:** Deliverables or similar documents from the international consortiums and organizations identified above cannot be conducted through the same process as they might lack of an abstract or mechanisms to perform a systematic search. Hence the search of these documents against the list of resources provided in the web pages of the organizations was conducted manually. The works satisfying the same inclusion criteria as previously were added into the systematic mapping.
- **Addition of further work (snowballing):** During the systematic mapping process, other works were included through the process of snowballing. In this sense, we included those works that were the basis for the development of the quality model proposed in the retrieved papers. The works satisfying the same inclusion criteria as previously were added into the systematic mapping.

From our searches we retrieved 1004 papers automatically from ISI WoS, 1105 from IEEE Xplore and 189 from ACM DL. From the total of 2298 papers, 438 were removed as they were repeated in the selected databases, leading to 1860 papers found automatically. Then, we added 28 papers found from the manual searches from the selected journals and conferences. From the initial set of 1888 papers retrieved, 518 were discarded by title and 1031 papers were excluded by abstract; resulting in 339 papers to evaluate by full paper. From them, 15 papers were initially not available, and after contacting the authors, we were able to retrieve only 5. Hence, 10 papers were not accessible. Then we evaluated the works by full paper, 284 papers were discarded, resulting in 45 papers to include in the systematic mapping.

Then 5 documents found in the literature from the listed organizations were added. Finally, 15 additional papers were added through snowballing resulting in the final 65 papers. The complete list of papers involved in the search process is available at [47]. A summary of the selection process is depicted in Figure 5.

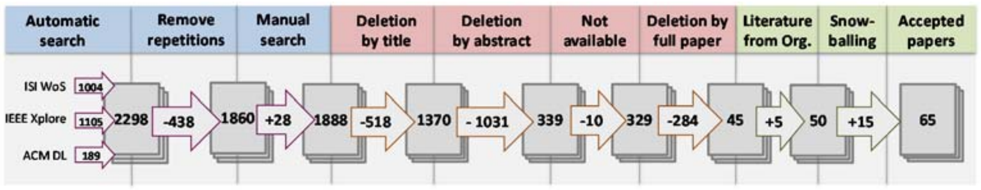


Figure 5 – Selection process of the systematic mapping.

2.3 Results of the review

We conducted the review according to the criteria and protocols defined in the previous section. First, we read the 65 selected works individually and consolidated the information therein. The 65 works present 47 different proposals of quality models (i.e. some proposals are explained in more than one work) that we summarized in tables that are included in [47]. Each table includes the name of the quality factor, an identifier system that reproduces the hierarchy and the definition as given by the authors. Next we address our research questions (RQ1.1, RQ1.2, RQ1.3 and RQ1.4) on the basis of this information.

2.3.1 What is the chronological overview of the research done so far in quality models for web services?

This section answers the Research Question RQ 1.1.

Quality models for web services have been an increasingly addressed research topic from 2001 to the current days. Figure 6 shows the number of papers considered in two-year periods starting at 2001. As shown there, the number of papers found (i.e. fulfilling the search criteria) has been increasing as well as the number of papers that have been selected in our systematic mapping (i.e. fulfilling the topic and quality criteria). However, the period 2011-2012 shows a decreased amount of papers on both aspects with respect to the previous period.

On the other hand, as shown in Figure 6, the number of papers found has been increasing at a faster pace than the selected ones. This fact implies that, although the topic has been gaining increasing attention, the actual proposals of new or improved quality models have not been increasing to the same degree.

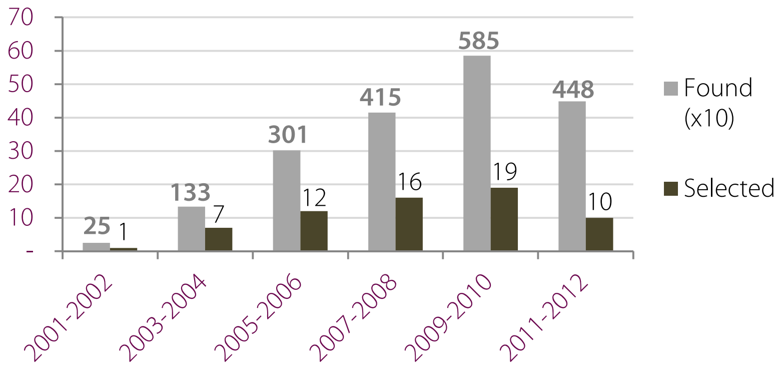


Figure 6 – List of papers found and selected by 2-year period.

Considering the 47 proposals and not the 65 individual papers, the first quality model we found specifically designed for web services was issued in 2002 [52]. Since then, we identified that many proposals have been developed based on other previous quality models, sharing many concepts and definitions. Most of the oldest quality models have been updated or enhanced by other researchers. In order to picture the evolution of such proposals, and depict the research done in quality models, we have built the genealogical tree, which is shown in Figure 7.

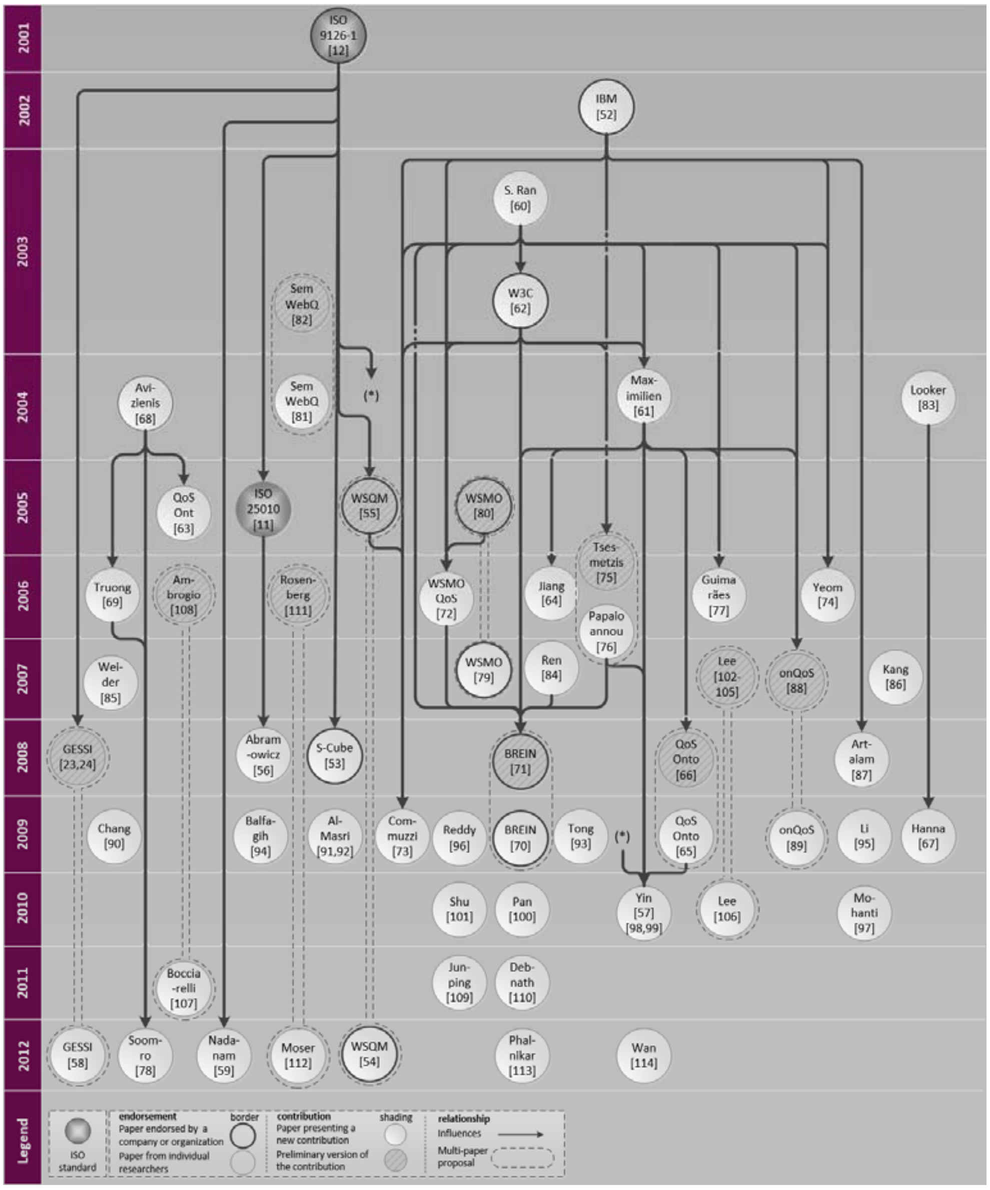


Figure 7 – Genealogical tree of quality models.

Some relevant observations follow:

- Contrary to what could be expected, most of the proposals do not take into consideration the standards ISO 9126-1 [12] or ISO/IEC 25010 [11] for the development of the quality model, with the exceptions of the S-Cube Quality Reference Model [53], WSQM [54], [55], Abramowicz et al. [56], Yin et al. [57], GESSI [23], [24], [58] and Nadanam et al. [59].

- 56% of the approaches have not considered other quality models for web services in their definition. Although this could be considered acceptable for the oldest ones, it is not so acceptable for the most recent ones. For instance, if we consider 2007 onwards, there are fewer proposals that build on top of others (13) than new proposals (20). We think that this is an indicator that a study like the systematic mapping undertook in this thesis may help researchers in the field to be more aware of the current state of the art.

- Symmetrically, 26% of the approaches have influenced the definition of other quality models. If we focus on particular proposals, the oldest quality models have had the biggest impact. The most used quality model is from S. Ran [60] which has been used to develop 8 quality models, followed by E. Maximilien and P. Singh [61] (influencing 6 quality models), and both IBM [52] and W3C QoS [62] (influencing 5 quality models each). Still a few more quality models have influenced the definition of other proposals [54]–[57], [63]–[69] . Please note that these numbers do not consider transitivity, i.e. if a quality model *A* influenced the definition of a quality model *B* and *B* influenced the definition of another quality model *C*, we have not counted A in the influencing set of *C*.

- The approach which takes more quality models into consideration for the development of its proposal is BREIN QoS Ontology [70], [71] (with 6 proposals) followed by WSMO-QoS [72], M. Comuzzi and B. Pernici [73] and Yin et al. [57] (with 4 proposals each). To a lesser extent, also Yeom et al. [74], Tsesmetzis et al. [75], [76] , E. Maximilien and P. Singh [61], Guimarães [77], and Soomro and Song [78] (with 2 proposals each). It is worth to clarify that this analysis is scoped to the relationship between the quality models in the field of web services. Other models (e.g. UML profiles) have not been considered although they have been taken into account for the development of certain quality models (e.g. S-Cube Quality Reference Model [53]).

- The approach that reflects the longest evolution (i.e. more quality models in its path) is Yin et al. [57], whose evolution path goes through 7 quality models starting from 2003.

- Last, we address endorsement of the proposals. The first quality models were developed by organizations such as IBM [52] and W3C [62]. More recent quality models such as WSMO are developed by the WSMO working group and supported by W3C [79], [80]; the OASIS group has also developed a quality model named WSQM [54], [55]. On the other hand, some quality models have been developed by European Networks, as S-Cube [53], or European Projects as BREIN [70], [71]. Figure 8 summarizes the overall results.

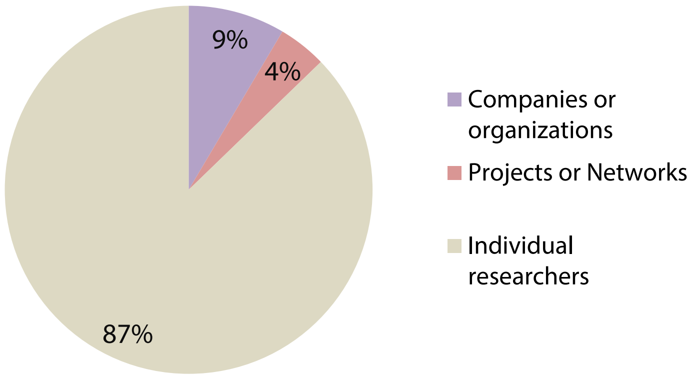


Figure 8 – Endorsement of the proposed quality models.

2.3.2 What are the characteristics of the proposed quality models?

RQ 1.2

This section answers the Research Question RQ 1.2.

To answer this question, we evaluate the structural characteristics of the proposed quality models in terms of its size (quantity) and definition coverage (quality).

- **Size**: We evaluated the size by the amount of nodes (representing quality factors) present in the quality models and its level of depth. Through this criteria, the most extensive quality models are the S-Cube Quality Reference Model [53] with 66 nodes, followed by quality model presented by GESSI [23], [24], [58] with 57 and Truong et al. [69] with 45 nodes. In all these cases, the presented quality models have a 3-level depth. In this regard, we found a clear correlation between the number of nodes and the depth level (See Figure 9). For instance, quality models presenting only between 6 and 10 nodes, were usually developed in 1-level depth (6 proposals). In the mid-range area, quality models of between 26 and 30 nodes were usually developed in a 2-level depth (9 proposals) and larger quality models (more than 30 nodes) were more often presented in a 3-level depth. On average, the proposed quality models have 24,38 nodes and a depth level of 2,23.

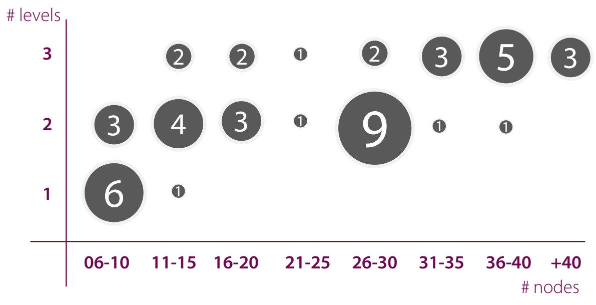


Figure 9 – Correlation map between nodes and levels.

- **Definition completeness**: A narrow majority of the presented proposals, up to 51%, have a unique and consistent definition for all the quality factors that are present in the quality model (i.e. 100% definition completeness), either by explicitly defining the quality factor in the paper or by referencing to the paper which has the definitions (See Figure 10).

The remaining proposals present problems of different scale: (1) some quality factors are not defined; (2) the quality model is based on several quality models and although they are referenced, it is not specified which is the chosen definition for each quality factor, leading to different definitions which are not consistent with each other; (3) the definitions on some quality factors are too vague or ambiguous.

Following these criteria, 15% of quality models have a definition completeness between 80% and 99% (i.e. between 80% and 99% of the quality factors are defined without any of the aforementioned problems), 15% of the quality models have a definition completeness between 60-79%, and 19% of the proposals have a definition completeness below 40%.

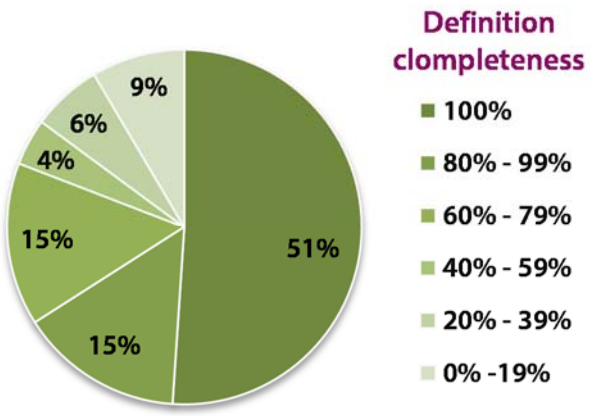


Figure 10 – Percentage of Quality models with their definition completeness.

Table 16 summarizes the most significant information about the 47 surveyed proposals in the answer of these two first research questions.

Table 16 – Characteristics of the quality models.

Proposal	Year	Develop. Method	Deph Level	Nodes	Definition completeness
IBM [52]	2002	From scratch	2	9	100%
S. Ran [60]	2003	From scratch	3	35	100%
W3C QoS [62]	2003	Based on [60]	2	30	100%
E. Maximilien and P. Singh [61]	2004	Based on [60,62]	2	26	100%
A. Avizienis et al. [68]	2004	From scratch	3	15	100%
SemWebQ [81], [82]	2003-2004	From scratch	3	15	100%
Looker et al. [83]	2004	From scratch	2	7	100%
QoSOnt [63]	2005	Based on [68]	1	6	100%
WSMO-QoS [72]	2006	Based on [80,62,60,52]	2	27	88%
S. Jiang and F. Aagesen [64]	2006	Based on [61]	2	16	81%
G. Yeom el al. [74]	2006	Based on [60,52]	3	39	100%
D.T. Tsesmetzis et al. [75], [76]	2006	Based on[62,52]	2	27	93%
Guimarães and Beatriz [77]	2006	Based on [60][61]	1	15	100%
Truong et al. [69]	2006	Based on [68]	3	45	71%
Ren et al. [84]	2007	From scratch	2	26	0%
WSMO [79], [80]	2005-2007	From scratch	2	29	66%
Wedier D. Yu et al. [85]	2007	From scratch	2	30	100%
Y. Kang [86]	2007	From scratch	1	8	100%
Abramowicz et al. [56]	2008	Based on ISO 25000	2	17	100%
N. Artaiam and T. Senivongse [87]	2008	Based on [52]	2	12	89%
S-Cube Quality Reference Model [53]	2008	Based on ISO/IEC 9126-1	3	66	100%
onQoS [88], [89]	2007-2009	Based on [60,61]	3	25	74%
BREIN QoS ontology [70], [71]	2008-2009	Based on [60,62,72,84,76]	3	40	67%
Chang et al. [90]	2009	From scratch	2	30	0%
Al-Masri and Mahmoud [91], [92]	2009	From scratch	3	19	41%
Tong et al. [93]	2009	From scratch	2	12	33%
WS-QoSOnto [65], [66]	2008-2009	Based on [61]	3	32	72%
M. Comuzzi and B. Pernici [73]	2009	Based on [55,52,60,62]	2	11	100%
S. Hanna et al. [67]	2009	Based on[83]	1	8	100%
Z. Balfagih and M.F. Hassan [94]	2009	From scratch	3	26	91%
Li and Zhou [95]	2009	From scratch	3	27	0%
Reddy [96]	2009	From scratch	3	39	77%
Mohanty et al. [97]	2010	From scratch	1	10	90%
Yin et al. [57], [98], [99]	2010	[61,65,76] and ISO/IEC 9126	3	40	20%
Z. Pan and J. Baik [100]	2010	From scratch	1	6	100%
Shu and Meina [101]	2010	From scratch	2	14	50%
Lee and Yeom [102]–[106]	2007-2010	From scratch	3	37	100%
P. Bocciarelli [107] **and A.D'Ambrogio** [108]	2006-2011	From scratch / Based on [107]	2	22	100%
Junping and Fan [109]	2011	From scratch	1	8	100%
Debnath et al. [110]	2011	From scratch	2	9	100%
Rosenberg et al.[111] **and Moser et al.**[112]	2006-2012	From scratch / Based on [111]	3	17	100%
GESSI [23],[24],[58]	2008-2012	Based on ISO/IEC 9126-1	3	57	98%
WSQM [54], [55]	2012	Based on ISO/IEC 9126-1	2	34	100%
Phalnikar and Khutade [113]	2012	From scratch	2	20	0%
Nadanam [59]	2012	Based on ISO/IEC 9126-1	3	34	100%
Soomro and Song [78]	2012	Based on [68,69]	2	29	79%
Wan Ab. Rahman et al. [114]	2012	From scractch / UML QoS Profile	2	40	27%

2.3.3 Which quality factors are the most addressed in the quality models? Which are the least addressed?

This section answers the Research Question RQ 1.3.

In this question we analyse which are the most addressed quality factors of the presented quality models. To do so, we use the standard ISO/IEC 25010 [11] as a reference framework for the comparison. This standard distinguishes eight high-level characteristics, each of them divided into several subcharacteristics. To evaluate the coverage of the different quality characteristics and subcharacteristics on each quality model, we define the criteria/values in Table 17.

Table 17 – Criteria of quality factors evaluation

Value	Definition
Y+ (Yes+)	The (sub)characteristic is explicitly defined in the quality model and contains further subdivisions.
Y (Yes)	The (sub)characteristic is explicitly defined in the quality model.
P+ (Partially +)	The (sub)characteristic is not explicitly defined, but the quality model has several quality attributes or metrics which can be classified into this (sub)characteristic.
P (Partially)	The (sub)characteristic is not explicitly defined, but the quality model has a quality attribute or metric which can be classified into this (sub)characteristic.
ND (Not defined)	The (sub)characteristic is not defined, neither its quality attributes nor metrics.

The results of this analysis at the level of the 8 characteristics are depicted in Table 18. As shown, the most addressed quality characteristic in quality models for web services is *reliability*. The importance of reliability in the field of web services is clear as most services rely on 3rd-party providers and it includes key quality attributes for selection, monitoring or adaptation. Therefore, its dominance is not surprising. The results show that 81% of the proposals explicitly cover this characteristic and the 19% remaining, although do not define the concept explicitly, cover it partially by defining some of its subcharacteristics. In other words, there is no single proposal without addressing to some extent this characteristic.

The next most important characteristics in terms of their adoption by quality models are *performance efficiency* and *security*. These are also two key characteristics for web services, due to the fact that services operate in a highly dynamic environment over the Internet. *Performance efficiency* and *security* are explicitly covered by 68% and 74% of the proposals respectively, partially in 23% and 8%, whereas proposals that do not cover them are only 9% and 17%. Remarkably, only 4,3% of the proposals (i.e. 2 out of 47) do not cover any of them.

With respect to the rest of quality characteristics, their presence in the proposed quality models decreases dramatically. None of the remaining quality characteristics are explicitly defined by more than 20% of the proposals: *maintainability* is defined in 17% of the proposals, followed by *usability* with 10%, *functional suitability* 4%, *portability* 4% and *compatibility* 0%. Although some quality models cover them partially by defining some of their subcharacteristics, their presence in quality models is very low. The clearest example is *usability*, which characteristic and its subcharacteristics are completely absent in 72% of the proposals. They are followed by *compatibility* (64%), *maintainability* (64%), *portability* (53%) and *functional suitability* (32%).

Table 18 – ISO/IEC 25010 quality characteristics coverage in the quality models.

	1. Func. suitability	2. Perf. efficiency	3. Compatibility	4. Usability	5. Reliability	6. Security	7. Maintainability	8. Portability
IBM [52]	P	Y			Y+	Y		
S. Ran [60]	P+	Y+			Y+	Y+	P	P
W3C QoS [62]	P	Y+	P		Y+	Y+		P
E. Maximilien and P. Singh [61]	P	Y+	P		Y+	Y+	P	P
A. Avizienis et al. [68]					Y+	Y+	Y	
SemWebQ [81], [82]		Y			Y+	Y		
Looker et al. [83]	P	Y			Y+	Y		
QoSOnt [63]					Y+	P+	Y	
WSMO-QoS [72]	P	Y+	P		Y+	Y+		P
S. Jiang and F. Aagesen [64]		Y			Y+	Y+		P
G. Yeom el al. [74]	P	Y+	P	P	Y+	Y+	Y+	
D.T. Tsesmetzis et al. [75], [76]	P	Y+			Y+	Y+	P	P
Guimarães and Beatriz [77]	P	P+	P		Y+	Y	P	P
Truong et al. [69]	P	Y+			Y+	Y+		
Ren et al. [84]	P	Y+			Y+	Y		
WSMO [79], [80]	P	Y			Y+	Y+		P
Wedier D. Yu et al. [85]	P	Y	P		Y+	Y+		P
Y. Kang [86]		Y			Y+	Y		P
Abramowicz et al. [56]	Y	Y	P	Y	Y	Y	Y	Y
N. Artaiam and T. Senivongse [87]		Y			Y+	Y		
S-Cube Quality Reference Model [53]	P+	Y+		P+	Y+	Y+	P	P
onQoS [88], [89]		Y+			Y+	Y+		P
BREIN QoS ontology [70], [71]	P	Y+	P		Y+	Y+	P	P

Chang et al. [90]	P	P+	P	Y	P+	P+		
Al-Masri and Mahmoud [91], [92]		Y+		Y+	Y+			
Tong et al. [93]					Y+			
WS-QoSOnto [65], [66]		Y+	P		Y+	Y+		
M. Comuzzi and B. Pernici [73]		P			P	P		
S. Hanna et al. [67]	P			P+	P			P+
Z. Balfagih and M.F. Hassan [94]		P	P	Y+	Y+	Y+	Y+	
Li and Zhou [95]	P	Y+			Y+	Y+		P
Reddy [96]	P+	Y+	P	P	P	Y+	Y+	
Mohanty et al. [97]	P	P		P	Y+			
Yin et al. [57], [98], [99]	P	Y+	P		Y+	Y+	P	P
Z. Pan and J. Baik [100]		P+			Y+			
Shu and Meina [101]	P	Y+			Y+	Y+		
Lee and Yeom [102]–[106]	P	P+	P	P	P+	Y+	P+	
P. Bocciarelli [107], A. D'Ambrogio [108]		P+			Y+	P		
Junping and Fan [109]	P	P+			Y	Y		
Debnath et al. [110]		Y+			Y+			
Rosenberg et al. [111], Moser et al. [112]	P	Y+			P+			P
GESSI [23], [24], [58]	Y+	Y+	P+	Y+	Y+	Y+	Y+	Y+
WSQM [54], [55]	P	P+	P	P	P+	Y+	Y+	
Phalnikar and Khutade [113]	P	Y+			P	Y+		P
Nadanam [59]	P	P+	P	P	Y+		P+	P
Soomro and Song [78]	P	Y+			Y+	Y+		P
Wan Ab. Rahman et al. [114]	P	Y+			P+	Y+		
Y+	**2%**	**49%**	**0%**	**6%**	**77%**	**55%**	**11%**	**2%**
Y	**2%**	**19%**	**0%**	**4%**	**4%**	**19%**	**6%**	**2%**
P+	**6%**	**17%**	**2%**	**4%**	**11%**	**4%**	**4%**	**2%**
P	**57%**	**6%**	**34%**	**13%**	**8%**	**4%**	**15%**	**40%**
(ND)	**32%**	**9%**	**64%**	**72%**	**0%**	**17%**	**64%**	**53%**

We describe below in further detail the completeness and coverage of subcharacteristics of each of the quality characteristics. We present here only the overall results, detailed results for each quality model can be retrieved at [47].

- **Functional suitability** is partially covered by 63% of quality model proposals, mainly due to the *functional correctness* subcharacteristic (see Table 19), which includes *accuracy* [60], [62], [72], [112] and *precision* [53], [84], [90].

Table 19 – Functional suitability coverage in the quality models.

	1. Functional suitability	1.1 Functional completeness	1.2 Functional correctness	1.3 Functional appropiatenes
Y+	2%	0%	32%	0%
Y	2%	4%	30%	4%
P+	6%	0%	2%	0%
P	57%	0%	4%	2%
(ND)	32%	96%	32%	94%

- **Performance** is covered by most of the quality models. However, nearly none of its subcharacteristics are explicitly defined as in ISO/IEC 25010. Instead, the quality models propose quality attributes and metrics that can be classified into these subcharacteristics, as *response time*, *throughput* and *memory used*. The reason for such a circumstance is that the attributes and metrics regarding performance are usually defined as direct subdivisions of the characteristic, without the subcharacteristic granularity (see Table 20). In this regard, quality attributes belonging to *time behaviour* are the most widespread, especially *response time* [60]–[62], [72] and *throughput* [52], [60]–[62].

Table 20 – Performance coverage in the quality models.

	2.Performance efficiency	2.1 Time behaviour	2.2 Resource utilization	2.3 Capacity
Y+	49%	2%	2%	2%
Y	19%	2%	2%	0%
P+	17%	68%	6%	19%
P	6%	15%	6%	45%
(ND)	9%	13%	83%	34%

- **Compatibility** is partially covered by 36% of the quality models, mainly due to the *interoperability* subcharacteristic. Co-existence is only present in 2% of the quality models, and no decomposition in finer-grained quality attributes is present (see Table 21).

Table 21 – Compatibility coverage in the quality models.

	3. Compatibility	3.1 Co-existence	3.2 Interopera-bility
Y+	0%	0%	2%
Y	0%	2%	34%
P+	2%	0%	0%
P	34%	0%	0%
(ND)	64%	98%	64%

- **Usability** is mostly ignored in the quality models. It is explicitly defined only in 10% of the quality models and partially covered in 17% of quality models. This partial coverage is due to the *appropriate recognizability, operability and learnability* subcharacteristics. *Appropriate recognizability* includes the quality attributes *documentation* [67], [91], [97] and *discoverability* [94]. *Operability* includes, as the main quality attribute, *controllability* [54], [55], [74]. On the contrary, *learnability* has no decomposition (see Table 22).

Table 22 – Usability coverage in the quality models.

	4. Usability	4.1 Appropriate recognizability	4.2 Learnability	4.3 Operability	4.4 User error protection	4.5 User interfa-ce aesthetics	4.6 Accessi-bility
Y+	6%	6%	0%	4%	0%	0%	0%
Y	4%	2%	4%	2%	0%	0%	0%
P+	4%	0%	0%	0%	0%	0%	0%
P	13%	6%	0%	2%	0%	0%	0%
(ND)	72%	85%	95%	91%	100%	100%	100%

▶ **Reliability** is the most covered characteristic by existing quality models. Regarding its subcharacteristics, the most covered is *availability*, which is remarkably defined in 94% of the proposed quality models. The other subcharacteristics have also some arguably good coverage with the exception of *maturity* which is only covered in 2% and partially covered in 11% of quality models (see Table 23). The partial coverage of *maturity* comes from the *consistency* quality attribute [57], [61], [65], [70], [71].

Table 23 – Reliability coverage in the quality models.

	5. Reliability	5.1 Maturity	5.2 Availability	5.3 Fault tolerance	5.4 Recovera-bility
Y+	77%	0%	28%	13%	23%
Y	4%	2%	66%	26%	6%
P+	11%	0%	0%	0%	2%
P	9%	11%	0%	0%	13%
(ND)	0%	87%	6%	62%	55%

▶ **Security** is explicitly defined by 74% of quality models. Regarding its subcharacteristics, all of them have been explicitly defined with similar percentages, which shows a clear consolidation on its decomposition. The main difference appears in *confidentiality* (predominance of Y+ over Y) because most quality models present an encryption-related quality attribute [54], [57], [61], [65] which is a concept that clearly falls into it (see Table 24).

Table 24 – Security coverage in the quality models.

	6. Security	6.1 Confiden-tiality	6.2 Integrity	6.3 Non-repudiation	6.4 Accounta-bility	6.5 Authenticity
Y+	55%	36%	0%	0%	19%	0%
Y	19%	9%	43%	43%	28%	45%
P+	4%	4%	0%	0%	0%	0%
P	4%	11%	0%	0%	0%	0%
(ND)	17%	40%	57%	57%	53%	55%

- **Maintainability** is only explicitly defined in 17% of the quality models, and partially supported in 19%. The partial coverage is mainly due to the *modifiability* subcharacteristic which is also partially covered by the *stability/change cycle* quality attribute [53], [60], [61], [75] (see Table 25).

Table 25 – Maintainability coverage in the quality models.

	7. Maintaina-bility	7.1 Modularity	7.2 Reusability	7.3 Analysa-bility	7.4 Modifia-bility	7.5 Testability
Y+	11%	0%	2%	0%	2%	0%
Y	6%	11%	2%	2%	0%	2%
P+	4%	0%	0%	9%	4%	0%
P	15%	0%	0%	0%	13%	0%
(ND)	64%	89%	96%	89%	81%	98%

- **Portability** is partially covered in nearly half of the presented quality models due to the *adaptability* subcharacteristic, which is partially covered by the *scalability* quality attribute that appears in a few proposals [60]–[62], [72] (see Table 26).

Table 26 – Portability coverage in the quality models.

	8. Portability	8.1 Adaptability	8.2 Installability	8.3 Replacea-bility
Y+	2%	2%	0%	0%
Y	2%	2%	2%	4%
P+	2%	0%	0%	0%
P	40%	40%	0%	0%
(ND)	53%	55%	98%	96%

- **Non-technical** characteristics are not present in ISO25010, because the standard focuses on technical aspects of the software. Nevertheless, since they represent an important aspect when evaluating web services for their prospective use, we have also considered them. We have taken as the basis for comparison the extension of ISO-9126 with non-technical quality characteristics presented in [115]. We adapted the subcharacteristics to the context of web services, therefore the non-technical subcharacteristics we propose are *regulatory* (i.e. related to laws and standards), *economic* (i.e. related to costs and penalties) and *reputation and recognition*. Analysing the results, *economic* is the most used non-technical subcharacteristic, being *cost* its main quality attribute [60], [64], [72], [81], [82] (see Table 27).

Table 27 – Non-technical coverage in the quality models.

	9. Non-technical	9.1 Regulatory	9.2 Economic	9.3 Reputation & Recognition
Y+	2%	4%	17%	4%
Y	0%	11%	0%	0%
P+	43%	13%	13%	9%
P	30%	9%	30%	26%
(ND)	26%	64%	40%	62%

Some relevant observations follow:

Unbalance between client-based and provider-based quality attributes: Client-based quality attributes are those which are measured from the client's perspective, whereas provider-based quality attributes are those measured from the provider's perspective. It is worth to remark the unbalance between client-based and provider-based quality attributes, being the latter less covered. For instance, *response time* (which is from the client's perspective) is present in 83% of the quality models, whereas *execution time* (which it is from the provider's perspective) is present in 23%

of quality models. This has an impact on the coverage of certain subcharacteristics. For instance *resource utilization* (which clearly deals with the provider's perspective) is only partially covered in 17% of the quality models whereas *time behaviour* is covered in 87%.

Lack of some subcharacteristics support: We have identified some ISO/IEC 25010 quality subcharacteristics that, to our understanding, have not been addressed enough in current quality models. For instance, the lack of the *modularity* and *reusability* subcharacteristics contrasts with the fact that they are two key principles in service oriented architecture [5]. On the other hand, *appropriate recognisability* and *functional appropriateness* are remarkably subcharacteristics for a proper web service discovery that have not been widely addressed. Similarly, other relevant characteristics that have not been much addressed are the *usability* and *maintainability* subcharacteristics. We argue that some of these results may be due to the unbalance between client-based and provider-based quality attributes.

Terminological inconsistencies with the standard: We found some terminological inconsistencies with the standard ISO25010. It is remarkable that many quality models have a quality attribute named *accessibility*, but they refer to a completely different concept. In most quality models, it refers to the availability of the system under certain circumstances (e.g. limit of concurrent users) [52], [62], [72], [81], [82] whereas in ISO25010 it refers to the accessibility for people with disabilities. Also some quality models have a quality attribute named *capacity*, but its definition is limited to the maximum throughput [60]–[62], [72], which is a subset of the definition of *capacity* in ISO25010.

2.3.4 What are the most consolidated quality factors?

RQ 1.4

This section answers the Research Question RQ 1.4.

In this question we aim at identifying the most consolidated definitions for quality factors in the surveyed quality models. We restrict the analysis to those quality attributes that have appeared most frequently in the surveyed quality models. We have determined a threshold value of 30% (i.e. we are considering those quality attributes that appear at least in 30% of the surveyed quality models). Such threshold value is arbitrary as we found no procedure in the literature to define it. Nevertheless, after several iterations, we considered it an appropriate value since it yielded as result 19 quality attributes, which is arguably a reasonable number for this kind of analysis. Furthermore, we observed that for most of quality attributes below this threshold, the number of terminology discrepancies is higher (details are presented in [47]). On the other hand, a higher threshold can be easily analysed since the usage percentage is shown in Table 28.

In order to obtain accurate and significant results, the analysis was based on the definition of the quality attributes. That is, we established a mapping between the different quality attributes that shared the same or nearly equivalent definitions. Nevertheless, as we need to use a name for each of them, we use the term that is more used along the different quality models. The list of terms and definitions is depicted in Table 28.

In the first column, we depict the quality attributes classified into the ISO/IEC 25010 quality subcharacteristic. In the second column, we provide the most consolidated definition. The percentage of quality models that have such quality attribute based on its definition (i.e. quality models that have another name for such quality attribute do also count in the percentage) is shown in the third column. Finally, from the quality models that have such quality attribute based on the definition, we depict the percentage of them that use the most common name.

For readability, the subcharacteristics are grouped into quality characteristics (*functional suitability*, etc.), see rows in grey in the table.

Table 28 – Most consolidated quality attributes and their definitions.

ISO/IEC 25010 subcharacteristics ⇨	Quality attribute	Definition	Definition usage %	Name usage %
		Functional suitability		
Functional ⇨ correctness	**Accuracy**	Error rate produced by the service [26][28].	62%	42%
		Performance efficiency		
Capacity ⇨	**Accessibility**	Degree the service is capable of serving a web service request [53][16].	47%	100%
Capacity ⇨	**Capacity**[1]	Limit of concurrent requests for guaranteed performance [26][55].	51%	65%
Time behaviour ⇨	**Response Time**	Time to complete a WS request (from a client perspective) [26][28].	83%	73%
Time behaviour ⇨	**Throughput**	Number of web service requests served at a given time period [28][16].	60%	100%
		Compatibility		
(subcharacteristic)	**Interoperability**	Ease with which a consumer application or agent interoperates with a service [27].	36%	100%
		Reliability		
Availability ⇨	**Availability**[1]	Probability that the service can respond to consumer requests [51][27].	94%	98%
Recoverability ⇨	**MTTR**	Mean Time to Repair [30][27].	30%	86%
Fault tolerance ⇨	**Robustness**	Degree to which a service can function correctly in the presence of invalid, incomplete or conflicting inputs [26][28].	38%	94%
		Security		
(subcharacteristic)	**Authentication**	Measure of how the service authenticates principals (users or other services) who can access service and data [26].	45%	95%
(subcharacteristic)	**Confidentiality**	Measure of how the service treats the data, so that only authorized principals can access or modify the data [26].	45%	86%
(subcharacteristic)	**Integrity**	Absence of improper system state alterations including accidental or malicious alternation or removal of information [8][29].	43%	100%
(subcharacteristic)	**Non-repudiation**	A principal cannot deny requesting a service or data after the fact [26][28].	43%	100%
Accountability ⇨	**Traceability and auditability**	Whether it is possible to trace the history of a service when a request was serviced [26].	38%	95%
Confidentiality ⇨	**Authorization**	Measure of how the service authorizes principals so that only them can access the protected services [26].	40%	67%
Confidentiality ⇨	**Encryption**	Measure of how the service encrypts data. Type and strength of encryption technology used for storage and messaging [26].	40%	100%
		Portability		
Adaptability ⇨	**Scalability**	Capability of increasing the computing capacity of service provider's system and system's ability to process more operations in a given period[26][8].	45%	100%
		Non-technical		
Economic ⇨	**Cost**	Measure of the cost involved in requesting the service [26][52].	60%	55%
Reputation ⇨ and recognition	**Reputation**	It is a measure of service trustworthiness. It depends on end user's experiences of using the service [8][40].	38%	73%

(1): The quality attribute has the same name as the ISO subcharacteristic, but it refers to a different concept, as the scope of its definition is more delimited

With respect to the list of quality attributes, it is worth to remark two issues that we have identified:

- The *interoperability* and *security* subcharacteristics are considered quality attributes in most quality models, although they are defined as subcharacteristics in ISO/IEC 25010.
- Similarly, *capacity* and *availability* are defined as quality attributes in most quality models, whereas in ISO/IEC 25010, the same terms are considered subcharacteristics. Nevertheless, in these cases, the definitions in the proposed quality models do not refer exactly to the same concept as in ISO/IEC 25010. *Availability*, in most quality models, refer to the probability that the service can respond to consumer requests [61], [85], whereas in ISO/IEC 25010 it is a subcharacteristics that may include other quality attributes to measure the degree to which the service is operational (e.g. *total uptime*). Similarly, *capacity*, as defined in the proposed quality models, is limited to the *maximum throughput* [60]–[62], [72].

Regarding the results of the analysis, the most widespread quality attribute we found is by far *availability*, which is used in 94% of the quality models, followed by *response time* (83%), *accuracy* (62%), *throughput* (60%) and *cost* (60%).

Regarding their names, there is a wide consensus for the terms to use over these quality attributes. All of them have a common terminology in the majority (i.e. more than a half) of the quality models. Moreover, most of these quality attributes share the same name in over 90% of the quality models. Nevertheless we can distinguish some discrepancies on the terms to use in the following quality attributes: *accuracy* (other names are, e.g. *error rate* [57], [72], *successability* [54], [74]), *capacity* (e.g. *maximum throughput* [74], [90]), *response time* (e.g. *latency* [52], [83]), *authorization* (e.g. *service access control* [107], [108]), *confidentiality* (e.g. *privacy* [54], [90]), *cost* (e.g. *price* [54], [91]) and *reputation* (e.g. *trust* [57], [80]).

The complete list of quality attributes mapped into the ISO 25010 standard can be accessed to [47].

2.4 Threats to validity

In this section we discuss all the aspects during the research process that might lead to a threat to validity, as well as the actions we have performed in order to mitigate those risks. In this regard, we identified and evaluated the threats following the common classification of construct validity, internal validity, external validity and conclusion validity.

2.4.1 Construct validity

Construct validity refers to the validity threats with respect to the observations performed in the study and if they really represent what is being investigated. At this respect, one of the inherent threats to any systematic mapping is that it does not guarantee the inclusion of all the relevant works in the field. This might be caused by several reasons, for instance, a relevant work may not be indexed on the selected database, the keywords used in the title or abstract of a relevant work do not match with the keywords of the search, etc. This threat was mitigated by combining several databases (ISI WoS, IEEE Xplore and ACM DL) and manual searches to selected journals and conferences, as well as studying accurately the keywords to use. However, the issue may not be solved since the problem goes beyond an accurate protocol and concerns also issues related to the paper (e.g. inaccurate abstracts). To mitigate this risk, we included a final step of snowballing, as described in Section 2.2.5, assuring that the quality models that have had the biggest impact in the field were also included. Also, the identification of some basic sources was helpful since, as summarized in Table 13, some of the conferences that we considered as the usual venues for the topic of the systematic mapping, had at least one edition not indexed in the selected databases, but still we handled them manually.

2.4.2 Internal validity

Internal validity refers to the validity of the analysis performed. Concerning this aspect, we have identified two major threats.

(1) Not all the quality models define all the presented quality factors accurately. As shown in Section 2.3.2, some definitions are ambiguous, inconsistent or simply absent. This situation poses a challenge when analysing the coverage of the characteristics and subcharacteristics, and a subsequent threat to validity. To overcome such threat, we examined the list of not-well defined quality factors and took the following strategy: for those factors lacking of a definition, if there was a clear consensus of the definition in the state of the art, or the name of the quality factor was self-explanatory (e.g. *total memory consumption* [70]), we recognized the meaning of the quality factor despite the lack of definition. For those factors whose definition was ambiguous or inconsistent, if such ambiguity or inconsistency did not affect the categorization of the quality factor, we included that quality factor in the analysis. Otherwise, the quality factor was discarded from the analysis. Following this criterion, 35 quality factors were discarded from more than 1000 quality factors analysed (< 3,5%).

(2) As each quality model implements its own hierarchical structure, we decided to map these structures into a reference quality model structure. To this aim, we used the standard ISO/IEC 25010 as it is described in [11], and decided the mapping of the nodes from each quality model to this standard. This mapping can be considered by itself a threat to validity. To mitigate this risk, the mapping was discussed and analysed closely by all three authors of the paper that presented this work [22]. Nevertheless, some nodes could not be clearly mapped into a subcharacteristic of ISO/IEC 25010. This issue is present only in just 14 nodes out of 134 (<10,5 %).

2.4.3 External validity

External validity is concerned about the extrapolation of the results from a particular scenario to the general case. Since our results are scoped in quality models in web services and we do not attempt to generalize conclusions beyond this scope, this validity threat does not apply.

2.4.4 Conclusion validity

Conclusion validity is concerned about whether the research performed is reproducible by other researchers with similar results. In this regard, we have explicitly described all the steps performed in the systematic mapping by detailing the procedure as defined in the systematic mapping protocol. Furthermore, the list of papers found and selected on each step is included in [47].

Part II

How to monitor

Study on monitoring systems

CHAPTER 3 State of the Art

"Think before you speak. Read before you think" – Fran Lebowitz.

To develop the State of the Art, we have also conducted an SLR following the guidelines of Kitchenham [46]. A summary of such guidelines are described in section 2.1.

3.1 Planning the review

3.1.1 The research question

The first step of the SLR is the formulation of the research question. In this section, we aim at the research question RQ 2.1, which was first described in Section 1.3 and focus on the State of the Art of monitoring frameworks. The research question is constructed following the PICO structure. The Population, Intervention and Outcome are identified as shown in Table 29.

Table 29 – Research question 2.1: Population, Intervention and Outcome.

RQ 2.1	In the field of **web services** (P), to which degree the proposed **monitoring frameworks** (I) **support the whole SBS lifecycle** (O)?

3.1.2 Bibliographic sources

The search process can be automatic through bibliographic databases or manual by searching on specific journals and conferences. Kitchenham et al. analysed in [48] the advantages and drawbacks of both approaches through a case study. One of the conclusions from their results was that "broad searches find more papers than restricted searches, but the papers may be of poor quality. Researchers undertaking SLRs may be justified in using targeted manual searches if they intend to omit low quality papers" [48].

For the development of this SLR we targeted manual searches in selected journals and conferences. We prioritized the quality of the contributions rather than quantity as the goal of this state of the art is not to gather all the work available in the literature, but to report and analyse in a systematic manner the list of relevant and high quality contributions similar to our work.

The list of venues was obtained from the top ranked journals and conferences in services, software engineering and information systems engineering based on their JCR Impact Factor and CORE-A status respectively, using the same criteria as in the Systematic mapping of Part I of this thesis. Details of such list and its obtention are described in Section 2.2.3. The list of journals and conferences are summarized here in Table 30.

Table 30 – List of journals and conferences used in the systematic mapping.

Journals	ACM - Computing Surveys, ACM – TOIT, ACM - TOSEM, ACM - TWEB, Elsevier - Advances in Engineering Software, Elsevier - IST, Elsevier - JSS, Elsevier – JWS, IEEE - Computer, IEEE - Internet Computing, IEEE - Software, IEEE - TSE, IGI global JWSR, Springer – SQJ and Springer - WWW
Conferences	ASE, CAiSE, E-Science, FSE, GRID, ICSE, ICSOC, ICWS, ISSTA, SCC, WISE, WWW

3.1.3 Keywords used

The keywords used for the search were retrieved from the PICO terms of the research questions. Particularly we extracted them from the Population and Intervention [46] of the research question. From each term of the Population and Intervention of the research questions, we identified a set of variants (see Table 31).

Table 31 – Keywords related to the Population and Intervention.

	Terms without wildcards	Terms with wildcards
Population: web service	"web service", "web services", service, services	service*
Intervention: monitoring	monitoring, monitor, monitors	monitor*

The resulting query is obtained by combining the Population and Intervention terms, which using wildcards are simplified to **(service* AND monitor*)**. These terms have been applied in the search to the title, abstract and keywords of the papers.

3.1.4 Selection criteria

The search was limited to the last 5 years, gathering hence the most updated monitoring solutions with respect to the latest standards. The selection criteria to filter and select the proposals are as follows:

- **Selection by title:** Documents whose scope is clearly unrelated to monitoring web services were removed.
- **Selection by abstract:** At this stage, we discarded all those works that although being related to monitoring web services, did not present a monitor as contribution of the paper.
- **Selection by full paper (fast reading):** At this point, we removed the papers which did not accomplish properly the following inclusion criteria:

(1) presenting a monitoring framework as one of the contributions of the paper; (2) defining explicitly the monitoring framework.

3.2 Results of the review

RQ 2.1

This section answers the Research Question RQ 2.1.

3.2.1 Search process

By applying the defined search protocol (see Figure 11), we found 233 papers covering the search criteria. 53 papers were discarded by title and 127 papers were discarded by abstract, leading to 53 papers. We discarded then 35 papers by fast read, resulting in 18 selected papers. We then identified that those 18 papers presented 15 different approaches. The complete list of papers and the filtering process can be checked at [116].

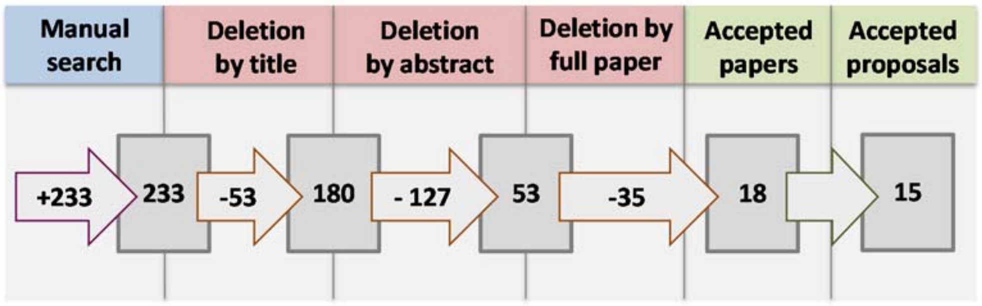


Figure 11 – Selection process of the SLR.

3.2.2 Qualitative analysis definition

In this subsection, we define the parameters for the qualitative analysis of the retrieved proposals. The list of parameters was defined in basis of the requirements for the different activities of the SBS lifecycle.

The requirements were obtained through the following elicitation techniques: on the one hand, we identified from the literature the common requirements for each activity presented. On the other hand, we conducted meetings and interviews with members of different research groups who work on these activities[4]. Such requirements were later validated through the implementation of a monitoring solution that fulfilled the needs of these research groups for service selection [21][22], service deployment [28][29][30], quality assessment [27][39] and service adaptation [34][35][36][37][45].

Firstly, we present the high-level requirements that are common along the different activities, and hence apply to the whole service lifecycle; and then we describe the specific requirements for the particular needs of each activity.

Core requirements along the SBS lifecycle: SBS may incorporate web services implemented in different languages (e.g. Java, BPEL, .NET, etc.) and executed in different engines (e.g. Axis2, Websphere, Glassfish, etc.), which might interact with different messaging protocols (e.g. SOAP, REST), leading to an heterogeneous system combining different technologies [117]. Hence, the monitor shall be capable of monitoring different types of web services, regardless of their technology or infrastructure details (R1.1). As SBS include different web services for different purposes, the list of quality attributes the monitor is able to handle shall be extensible

[4] These collaborations were made in the context of the S-Cube FP7 Network of Excellence, http://www.s-cube-network.eu/.

as to meet the end user's needs, including domain specific quality attributes [112] (R1.2). On the other hand, requirements and quality attributes to monitor might change, new web services may emerge, adaptations might be performed, etc. The monitor shall be dynamically reconfigurable to cope with such dynamicity of an SBS (R1.3). Finally, the monitor should be easily interoperable with the different tools that support the aforementioned activities in the SBS lifecycle (R1.4).

Requirements on service selection: Service selection frameworks discover and rank web services registered in a repository. These frameworks provide a ranked list of web services that fulfils not only the functional requirements of the users, but also their non-functional requirements. These non-functional requirements are expressed in terms of conditions over quality metrics (e.g. "response time < 200 ms AND availability > 90%").

To obtain the values of such quality metrics, two approaches have been proposed in the literature. On the one hand, some approaches require a distributed monitoring system in which the services are being monitored during its execution [118][119] (R2.1). On the other hand, in other approaches, it is the same framework that requires a monitor to execute periodically the service to obtain the QoS [120][26] (R2.2).

Requirements on SBS deployment: The QoS of the SBS and its web services is strongly affected by the QoS of the underlying infrastructure where they are deployed [121]. The deployment of an SBS includes the composition, internal web services and their resources. Cloud computing is emerging as a leading solution to deploy an SBS. According to Gartner, by 2017, over 50% of large Software as a Service (SaaS) providers will offer an integrated Platform as a Service (PaaS) in the cloud [122]. In the field of cloud computing, there are frameworks that select the cloud infrastructure that better fulfils the requirements when allocating the resources for the SBS to be deployed [123][124]. As a primary requirement, these frameworks require of monitoring solutions to retrieve the QoS at the infrastructure level [28][29][30] (R3.1).

On the other hand, as the cost of PaaS is usually linked to its usage and consumption, the monitor is required to minimize the consumption of resources to lower operating costs [125][30] (R3.2).

Requirements on quality assessment: During the execution of the SBS, the QoS of the constituent web services is dynamic and highly changing. The primary goal for monitoring in this phase is to ensure that the dynamic quality attributes meets at runtime the agreed SLA, and hence, being able to configure automatically the monitor from SLAs is a key aspect in this activity [44] (R4.1). Moreover, the monitor shall be able to recognize different SLA notations (R4.2).

Since the QoS stated in the SLA refers explicitly to the interaction between the service client and the service, the monitored QoS shall be from the real execution of the involved parties [101] (R4.2).

Requirements on SBS adaptation: In self-adaptive SBS, when the QoS of a web service does not meet the required level objectives, an adaptation action is triggered to restore the QoS. We distinguish between proactive adaptation (the system adapts before the malfunction occurs) and reactive adaptation (the system adapts when the malfunction has already occurred). Proactive adaptation is usually accomplished by using a monitor that periodically executes the service to identify any violation before the user experiences the malfunction [121] (R5.1), whereas reactive adaptation is accomplished by monitoring the real execution of the service client [126][127] (R5.2).

A brief summary of these requirements is depicted in Table 32. It must be taken into account that the list is not intended to provide a comprehensive set of requirements for particular solutions, as each one differs on their own specific needs. Instead, the list provides a framework of understanding in the main requirements that a monitor should achieve to support each activity.

Table 32 – Qualitative analysis criteria.

ID	Question
1. Along the lifecycle	
R1.1	The monitor shall be capable of monitoring different types of web services, regardless of their technology or infrastructure details.
R1.2	The monitor shall be extensible to monitor new quality attributes.
R1.3	The monitor shall be dynamically configurable.
R1.4	The monitor shall be interoperable with the different required components to support the activity of the SBS lifecycle.
2. Service selection	
R2.1	The monitor shall be able to passively monitor services in a distributed environment.
R2.2	The monitor shall be able to actively test services periodically.
3. SBS Deployment	
R3.1	The monitor shall retrieve the values of quality attributes at the infrastructure level.
R3.2	The monitor shall minimize the number of resources consumed.
4. Quality Assurance	
R4.1	The monitor shall be automatically configurable from SLAs.
R4.2	The monitor shall be able to recognize different SLA notations.
R4.3	The monitor shall retrieve the measurements from the real usage of web service clients.
5. SBS adaptation	
R5.1	The monitor shall be able to identify QoS violations proactively.
R5.2	The monitor shall be able to identify QoS violations reactively.

3.2.3 Qualitative analysis results

In this subsection, we analyse to which degree the retrieved papers fulfil the requirements identified in the previous subsection, and subsequently their suitability to support the different activities of the SBS lifecycle.

For each proposal, we have evaluated the fulfilment of each requirement. The different possible values can be either satisfied (tick figure), unsatisfied (cross figure), or in the case it was not clearly described, unknown (question mark).

We have also checked if the monitoring framework has been validated through the inclusion of the monitor in each of the activities presented (depicted with the same symbols in the rows labelled as 'Val.').

The results on the fulfilment of the requirements are summarized in Table 33.

Table 33 – Fulfilment of the requirements in the related work.

	O. Moser [128][129]	**SECMOL-WSCol**[130] [131]	**Dynamo&Astro** [126][127]	**K. Lin** [132]	**M. Comuzzi** [133]	**G. Ortiz** [134]	**L. Fei** [135]	**K. Mahbub** [136]	**M. Psiuk** [137]	**Q. Wang** [138]	**A. Benharref**[139]	**F. Raimondi** [140]	**G. Katsaros** [141]	**MELA** [142]	**CHOReOS monitor** [143]	**SALMon**
1. Along the lifecycle																
R1.1	✗	✗	✗	✓	✓	✓	✗	✗	✓	✗	✗	✗	✗	✓	✓	✓
R1.2	✓	✓	✓	✓	✓	✗	✓	✓	✓	✓	✓	✗	✓	✓	✓	✓
R1.3	✓	✓	✓	✓	✓	✓	✓	?	✓	✓	✗	✗	✓	✓	✓	✓
R1.4	*They have been proven compatible with their respective frameworks*															✓
2. Service selection																
R2.1	✓	✓	✓	✓	✓	✓	✓	✗	✓	✓	✓	✓	✗	✓	✓	✓
R2.2	✗	✓	✓	✗	✗	✗	✗	✓	✓	✓	✗	✗	✓	?	✗	✓
Val.	✗	✗	✗	✗	✗	✗	✗	✓	✗	✗	✗	✗	✗	✗	✗	✓
3. SBS Deployment																
R3.1	✗	✗	✓	✓	✓	✗	✗	✗	✗	✓	✗	✗	✓	✓	✓	✓
R3.2	✓	✗	✗	✗	✗	✗	✓	✗	✓	✗	✗	✗	✗	✗	✗	✓
Val.	✗	✗	✓	✓	✗	✗	✗	✗	✗	✗	✗	✗	✓	✓	✓	✓

4. Quality Assurance																
R4.1	✗	✓	✗	✗	✓	✗	✗	✗	✗	✗	✗	✓	✗	✗	✓	✓
R4.2	✓	✓	✓	✓	✓	✓	✓	✗	✓	✓	✓	✓	✗	✗	✗	✓
Val.	✗	✓	✗	✗	✓	✓	✗	✗	✓	✓	✓	✓	✗	✗	✓	✓
5. SBS adaptation																
R5.1	✗	✓	✓	✗	✗	✗	✗	✓	✓	✓	✗	✗	✓	✗	✗	✓
R5.2	✓	✓	✓	✓	✓	✓	✓	✗	✓	✓	✓	✓	✗	✓	✓	✓
Val.	✓	✓	✓	✓	✗	✗	✓	✓	✗	✓	✗	✗	✗	✓	✓	✓

▶ **Requirements along the lifecycle:** Most of the presented approaches are limited to specific web service technologies and cannot deal with any type of service (R1.1). Most approaches are limited to service compositions written in WS-BPEL [128][129][130][131][126][127], SOAP-based web services [135][136][139], RESTful [141], or services implemented in Axis/Axis2 [138][140]. These technological limitations constrain the applicability of the monitors to a narrow set of scenarios. On the contrary, there are some approaches that provide a solution not attached to a particular technology [132][133][134][137][142][143]. Regarding the quality attributes, most of the approaches are extensible to monitor new quality attributes (R1.2) with a few exceptions. Ortiz et al. [134] is only able to monitor a set of predefined metrics and lacks of important ones (e.g. availability) and Raimondi et al. [140] can only monitor time-related metrics. Finally, most of the monitors are dynamically configurable (R1.3), with the exception of Benharref [139] and Raimondi [140], which cannot be reconfigured at runtime.

▶ **Service selection**: All of the aforementioned monitoring frameworks can monitor the services either passively (R2.1), or actively using online testing (R2.2). However, only SECMOL-WSCol [130][131], Dynamo&Astro [126][127], M. Psiuk [137] and Q. Wang [138] are able to combine both approaches. This ability to combine both approaches allows the monitoring framework to be used in different service selection frameworks. Nevertheless, none of these approaches has been used specifically in service selection scenarios to validate its adequacy in a service selection framework. Only

Mahbub et al. [136] has been used in service selection, although just for SOAP-based passive monitoring.

▶ **SBS deployment**: Only a few approaches are able to monitor quality attributes at the infrastructure level (R3.1). However, they either require to plug an infrastructure monitoring engine [126][127][133][141][142], which requires more resources, or they directly interact with the operating system commands [132][138], and hence are limited to a specific operating system. None of these approaches apply techniques to minimize the consumption of resources (R.3.2). Regarding validation, only Dynamo&Astro [126][127], G. Katsaros [141] and K. Lin [132] have been used for SBS deployment.

▶ **Quality assurance**: Most of the approaches are able to monitor the QoS from the real usage of web service clients (R4.3) but only a few of them are able to automatically configure the monitor from an SLA document (R4.1). Moreover, the automated configuration of monitors presented are designed for a specific language, such as WS-Agreement [130][131][133] or SLAng [140] and are not extensible to other SLA notations (R4.2). Interestingly enough, a vast number of monitors have been validated for quality assurance [130][131][133][134][137][138][139][140][143], although with the aforementioned limitations.

▶ **SBS adaptation**: All of the aforementioned monitoring frameworks can either support proactive adaptation (R5.1) or reactive adaptation (R5.2). However, only SECMOL-WSCol[130][131], Dynamo&Astro [126][127] and Q. Wang [138] are able to support both types of adaptation approaches. Most of the monitors in the literature have been integrated into an SBS adaptation framework [128][129][130][131][126][127][132][135][136][142][143].

To conclude, all of the aforementioned monitoring frameworks support (partially) some of the activities, but none of them satisfactorily cover the whole SBS lifecycle, which requires a flexible approach to support the different activities in a general and extensible way. Therefore, we envision the need of a new approach that fulfils the requirements needed for all the activities that embrace the SBS lifecycle in a generic and extensible manner.

CHAPTER 4 The proposed monitoring framework

"*The most exciting phrase to hear in science, the one that heralds new discoveries, is not 'Eureka!' but 'That's funny...'*" – Isaac Asimov.

This section presents the insights of the framework, showing the monitoring framework designed to accomplish the requirements identified in the previous chapter. The framework, named SALMon[5], has been developed following an incremental and iterative approach. Nevertheless, we illustrate the results of such development in a linear way for the sake of readability.

4.1 SALMon's features

RQ 2.2

This section answers the Research Question RQ 2.2.

SALMon has been implemented with a set of features designed to accomplish the different requirements identified on each activity. We describe these features below.

[5] SALMon comes from SLA Monitoring with a swap of two letters to make it easier to remember.

4.1.1 Combination of model-based and invocation-based strategies

Quality assurance requires configuring the monitor automatically from an SLA specification (R4.1). However, other activities do not handle SLAs to configure the monitor. To this respect, the monitor must be able to combine two strategies to handle its configuration. On the one hand, it should be able to be configured automatically from an SLA, on the other hand, it should provide an API to configure the monitor directly. SALMon combines both approaches seamlessly.

Considering automatic configuration through SLAs, the current SLA standard is WS-Agreement [144]. However, WS-Agreement just provides a general-purpose schema that must be extended with internal sublanguages, which leads to different WS-Agreement compliant notations [44], and therefore a mechanism able to handle these different WS-Agreement notations is required.

To address this challenge, we use model-based strategies, and in particular model transformations, to obtain, from an arbitrary SLA written in any language, a unified monitoring configuration model able to configure the monitor. We propose to use a specific type of document, the Monitoring Management Document (MMD) that includes the required information to configure the monitor. By decoupling the monitor from an SLA, the same monitor can be used to monitor different SLAs in different notations (See Figure 12). Details of the MMD format can be checked at [44].

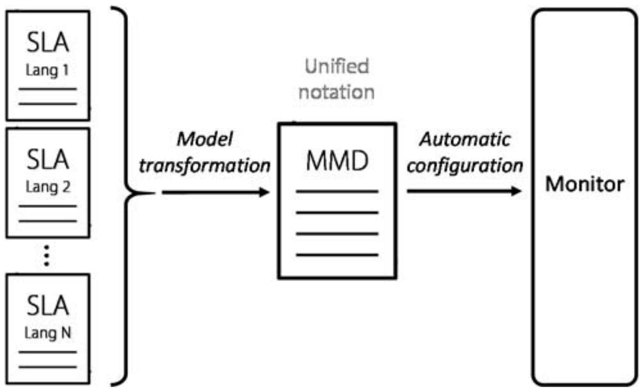


Figure 12 – Automatic configuration of monitors from different SLA notations.

Considering the configuration of the monitoring without SLAs, an invocation-based approach is followed. We provide an API in order to set the services, methods and metrics as well as other parameters to configure the monitor. The API is a WSDL interface, which can be invoked remotely by the client using any programming language. An extract of the WSDL interface is depicted in Figure 13.

Monitor Interface

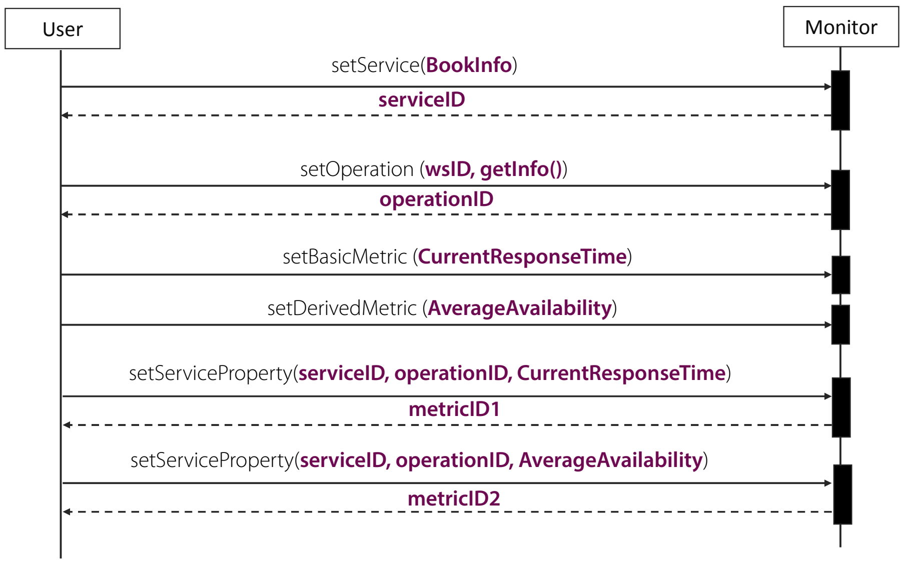
CreateSOASystem(soaName**):** soaID
SetClient(client): clientID
SetService(serviceInfo): serviceID
SetOperation(serviceID, operationInfo): operationID
SetBasicMetric(basicMetric)
SetDerivedMetric(derivedMetric)
SetServiceProperty(serviceID, operationID, metric, clientID): metricID
UpdateService (serviceID, serviceInfo)
UpdateOperation(serviceID, operationID, operationInfo)
UpdateServiceProperty(metricID, metric)
MonitorMetricForOperation(serviceID, operationID, metric, clientID)
StopMonitoringMetricsForOperation(serviceID, operationID, metric)
GetAllServicesFromSOASystem(soaID)
GetAllInvocationInformation(serviceID,operationID, timeInterval)
GetAllInputInformationFromService(serviceID, timeInterval)
GetAllInputInformationFromOperation(serviceID, operationID, timeInterval)
RemoveServiceProperty(metricID)
RemoveMetric(metric)
RemoveOperation(serviceID,operationID)
RemoveService(serviceID)
RemoveSOASystem(soaID)

Figure 13 – Monitor interface.

Regardless of the adopted strategy, SALMon can dynamically be reconfigured to adapt to changes in both the SBS and the SLA (e.g. add/remove services, metrics, etc.) (R1.3).

Running example:

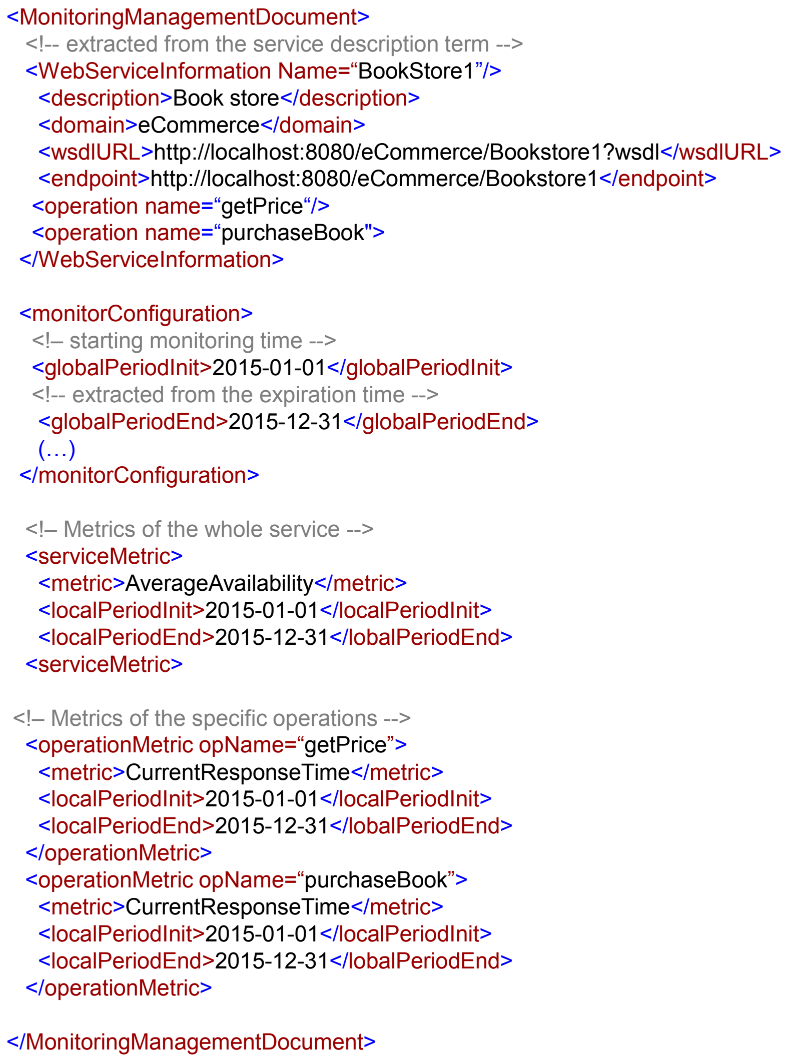
In our example, *BookInfo* is a web service developed in-house and lacks of any SLA. To monitor this web service, we use the invocation-based strategy. Figure 14 shows the sequence diagram to monitor the metrics *AverageAvailability* and *CurrentResponseTime* for the method *getInfo()* of the *BookInfo* web service.


User
Monitor
setService(BookInfo)
serviceID
setOperation (wsID, getInfo())
operationID
setBasicMetric (CurrentResponseTime)
setDerivedMetric (AverageAvailability)
setServiceProperty(serviceID, operationID, CurrentResponseTime)
metricID1
setServiceProperty(serviceID, operationID, AverageAvailability)
metricID2
Figure 14 – Sequence diagram to monitor the response time and availability of BookInfo.

On the contrary, the different *bookStore* and *currencyConvertor* web services, are external web services from different service providers and use different SLA notations to define their SLA clauses. For these web services, we use the model-driven approach. Figure 15 shows the SLA of *BookStore1* web service, with conditions on the *AverageAvailability* and *CurrentResponseTime* for the methods *getPrice* and *purchaseBook*. Such SLA is automatically transformed in a uniform MMD. Figure 16 shows the result of such transformation. All the generated MMDs use the same syntax and are used as the input to configure the Monitor.

```
<wsag:Agreement>
  <wsag:Name>BookStore1 - SLA agreement</wsag:Name>
  <wsag:Context>
    <wsag:ExpirationTime>2015-12-31</wsag:ExpirationTime>
     (…)
 </wsag:Context>
 <wsag:Terms wsag:Name=“BookStore1">
   <wsag:All>
     <wsag:ServiceProperties wsag:Name="Service Property 1" wsag:ServiceName=“BookStore1">
       <wsag:VariableSet>
           <wsag:Variable wsag:Name=“CurrentResponseTime" wsag:Metric="metric:LowInteger“/>
           <wsag:Variable wsag:Name="AverageAvailability" wsag:Metric="metric:Percentage“/>
       </wsag:VariableSet>
    </wsag:ServiceProperties>
    <wsag:ServiceDescriptionTerm wsag:Name=“BookStore-SDT" wsag:ServiceName="BookStore1">
       <WebServiceInformation name="BookStore1-WSDL">
       <description>Book store</description>
       <domain>eCommerce</domain>
       <wsdlURL>http://localhost:8080/eCommerce/Bookstore1?wsdl</wsdlURL>
       <endpoint>http://localhost:8080/eCommerce/Bookstore1</endpoint>
       <operation opName=“getPrice“/>
       <operation opName=“purchaseBook">
     </WebServiceInformation>
    </wsag:ServiceDescriptionTerm>
    <wsag:GuaranteeTerm wsag:Name=“AverageAvailability-GuaranteeTerm" wsag:Obligated="ServiceProvider">
      <wsag:ServiceLevelObjective>
        <wsag:CustomServiceLevel>AverageAvailability>=95</wsag:CustomServiceLevel>
      </wsag:ServiceLevelObjective>
    </wsag:GuaranteeTerm>
    <wsag:GuaranteeTerm wsag:Name=“getPrice-ResponseTime" wsag:Obligated="ServiceProvider">
      <wsag:ServiceScope wsag:ServiceName=“BookStore1">getPrice</wsag:ServiceScope>
      <wsag:ServiceLevelObjective>
        <wsag:CustomServiceLevel>CurrentResponseTime<3</wsag:CustomServiceLevel>
      </wsag:ServiceLevelObjective>
   </wsag:GuaranteeTerm>
   <wsag:GuaranteeTerm wsag:Name=“purchaseBook-ResponseTime" wsag:Obligated="ServiceProvider">
      <wsag:ServiceScope wsag:ServiceName=“BookStore1">purchaseBook</wsag:ServiceScope>
      <wsag:ServiceLevelObjective>
        <wsag:CustomServiceLevel>CurrentResponseTime<5</wsag:CustomServiceLevel>
      </wsag:ServiceLevelObjective>
   </wsag:GuaranteeTerm>
  </wsag:All>
 </wsag:Terms>
</wsag:Agreement>
```

Figure 15 – WS-Agreement for BookStore1 web service.



```
<MonitoringManagementDocument>
  <!-- extracted from the service description term -->
  <WebServiceInformation Name="BookStore1"/>
    <description>Book store</description>
    <domain>eCommerce</domain>
    <wsdlURL>http://localhost:8080/eCommerce/Bookstore1?wsdl</wsdlURL>
    <endpoint>http://localhost:8080/eCommerce/Bookstore1</endpoint>
   <operation name="getPrice"/>
   <operation name="purchaseBook">
 </WebServiceInformation>

 <monitorConfiguration>
   <!– starting monitoring time -->
   <globalPeriodInit>2015-01-01</globalPeriodInit>
   <!-- extracted from the expiration time -->
    <globalPeriodEnd>2015-12-31</globalPeriodEnd>
    (…)
 </monitorConfiguration>

  <!– Metrics of the whole service -->
  <serviceMetric>
    <metric>AverageAvailability</metric>
    <localPeriodInit>2015-01-01</localPeriodInit>
    <localPeriodEnd>2015-12-31</lobalPeriodEnd>
  <serviceMetric>

 <!– Metrics of the specific operations -->
  <operationMetric opName="getPrice">
    <metric>CurrentResponseTime</metric>
    <localPeriodInit>2015-01-01</localPeriodInit>
    <localPeriodEnd>2015-12-31</lobalPeriodEnd>
  </operationMetric>
  <operationMetric opName="purchaseBook">
    <metric>CurrentResponseTime</metric>
    <localPeriodInit>2015-01-01</localPeriodInit>
    <localPeriodEnd>2015-12-31</lobalPeriodEnd>
  </operationMetric>

</MonitoringManagementDocument>
```

Figure 16 – MMD generated from the BookStore1 WS-Agreement.

4.1.2 Combination of passive monitoring and online testing strategies

Passive monitoring consists on gathering the QoS information from the interaction between the web service and the service client. On the other hand, online testing consists on invoking periodically the web service to obtain the QoS information. Depending on the activity, it is required to use one passive monitoring (R2.1, R5.2) or online testing (R2.2, R5.1) to gather the QoS.

In order to satisfy the needs of the different activities, we combine both online testing and passive monitoring strategies in the same framework. To do so, the tester uses exactly the same monitoring infrastructure as an end-user of the service (see Figure 17).

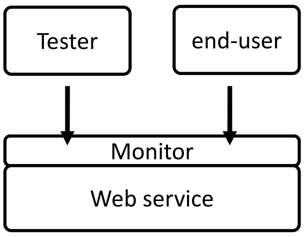


Figure 17 – Combination of passive monitoring and on-line testing.

The tester uses BPEL to define the test cases over the different web services of the SBS. One of the advantages of using BPEL is the ability to test web services in defined workflows. To set up the tester, another WSDL interface is provided with the methods to create a new workflow and configure such workflow with the settings of the test (see Figure 18).

Tester Interface
NewWorkflow(bpelProcess**):** workflowID **ConfigureWorkflows**(workflowID, settings)

Figure 18 – Tester Interface.

Both strategies are not mutually exclusive. That is, this feature also facilitates the combination of data obtained from passive monitoring and online testing to have more data and perform a better analysis.

Running example:

In our example, the *CurrentResponseTime* and *AverageAvailability* of *bookstore* and *currencyConvertor* web services are passively monitored to assess the fulfilment of the SLA. The passive monitoring strategy is used because the metrics gathered have to be from the real interaction between users and the web services. The interaction to set up the monitor has been shown previously in Figure 14.

At the same time, whenever any web service becomes unavailable, the SBS performs an adaptation following the MAPE loop (i.e. Monitor, Analyze, Plan, Execute) [145]. For instance, if the *currencyConvertor* is unavailable, the SBS will replace the *currencyConvertor* for another web service with the same functionality. To accomplish this task, the testing strategy is used. Particularly, to get the *currentAvailability* periodically and detect any problem before the user experiences any malfunction. To set the monitor to use the testing strategy, the user has to specify the workflow and the settings, which include the *set of inputs* to perform the tests and the *time interval* between invocations (see Figure 19).

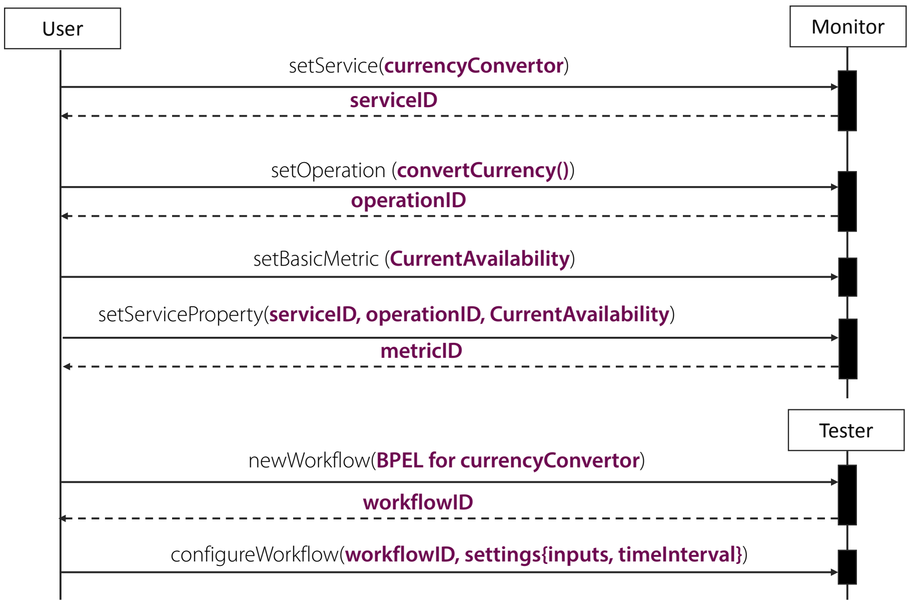

User
Monitor
setService(currencyConvertor)
serviceID
setOperation (convertCurrency())
operationID
setBasicMetric (CurrentAvailability)
setServiceProperty(serviceID, operationID, CurrentAvailability)
metricID
Tester
newWorkflow(BPEL for currencyConvertor)
workflowID
configureWorkflow(workflowID, settings{inputs, timeInterval})

Figure 19 – Sequence diagram to test the current Availability of currencyConvertor.

4.1.3 Extensible with new quality attributes

In a monitoring system, it is mandatory to provide the ability to compute new quality attributes as they are required (R1.2). To provide the extensibility with respect to new quality attributes, we provide two techniques, namely, at the conceptual and execution levels respectively (see Figure 20).

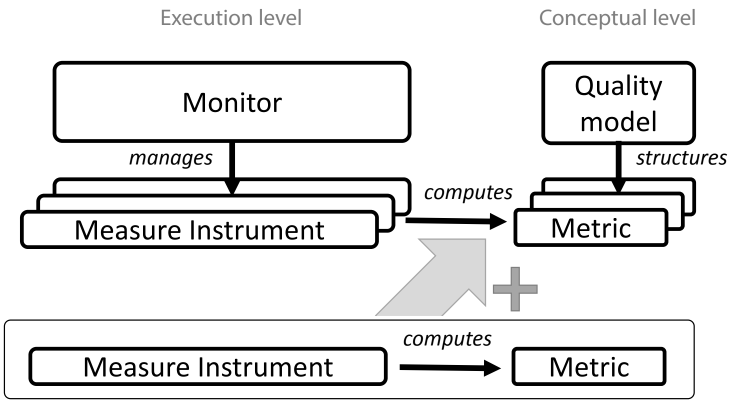


Figure 20 – Extensibility of metrics at the conceptual and execution level.

At the conceptual level, quality characteristics, attributes and metrics based may be structured following some quality model for web services, e.g. the standard ISO/IEC 25010 [11] and aligned with the common terminology used. In the previous chapter we have conducted a systematic mapping in order to identify the list of quality attributes and map them in the standard ISO/IEC 25010. By following this technique, new metrics can be added in a clearly structured manner following the standard, and interoperability and integration capabilities with other frameworks are also easier as they share a common framework of understanding.

At the execution level, SALMon is able to plug-in the required business logic that computes new metrics through a piece of software that we name measure instrument. A measure instrument is a component that is responsible to compute the values of a single metric. Hence, new metrics can be added by plugging the respective Measure Instruments. Adding new measure instruments can be done at runtime without the need of decommissioning or stopping the monitoring system. To do so, external Measure Instruments are implemented as web services, following a defined API in a WSDL interface. Such API consists on a single method that notifies events to the Measure Instruments and returns the results of the metrics (see Figure 21).

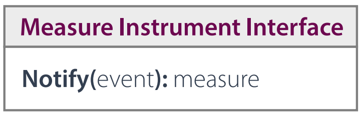


Figure 21 – Measure Instrument Interface.

These techniques enable SALMon to monitor metrics at the infrastructure level (R3.1) and also domain-dependent metrics.

Running example:

In our example, during the execution of the SBS some new metrics are required. Particularly, we are interested on monitoring the number of requests served per minute and the time required to repair the web service whenever it fails. Checking at the study of quality models, the former is usually known as *throughput* and belongs to the *Time behaviour* subcharacteristic, whereas the latter is usually known as *Mean Time to Repair (MTTR)* and belongs to the *Recoverability* subcharacteristic.

To compute the values of these metrics, the SBS developer has to implement the corresponding business logic using the interface given for Measure Instruments and register these Measure Instruments into the Monitor service. These Measure Instruments are then notified whenever an event happens, and the business logic that computes these metrics is executed. Figure 22 shows the interaction with the MTTR Measure Instrument, and Figure 23 shows the interaction with the Throughput Measure Instrument.

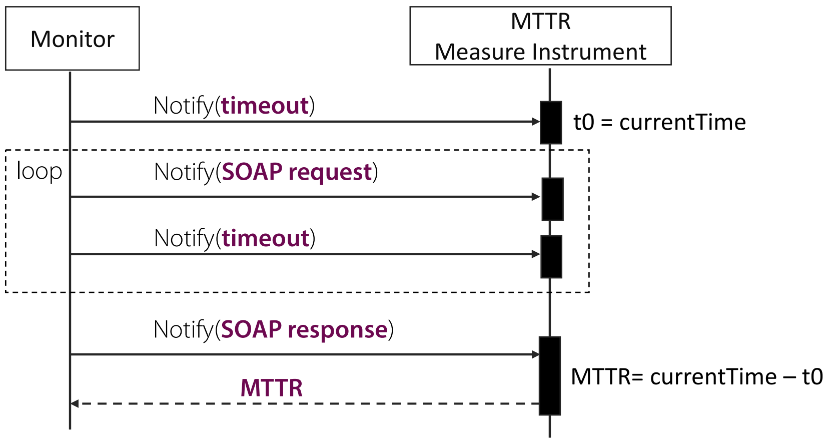


Figure 22 – Sequence diagram of the MTTR Measure Instrument.

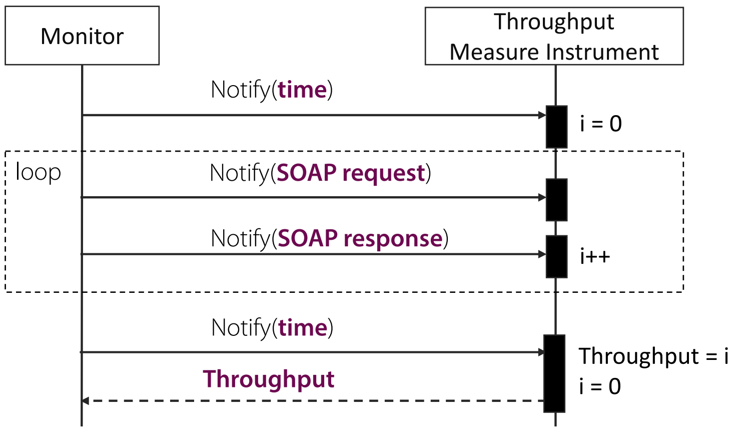


Figure 23 – Sequence diagram of the Throughput Measure Instrument.

4.1.4 For any type of web service

The high heterogeneity of the technologies used to implement web services requires a monitoring strategy not limited to a particular technical solution (R1.1). The monitor should not be attached to a particular technology and include the required modularity to extent it to support new kinds of web services. In contrast to other solutions, in our proposal, the messages are processed by the Measure Instruments, whereas the core of the monitor is agnostic over the technical specifications of the messages, leading the capability to monitor different type of web services (e.g. SOAP, RESTful services).

Running example:

During the selection of new web services for bookstores, a new *bookStore* web service is found that uses the RESTful protocol. To monitor such web service a Measure Instrument able to 'understand' RESTful requests and responses has to be implemented. This is done be implementing the WSDL interface for the Measure Instruments for RESTful. Figure 24 shows the MTTR Measure Instrument for RESTful.

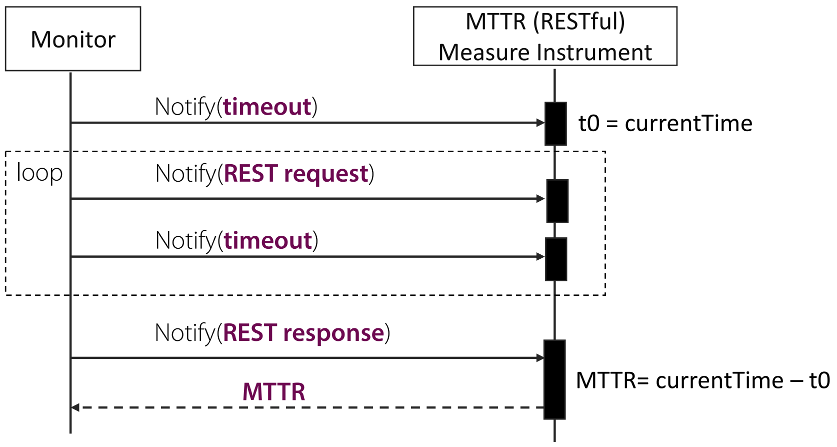


Figure 24 – RESTful Measure Instrument.

4.1.5 Combines push and pull notification mechanisms

Each activity requires different mechanisms to report the QoS data. In self-adaptive systems it is required to be notified as soon as possible with the updated QoS of the web services in order to perform an adaptation, whereas in service selection and deployment it is more convenient to provide the QoS of the web services only when they are required. To this aim, two strategies to retrieve the monitored QoS exist:

- **Push strategy:** QoS is notified to the subscribed components as soon as the monitored web service is invoked and the quality attributes are computed. The Push strategy implements the publish-subscribe pattern for SOA [5]. Using this pattern, a component named Publisher notifies the different Measurements to the subscribed clients (see Figure 25).

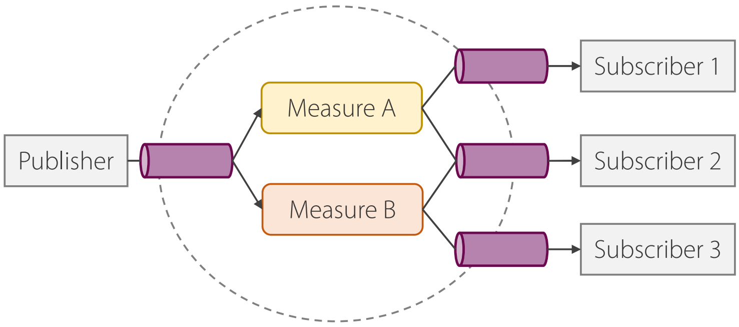


Figure 25 – Publish-Subscribe pattern.

The Publisher is a component of SALMon in charge of managing the different subscriptions; whereas the Subscribers have to implement a WSDL interface to receive the notifications (see Figure 26).

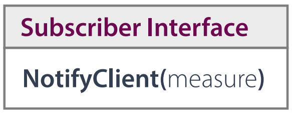


Figure 26 – Subscriber Interface.

- **Pull strategy:** QoS is requested by the components whenever they need it using the methods given in the WSDL interface of the Monitor.

SALMon combines both notification mechanisms to provide either the monitored data on-demand or as soon as the monitored data is gathered.

Running example:

In our example, if we require the QoS on demand, the user follows the pull strategy and invokes the methods of the monitor to get the measurements. Figure 27 shows the interaction to get the monitored QoS for a given web service.

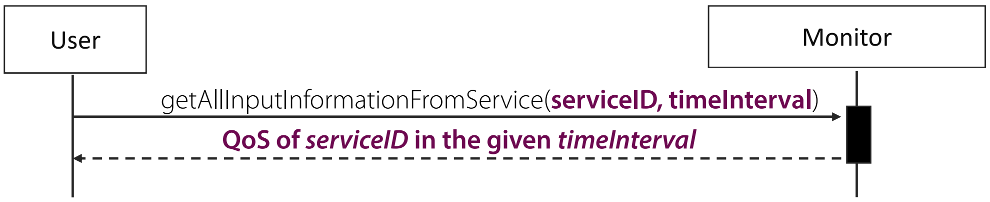

User
Monitor
getAllInputInformationFromService(serviceID, timeInterval)
QoS of serviceID in the given timeInterval

Figure 27 – Getting the QoS of a web service using the pull strategy.

If the QoS is required as soon as new measurements are computed, the user follows the push strategy. Using this strategy, the users implements a web service following the WSDL interface of the subscriber to receive the notifications. Figure 28 shows how a user subscribes to the measurements of *CurrentAvailability* for the method *convertCurrency* of *currencyConvertor* web service.

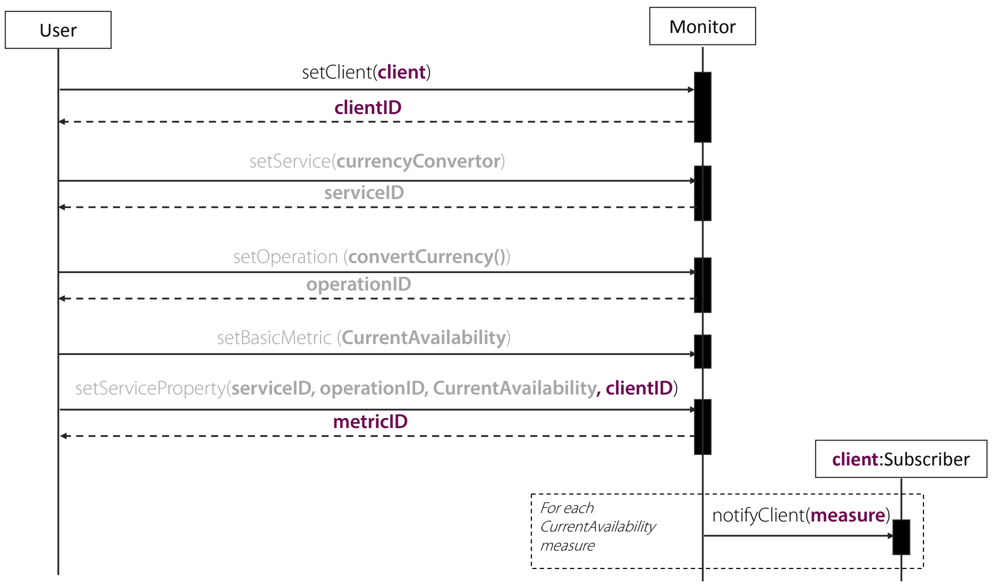

User
Monitor
setClient(client)
clientID
setService(currencyConvertor)
serviceID
setOperation (convertCurrency())
operationID
setBasicMetric (CurrentAvailability)
setServiceProperty(serviceID, operationID, CurrentAvailability, clientID)
metricID
client:Subscriber
For each CurrentAvailability measure
notifyClient(measure)

Figure 28 – Getting the measures of a web service using the push strategy.

4.1.6 High interoperability

Each SBS life-cycle activity requires specific features and specialized functionality that are implemented by dedicated software components. In order to be interoperable with these components (R1.4), the monitoring system must present a highly versatile architecture. To this aim, SALMon has been implemented following the SOA paradigm, where the different monitoring modules are web services, which facilitates the capability of each module to be easily coupled, replaced and decoupled in the monitoring system.

4.1.7 High efficiency

The monitor has to be efficient in reduce the consumption of resources (R3.2). SALMon has a modular architecture where the modules which are not required can be unplugged. Moreover SALMon has the capability to be deployed and decommissioned at runtime in order to reduce the consumption of resources.

4.2 SALMon's architecture

This section answers the Research Question RQ 2.3.

As mentioned above, SALMon has been designed following the SOA principles. These principles focus on high cohesion and low coupling aspects, which improve the reusability and maintainability of the software [3]. To this aim, SALMon is composed of several modules, each one with the constituent services required to fulfil a specific activity:

- **Core module:** implements the features that are required along the different activities.
- **MMD module:** implements the model-based management system.
- **Testing module:** implements the online testing strategy.
- **Subscription module:** implements the push notification strategy.
- **Data module:** implements the storage repository of the monitored data.
- **Monitor DB module:** implements a facade to interact with the data module.

Figure 29 shows these modules using a variation of the SAP-TAM notation [146].

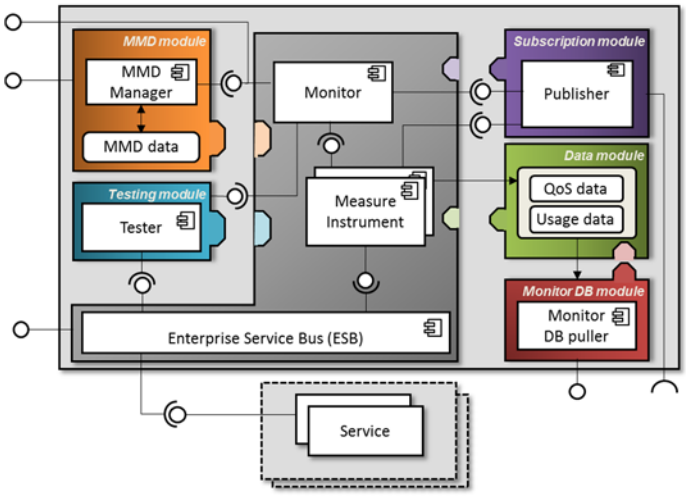


Figure 29 – SALMon's architecture.

4.2.1 Core module

The *Core module* is the main module of the monitoring system and the only one which is mandatory. It consists of the components required along the different activities to correctly fulfil the monitoring process. These components are the *Monitor*, the *Measure Instruments* and the *ESB*.

Monitor: The *Monitor* is the main component of the system. It is a service that includes in its WSDL interface the capability to be managed directly, accomplishing the invocation-based strategy. The Monitor does not compute directly the values of the quality metrics; instead, the *Monitor* is responsible for managing the *Measure Instruments*. For each quality metric to evaluate, the *Monitor* activates the corresponding *Measure Instrument*. The *Monitor* also manages other (optional) modules, namely the *Tester* and *Publisher* modules, if present.

The *Monitor* has been implemented in Java using Axis2 engine running under Tomcat.

Measure Instrument: Each *Measure Instrument* includes the business logic to compute the value of a specific quality metric. A *Measure Instrument* can be either internal or external. An *Internal Measure Instrument* is included as a component of the *Core Module* and are the mostly wide used quality metrics (e.g. response time, availability, …), whereas *External Measure Instruments* are web services with a common interface that implement a specific quality metric required by the user (e.g. size of attached files). In such a manner, the *Monitor* is extensible with the addition of new metrics. The *Measure Instruments* receive the messages to monitor from the ESB.

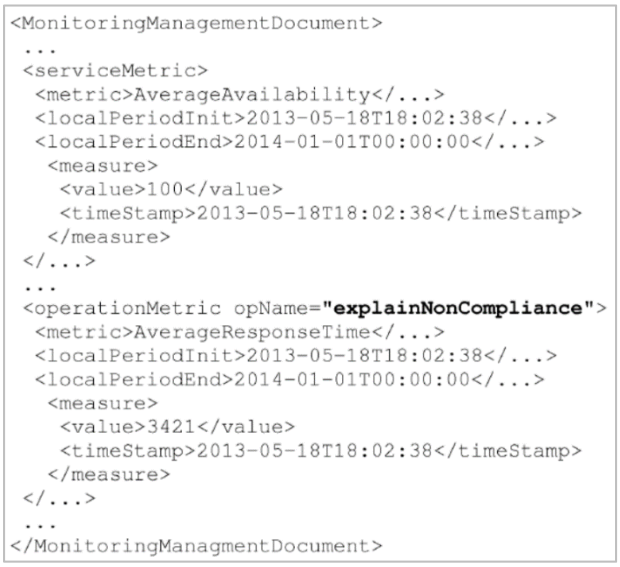
Internal Measure Instruments have been implemented as Java packages of the *Monitor* web service. *External Measure Instruments* implements a WSDL interface and can be implemented using any technology compliant with WSLD and SOAP.

ESB: In order to capture the messages from a service consumer, the service consumer must not invoke the service directly but through the usage on an *Enterprise Service Bus (ESB)*. This is fulfilled by specifying the target address using the standard WS-Addressing in the header of the messages and performing the invocation to the *ESB*. The *ESB* receives the message requests and forwards them to the actual web service. In order to avoid delays caused by message redirections, the *ESB* can be placed to the server or client side. The location of the *ESB* also depends on the metrics to calculate. For instance, to compute the response time, which includes the network delay, the *ESB* is required to be deployed near or at the client side. On the contrary, to compute the execution time, excluding the network delay, the *ESB* has to be deployed near or at the server side. The architecture allows deploying more than one *ESB*, which can be combined by applying the required redirections. In such a manner, SALMon (1) allows the computation of server and client side metrics and (2) provides the ability to avoid bottlenecks, by instantiating new *ESBs* in case of high traffic. In parallel to invoking the web service, the *ESB* forwards the messages accompanied with their timestamps to the activated *Measure Instruments*, so they can compute the values of the quality metrics.

The *ESB* used is Apache Synapse. The rules or redirection have been implemented combining XML directives provided by Apache Synapse and particular rules implemented in Ruby.

4.2.2 MMD module

This module is used to implement the model-based management system. In this regard, it is required a model transformation from the source model to the model that configures the monitor. The source model is not limited to SLA documents, but also can deal with other types of models (e.g. goal-based models). By specifying the appropriate transformation rules and using techniques as those presented in [147], we are able to derive from the source model to the model that configures the monitor (i.e. the target model). With respect to the target model, we propose to use the MMD [44], a specification which includes what is required to monitor and how the data is to be gathered. The MMD, as a standalone model, is a formal XML document whose main functions are (1) the specification of a monitoring configuration to gather properly the required metrics and (2) a container to store and retrieve the monitoring results in the same document structure. In this sense, the MMD is used as both input and output model of the system. An excerpt of an MMD with monitoring results is depicted in Figure 30.



```
<MonitoringManagementDocument>
 ...
 <serviceMetric>
  <metric>AverageAvailability</...>
  <localPeriodInit>2013-05-18T18:02:38</...>
  <localPeriodEnd>2014-01-01T00:00:00</...>
   <measure>
    <value>100</value>
    <timeStamp>2013-05-18T18:02:38</timeStamp>
   </measure>
 </...>
 ...
 <operationMetric opName="explainNonCompliance">
  <metric>AverageResponseTime</...>
  <localPeriodInit>2013-05-18T18:02:38</...>
  <localPeriodEnd>2014-01-01T00:00:00</...>
   <measure>
    <value>3421</value>
    <timeStamp>2013-05-18T18:02:38</timeStamp>
   </measure>
 </...>
 ...
</MonitoringManagmentDocument>
```

Figure 30 – Excerpt of an MMD with monitoring results.

MMD Manager: the *MMD Manager* is responsible for managing the MMD. It also invokes the *Monitor* accordingly to the *MMD* rules and updates the document with the monitoring results.

The *MMD Manager* has been implemented as a web service in Java using Axis2 engine running under Tomcat.

MMD data: The MMDs are stored and updated in a secured repository.
The *MMD data* is stored using MySQL.

4.2.3 Testing module

The testing module is used to perform online testing to the target web services. It consists of the *Tester Engine.*

Tester Engine: The *Tester Engine* is activated by the monitor by specifying the BPEL workflow to test and the settings, which include the set of inputs to perform the tests and the time interval between invocations. The tester uses exactly the same infrastructure that is used for the messages from a real service consumer. The test messages goes through the same ESB redirections. In such a manner, any duplicity is avoided, and the combination of both approaches is assured. To identify that these requests do not belong to a real user but are from the *Tester Engine*, the test messages include an identifier tag in the header of the message.

The *Tester Engine* has been implemented in Java using Axis2 engine running under Tomcat. The *Tester Engine* uses Apache ODE to execute the tests using the BPEL language.

4.2.4 Subscription module

The subscription module implements the push notification strategy to inform updates regarding the monitored web services as soon as there are gathered. It is operationalized by the *Publisher* service.

Publisher: The *Publisher* implements the observer pattern for web services. The Monitor subscribes the metrics or other data computed by the *Measure Instruments* to the interested party (e.g. the consumer, provider, an adaptation component, etc.). The *Publisher* handles the list of subscribed parties and their subscriptions and notifies the events whenever a new value of interest is retrieved by the *Measure Instruments*. The subscribed parties must implement a web service interface that handles such notifications.

The *Publisher* has been implemented in Java using Axis2 engine running under Tomcat.

4.2.5 Data module

The Data module is a repository to store the monitored data, which can be accessed internally by the *Monitor* or externally by the *Monitor DB puller*. It consists of two major building blocks:

QoS data: Is the part of the repository which stores all the monitored QoS information (e.g. response time, availability, etc.).

Usage data: is the part of the repository which stores all the information related to the usage of the web services (e.g. invocations performed, inputs used, etc.).

The current repository has been implemented as a MySQL database, but other technologies could be used as required by the SBS.

4.2.6 Monitor DB module

This module is used to get the information from the Data module externally from the monitor. It consists of the *Monitor DB puller*.

Monitor DB puller: It provides a WSDL interface to access the data from the repository, decoupling the user to a particular storage technology.

The Monitor DB puller has been implemented in Java using Axis2 engine running under Tomcat.

4.3 Technologies used in SALMon

In this section we summarize the list of technologies used in the different components of SALMon. Such summary is depicted bellow in Table 34.

Table 34 – List of technologies used in SALMon.

Module	Component	Technologies
Core	Monitor	Java, Axis2, Tomcat
Core	Measure Instrument	Java, Axis2, Tomcat
Core	ESB	Apache Synapse, Synapse directives, Ruby.
MMD	MMD Manager	Java, Axis2, Tomcat
MMD	MMD Data	MySQL
Testing	Tester Engine	Java, Axis2, Tomcat, Apache ODE
Subscription	Publisher	Java, Axis2, Tomcat
Data	QoS data	MySQL
Data	Usage data	MySQL
Monitor DB	Monitor DB puller	Java, Axis2, Tomcat

4.4 SALMon execution

In this section we describe how SALMon is configured and executed. As shown in Figure 31, the client of SALMon can configure the *Monitor* either by invoking directly its interface (1) or by using an MMD that specifies the required configuration (2). In the latter case, the *MMD Manager* processes the MMD and invokes the *Monitor* interface accordingly (3). Once configured, the *Monitor* activates the *Tester* (if required) (4), the *Measure Instruments* needed (5) and subscribes the client for the measurements to be notified (10). During the execution of the SBS, the *Tester* invokes the service using the same infrastructure (6) as the real service clients (7). The ESB captures such invocations and, they are forwarded to the *Service*. In parallel, such invocations are notified to the *Measure Instruments* (9). The *Measure Instruments* compute and store the monitored metrics, and notifies such measurements to the *Publisher* (11). The *Publisher*, in turn, notifies the measurements to any subscribed client (13), who has also access to the monitored data through the API (12).

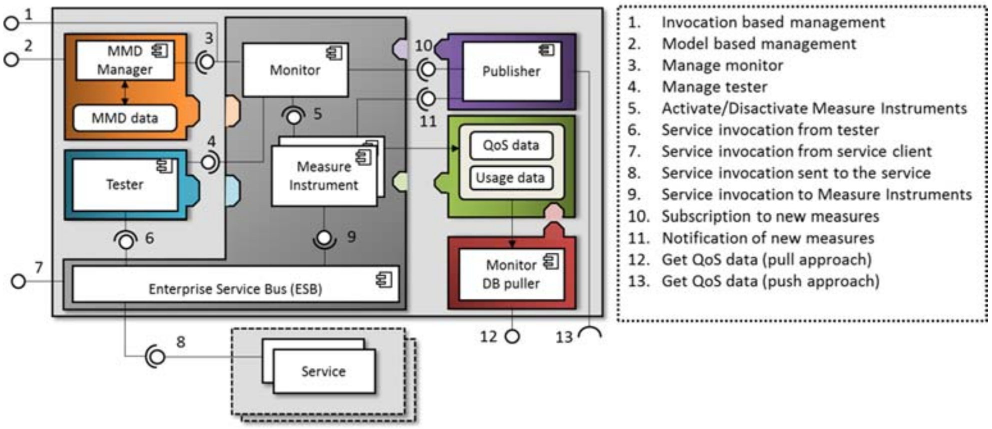

1
2
3
MMD Manager
MMD data
Monitor
5
4
Tester
Measure Instrument
6
9
7
Enterprise Service Bus (ESB)
8
Service
10
Publisher
11
QoS data
Usage data
Monitor DB puller
12
13
1. Invocation based management
2. Model based management
3. Manage monitor
4. Manage tester
5. Activate/Disactivate Measure Instruments
6. Service invocation from tester
7. Service invocation from service client
8. Service invocation sent to the service
9. Service invocation to Measure Instruments
10. Subscription to new measures
11. Notification of new measures
12. Get QoS data (pull approach)
13. Get QoS data (push approach)

Figure 31 – SALMon's execution process.

4.5 Performance evaluation

In this section we evaluate the performance of SALMon. Particularly, we aim at evaluating the overhead and the capacity of SALMon. To quantify it, we evaluate by means of an adequate benchmark: (1) the response time overhead under normal operating conditions (i.e. one invocation at a time), and (2) the maximum throughput SALMon is able to handle without incrementing such overhead.

To perform the evaluation, we invoke a set of real services, and compare the response time by invoking the services both directly and through SALMon. To obtain a set of representative services, we started from a list of 393 services available in a public repository[6]. Then we applied the following criteria: (1) We first considered the most recently submitted services under the assumption that recent services are more likely to be available than older services. Considering the length of the list, we established as threshold the 1/3 of the complete list. (2) From the resulting 131 services, we removed those ones falling into any of the following situations: were not available, were payment services, required registration or did not have stateless operations, resulting in 23 services. (3) We tested these 23 services and removed those ones that had errors in their descriptions (WSDL), or that gave faulty results in their functionality when invoked, resulting in a final list of 11 services from 8 different service providers, deployed on their respective servers. The list of services and their WSDLs are available at [148].

[6] http://www.xmethods.net/ve2/Directory.po

4.5.1 Overhead evaluation

The ESB Apache Synapse included in SALMon adds a low overhead while handling the HTTP messages. The ESB has a non-blocking HTTP transport and multithreaded mediation, which as we measured, results in a negligible 1-3 ms overhead. Nevertheless, in our approach, there are three possible locations where SALMon can be deployed: at the server side, at the client side, or in an intermediate server (i.e., in the middle). Depending on the location, the overhead experienced by the consumer varies. If SALMon is placed at the server or client side, there is an overhead on the resources due to the execution of the monitoring components. However, this overhead can be easily compensated by adding more resources.

If SALMon is placed in the middle, it does not produce an overhead on the resources of the client or server side. However, the deployment of SALMon in an intermediate server adds a network delay from Internet Service Providers due to the redirection of the messages.

To evaluate the overhead, we executed each of the selected services 100 times, with a throughput of 1 invocation per second. One key issue regarding the analysis of the results is dealing with outliers (e.g. network failures that increase the response time of an invocation). Commonly used methods to deal with outliers require that the data follow a Gaussian distribution [149]. However, from the experiment results we have observed that response times do not follow a Gaussian distribution, but an exponentially modified Gaussian or inverse Gaussian distribution (See Figure 32).

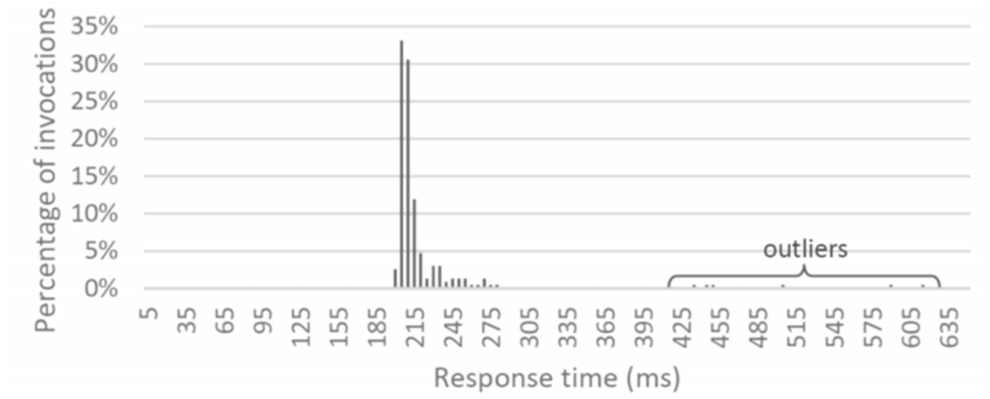


Figure 32 – Response time distribution of a monitored web service.

As shown, the population grows rapidly on the left-hand side and decreases slowly on the right-hand side in the form of a tail. Those elements that are far away from the mean are considered outliers. To deal with these outliers, we followed the methods described and evaluated by Ratcliff for dealing with response time outliers [150]. Although Ratcliff studied response time of people in the field of psychology, the results can be applied to any model that follows the inverse Gaussian distribution. According to Ratcliff, we will not compute directly the average response time (which is not a robust estimator in front of outliers), but we will use two other robust estimators, namely, the inverse transformation and removing outliers at a standard deviation distance. The first estimator consists on applying the inverse response time (1/R) on each individual invocation, calculate the average, and then invert the result. The second estimator consists on calculating the average response time after removing the outliers at a standard deviation distance. We computed these methods over the invocations on each service for both directed and redirected invocations. As a result, we got two robust estimators per each service. We applied these estimators to the response time of direct and redirected invocations in order to calculate the response time overhead introduced in the web service by the deployment of

SALMon. We decided to relate the two parameters with a linear interpolation curve fitting method with the aim of obtaining mathematical functions approximating the response time overhead.

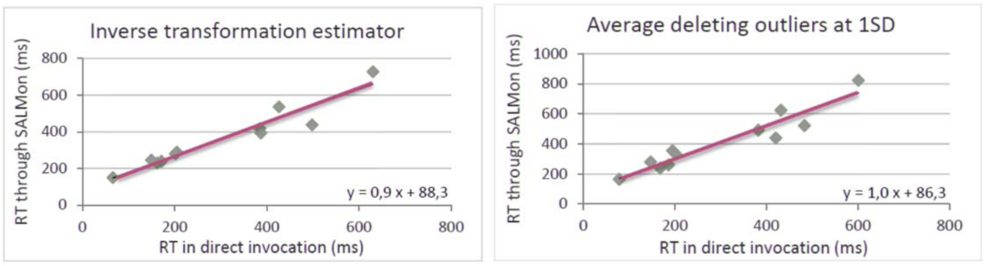


Figure 33 – Response time of the web services, invoked directly and through SALMon.

Figure 33 shows the obtained functions for each of the two applied robust estimators methods, which are: y = 0.9x + 88.3ms and y = 1.0x + 86.3ms. These results show that the overhead of SALMon is a constant value between 86 and 89 ms. It is worth to mention that some deployment strategies can be applied to mitigate any concern. We argue that for web services that require extremely fast response times and need to avoid the 86-89 ms overhead, SALMon can be deployed at the client or server side, as the overhead is mainly caused by the network delay. For other types of web services, we argue that a deployment in the middle is preferred, since this solution is less intrusive to both the client and the provider server, as it does not require the installation of the monitor in their infrastructures.

4.5.2 Capacity evaluation

To evaluate the capacity of SALMon we invoked each web service directly and through SALMon with different throughputs. We tested the list of web services with throughputs from 1 invocation per second up to 50 invocations per second, and per each throughput we made 100 invocations.

For web services with a capacity lower than SALMon, the results are not useful to identify the capacity of SALMon. For web services with a high capacity, e.g. CalcService (Figure 34), we identified that SALMon is able to operate correctly with throughputs up to 30 invocations per second. For throughputs that go beyond 30 invocations per second, the capacity of SALMon is outreached after several invocations (usually taking more than 100 invocations). Nevertheless, such limitations can be overcome by deploying more ESBs in a distributed set of servers. The complete list of results (graphical and raw data) is available at [148].

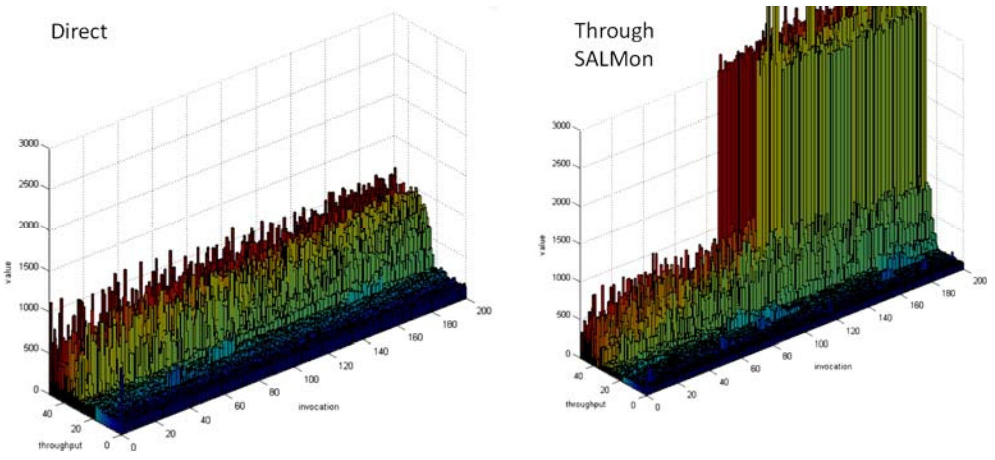


Figure 34 – Invocations to CalcService with different throughput.

4.6 SALMon in the SBS lifecycle

SALMon has been validated in depth in the different scenarios supporting the previously mentioned stages of the lifecycle of an SBS. We detail here, how SALMon has been used in the different frameworks of several research groups to accomplish a varied set of activities. The features that were implemented and the results of such collaborations are detailed.

4.6.1 Service selection

The selection framework

SALMon has been integrated with one web service selection framework to demonstrate the feasibility of the monitoring framework in this activity. Particularly, SALMon has been integrated with WeSSQoS [26][25]. WeSSQoS is a configurable quality-aware web service selection framework that combines multiple web service repositories and algorithms to provide and augmented user experience in the selection of web services. The repositories integrated with WeSSQoS provide information about the web services and some statically defined quality attributes. WeSSQoS combines all this information with SALMon for an accurate and up-to-date QoS data of the web services.

Required components of SALMon and usage

The required components of SALMon in this activity are: the *Tester* and the *Data* module (see Figure 35).

To monitor these web services, the *Tester* component of SALMon performs the tests over the web services registered in the repository of WeSSQoS. The QoS data is then stored in the *QoS data* module, which can ultimately be retrieved by the *Selector* module of WeSSQoS.

The *Selector* module merges all the data and executes the normalization and ranking algorithms provided by the Normalize & Ranking Module, resulting in a ranked list of the discovered web services that better fulfils the user’s needs.

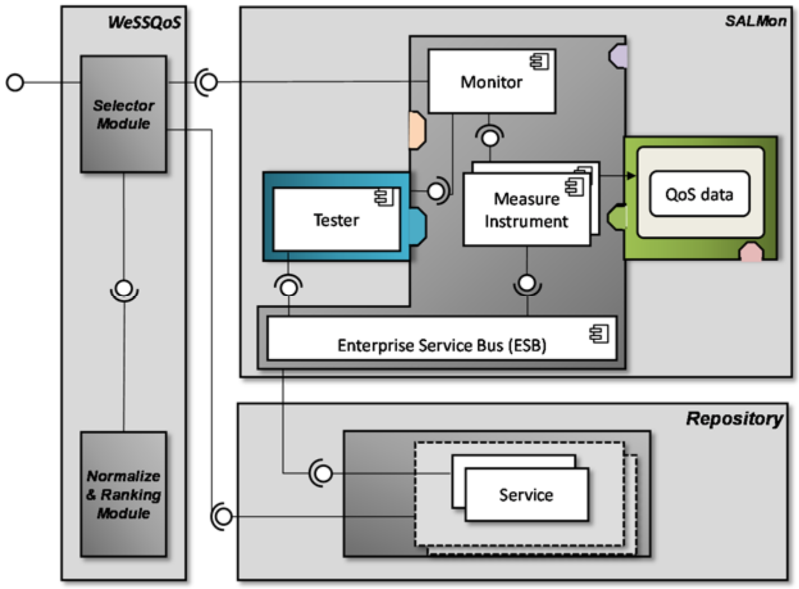


Figure 35 – SALMon in service selection.

4.6.2 SBS deployment

SBS deployment framework

To demonstrate the feasibility of SALMon during the SBS deployment activity on heterogeneous deployment infrastructures, SALMon has been integrated with a cloud federation system. Particularly, in the Federated Cloud Management (FCM) framework [28], [124]. FCM is a brokering system that integrates heterogeneous cloud systems and provides a unique broker to deploy and use SBS in the cloud seamlessly. SALMon has been integrated with this solution to monitor the QoS at the infrastructure level.

Required components of SALMon and usage

The required components of SALMon in this activity are: the *Tester*, the *Data* module and the *Monitor DB puller* module. Moreover, an external web service designed for the framework, named *M3S,* is also required (see Figure 36).

M3S is a web service that incorporates several methods designed to make usage of the infrastructure of the platform where it has been deployed (e.g. network, CPU, etc.). By monitoring the performance of the different methods provided by the web service, the QoS at the infrastructure level is computed.

To be accurate in the results of monitoring *M3S*, the core of SALMon and the *Tester* is deployed in the same platform as the *M3S* service. The *Data* module and the *Monitor DB puller* are deployed outside of the aforementioned platform in order to (1) avoid the consumption of memory and disk in the platform that is being monitored and (2) be able to access the data even when the core of SALMon and the *M3S* have been decommissioned. The Federated Cloud Management System is then able to retrieve the QoS through the *Monitor DB puller*.

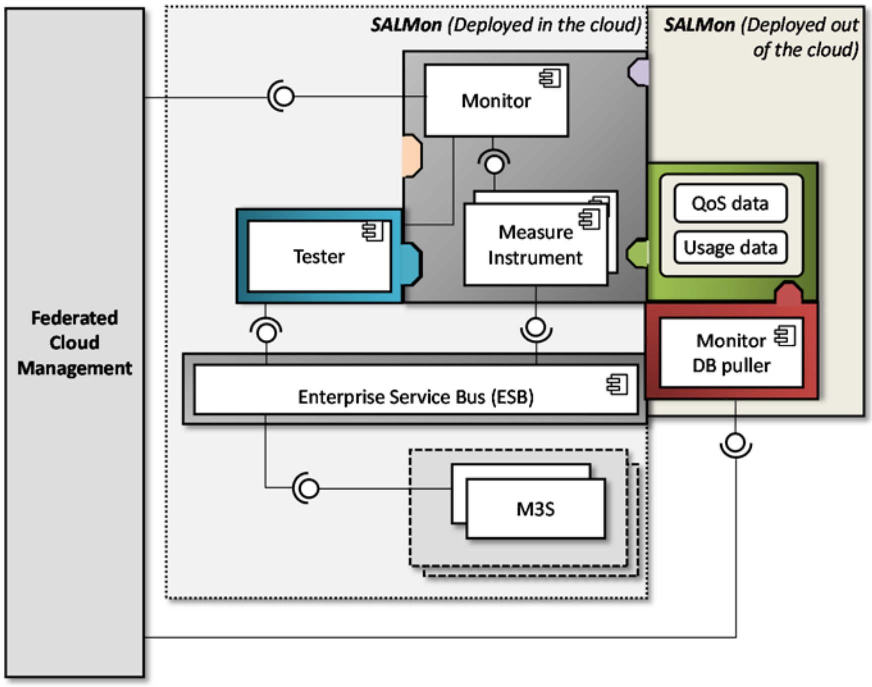

SALMon (Deployed in the cloud)
SALMon (Deployed out of the cloud)
Monitor
Tester
Measure Instrument
QoS data
Usage data
Monitor DB puller
Enterprise Service Bus (ESB)
M3S
Federated Cloud Management

Figure 36 – SALMon in SBS deployment.

4.6.3 Quality assessment

Quality assessment framework

The QoS to be achieved during the execution of a web service is usually agreed in the form of an SLA. SALMon has been integrated with an SLA analysis platform called ADA [31], and extended to form a new technological solution named SALMonADA [32], [44]. SALMonADA is a framework able to automatically monitor, detect and analyse violations of SLA clauses during the execution of the web services. Beyond other features, SALMonADA is able to provide human readable explanations of SLA violations for highly expressive SLAs as soon as they occur.

Required components of SALMon and usage

The required components of SALMon in this activity are: *MMD* module, *Subscription* module and *Data* module (see Figure 37).

To perform the analysis, SALMonADA automatically generates from the SLA, the MMD document to configure the monitor. As the service client invokes the web service through the *ESB*, the web services are being monitored and every new computed metric is notified through the *Subscription* module to (1) generate the updated MMD with the monitored metrics and (2) compute the analysis if any violation has occurred.

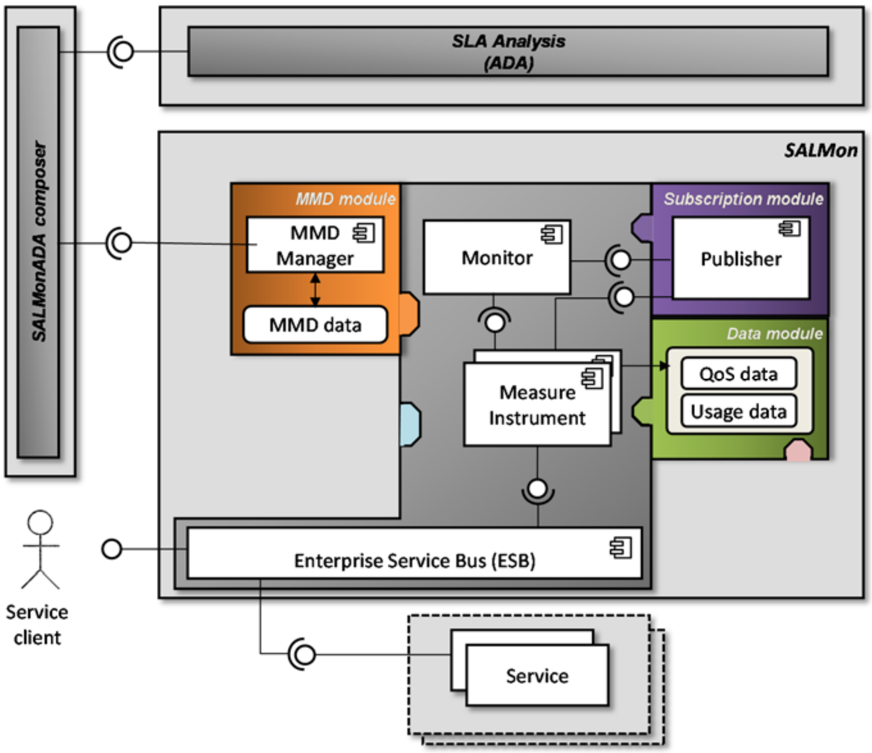


Figure 37 – SALMon in quality assessment.

4.6.4 SBS adaptation

SBS adaptation frameworks

SALMon has been integrated with different frameworks for self-adaptive systems: MAESoS [35], PROSA[36], PROTEUS [45] and CARE [37].

MAESoS [35] is a framework that supports self-adaptation of SBS by combining several models. It uses *i** models to specify requirements, tasks and dependencies; feature models to specify rules and alternatives; and quality models for the description of the QoS required.

PROSA [36] is a framework that supports self-adaptation of SBS based on failure prediction. This is achieved by combining passive monitoring and active testing dynamically.

PROTEUS [45] is a framework that supports self-adaptation of SBS to prevent violations due to malfunction of the executed services. The adaptation strategies are focused on mitigating the effects of any service malfunction.

CARE [37] is a framework that supports the adaptation of requirements in SBS by involving the end-users, if needed, to satisfy their needs.

Required components of SALMon and usage

The required components of SALMon in this activity are: the *Subscription* module and *Data* module. It also requires a new component, which is an *Analyser* to detect the adaptation needs (see Figure 38).

The *Analyser* is able to check simple conditions and trigger an adaptation need if such conditions are not met. The *Analyser* configures the monitor either in passive monitoring or online testing strategies, and hence, both proactive and reactive adaptations are supported.

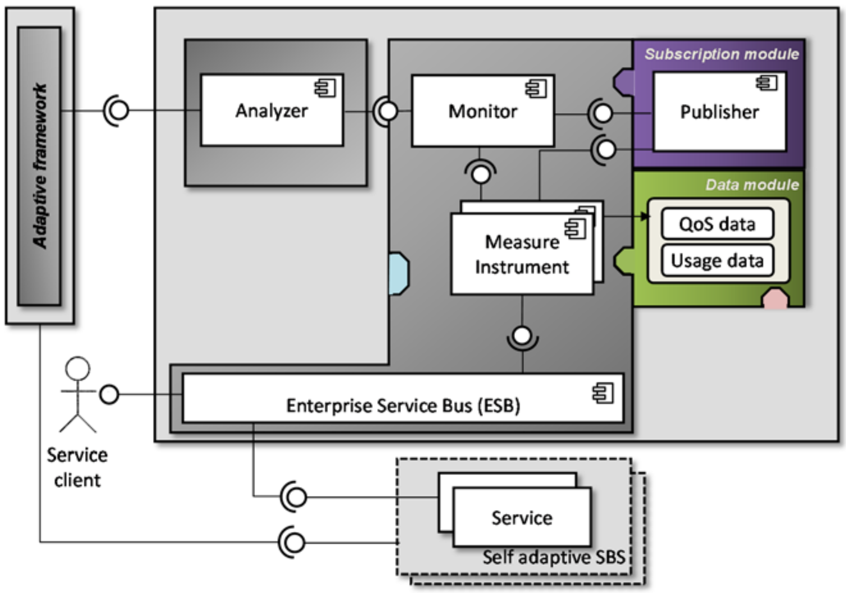


Figure 38 – SALMon in SBS adaptation.

CHAPTER 5 Conclusions

Once more, *"Every new beginning comes from some other beginning's end"* - Seneca.

5.1 Conclusions of Part I

In Part I we have surveyed the state of the art in the definition of quality models for web services. The interest on quality models stems from their applicability in the study of the wider concept of quality of service. We designed and followed a rigorous protocol which uncovered up to 65 papers presenting 47 proposals to answer the different research questions that we identified. We summarize these answers below:

RQ 1.1. What is the chronological overview of the research done so far in quality models for web services?

H 1.1	It is possible to draw a chronological overview of the proposed quality models for web services, with their relationships and influences.	✓

Quality models for web services have been an increasingly addressed research topic from 2001 to the current days, with the exception of the last 2-year period. In section 2.3.1 we have distributed chronologically the 47 proposals showing their relationships and identifying which ones are the most consolidated ones, the most influencing ones and the most influenced ones.

H 1.2	It is possible to identify the list of characteristics and evaluate the different quality models based on their size and definition.	✓

RQ 1.2. What are the characteristics of the proposed quality models?

In section 2.3.2, we have observed that the size varies among 6 and 66 nodes (with an average of 24,38) and the number of levels among 1 and 3 (with an average of 2,23), with a correlation between both factors. We have also observed that a narrow majority of proposals (51%) have a unique and consistent definition for all their quality factors, and only 19% of them have a completeness definition below 40%.

RQ 1.3. Which quality factors are the most addressed in the quality models? Which are the least addressed?

H 1.3	It is possible to identify which characteristics are the most and least addressed.	✓

In section 2.3.3, we have shown that first *reliability* and then *security* and *performance efficiency* are the ISO/IEC 25010 characteristics explicitly defined in at least half of the surveyed proposals. Concerning subcharacteristics, the ones explicitly or defined in at least half of the proposals are: *functional correctness* and *availability*. Also, *time behaviour*, capacity, *confidentiality* and *economy* exceed this threshold if implicit definitions are considered too.

RQ 1.4. What are the most consolidated quality factors?

H 1.4	It is possible to identify the most consolidated quality factors.	✓

In section 2.3.4, we have obtained up to 19 quality attributes that appear in at least 30% of the surveyed quality models. Among them, *availability* is the most used and has an almost universally-agreed definition; also *response time*, *throughput*, *cost* and *accuracy* appear in more than half of the proposals. In general, we can say that there is much consensus in the definitions of these 19 most popular quality attributes: up to 7 of them are defined the same way in all the quality models that introduce them, and the rest of them are still consistently used in more than a half of the proposals, with the exception of *accuracy*.

5.2 Conclusions of Part II

In Part II, we have presented SALMon, a highly versatile QoS monitoring framework able to support the whole SBS lifecycle.

RQ 2.1. To which degree the proposed monitoring frameworks support the whole SBS lifecycle.

H 2.1	Current monitoring frameworks do not support the whole SBS lifecycle.	✓

In section 3.2.2, we have identified the requirements of the different activities in the lifecycle by (1) analysing the needs as described in the literature, and (2) conducting meetings and interviews with members of different research groups who work on these activities. In section 3.2.3, we have presented the results of the Systematic Literature Review. We have shown that current monitoring frameworks support (partially) some of the requirements, but none of them satisfactorily cover the whole SBS lifecycle.

RQ 2.2. What are the features required when monitoring QoS to support the SBS lifecycle?

H 2.2	A list of required features to support the SBS lifecycle can be identified.	✓

In section 4.1, from the elicited requirements, we have designed a set of features and developed the SALMon framework. The main features that SALMon offers are:

- Combination of model-based and invocation-based strategies
- Combination of passive monitoring and online testing strategies
- Extensibility with new quality attributes
- Low coupling to the monitored services' technology
- High interoperability

RQ 2.3. How these features can be implemented in a single framework?

H 2.3 All the required features can be implemented in a single framework . ✓

As described in section 4.2, we have implemented SALMon following a modular service oriented architecture. We have shown that, with this modular architecture, we are able to deploy the required SALMon's components suitable for each activity. In section 4.6, we have executed and validated our approach by including SALMon in several frameworks from different research centers and universities for the different activities of the SBS lifecycle. Namely, web service selection, SBS deployment, quality assessment and SBS adaptation.

Finally, in section 4.5, we have conducted a performance evaluation over real web services using suitable estimators for response time and evaluated both its overhead and capacity.

5.3 Future work

Both the work in Part I (what to monitor) and Part II (how to monitor) can be extended in different directions.

In Part I, the analysis of the systematic mapping can be extended to answer new research questions of interest for the community. We believe that the results of the systematic mapping can be used beyond the scope of QoS monitoring, and thus, it can be improved to analyse what is the current usage and applications of the different quality models.

In Part II, SALMon can be extended to provide some enhanced features. The testing module of SALMon could be extended to define test cases using a standard model (e.g. UML Testing Profile), and thus facilitating Test-driven Development for SOA. It would also be interesting to implement a public repository of Measure Instruments where the user can select the Measure Instruments to gather the quality metrics, and thus (1) facilitating the reuse of Measure Instruments and (2) provide a framework to compare and evaluate the different Measure Instruments.

As for the scope of SALMon, we plan to extend the framework beyond the field of SOA and assess the health of Open Source Software ecosystems. To do so, SALMon will monitor tools used by Open Source Software communities which are implemented as web services. Preliminary results have already been published in [39].

CHAPTER 6 Bibliography

[1] C. Ghezzi, M. Jazayeri, and D. Mandrioli, *Fundamentals of Software Engineering*, 2nd ed. Prentice Hall, Pearson Education, 2003.

[2] M. P. Papazoglou, "Service-oriented computing: concepts, characteristics and directions," in *Proceedings of the 7th International Conference on Properties and Applications of Dielectric Materials (ICPADM)*, 2003, pp. 3–12.

[3] E. Di Nitto, A.-M. Sassen, P. Traverso, and A. Zwegers, *At Your Service: Service-oriented Computing from an EU Perspective*, 1st ed. MIT Press, 2009.

[4] V. Andrikopoulos, A. Fairchild, Willem-Jan, van den Heuvel, R. Kazhamiakin, P. Leitner, A. Metzger, Z. Nemeth, E. di Nitto, M. P. Papazoglou, B. Pernici, and B. Wetzstein, "Deliverable CD-IA-1.1.1, Comprehensive overview of the state of the art on service-based systems," 2008.

[5] T. Erl, *Soa: principles of service design*, vol. 1. Prentice Hall Upper Saddle River, 2008.

[6] N. Bieberstein, R. Laird, K. Jones, and T. Mitra, *Executing SOA: A Practical Guide for the Service-Oriented Architect*, 1st ed. Addison-Wesley Professional, 2008.

[7] M. P. Papazoglou, K. Pohl, M. Parkin, and A. Metzger, Eds., *Service Research Challenges and Solutions for the Future Internet*, vol. 6500. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010.

[8] ISO IEC, “ISO 8402:1994 - Quality management and quality assurance - Vocabulary.” 1994.

[9] S. T. Albin, *The Art of Software Architecture: Design Methods and Techniques*, 1st ed. John Wiley & Sons, 2003.

[10] M. P. Papazoglou, P. Traverso, S. Dustdar, and F. Leymann, “Service-Oriented Computing: A Research Roadmap,” *International Journal of Cooperative Information Systems*, vol. 17, no. 02, pp. 223–255, Jun. 2008.

[11] ISO/IEC, “ISO/IEC 25010 - Systems and software engineering - Systems and software Quality Requirements and Evaluation (SQuaRE) - System and software quality models,” ISO/IEC, 2010.

[12] ISO IEC, “ISO 9126/ISO, IEC (Hrsg.): International Standard ISO/IEC 9126: Information Technology-Software Product Evaluation.” 1991.

[13] Institute of Electrical and Electronics, “IEEE 610.12-1990: IEEE Standard Glossary of Software Engineering Terminology.” 1990.

[14] I. Burnstein, *Practical software testing: a process-oriented approach*, 1st ed. Springer, 2003.

[15] Q. Gu and P. Lago, “A stakeholder-driven service life cycle model for SOA,” in *2nd International Workshop on Service oriented software engineering in conjunction with the 6th ESEC/FSE joint meeting - IW-SOSWE ’07*, 2007, pp. 1–7.

[16] S. Benbernou, L. Cavallaro, M. Hacid, R. Kazhamiakin, G. Kecskemeti, J. Pazat, F. Silvestri, M. Uhlig, and B. Wetzstein, “Deliverable PO-JRA-1.2.1, State of the Art Report, Gap Analysis of Knowledge on Principles, Techniques and Methodologies for Monitoring and Adaptation of SBAs,” S-Cube Deliverable, 2009.

[17] A. Marconi, A. Bucchiarone, K. Bratanis, A. Brogi, J. Camara, D. Dranidis, H. Giese, R. Kazhamiakink, R. de Lemos, C. C. Marquezan, and A. Metzger, “Research challenges on multi-layer and mixed-initiative monitoring and adaptation for service-based systems,” in *2012 First International Workshop on European Software Services and Systems Research - Results and Challenges (S-Cube)*, 2012, pp. 40–46.

[18] F. Z. Safy, M. El-Ramly, and A. Salah, “Runtime Monitoring of SOA Applications: Importance, Implementations and Challenges,” in *2013 IEEE Seventh International Symposium on Service-Oriented System Engineering*, 2013, pp. 315–319.

[19] M. B. Juric, “A Hands-on Introduction to BPEL, Part 2: Advanced BPEL,” *Oracle*. [Online]. Available: http://www.oracle.com/technetwork/articles/matjaz-bpel2-082861.html.

[20] M. Shaw, “The coming-of-age of software architecture research,” in *Proceedings of the 23rd International Conference on Software Engineering (ICSE)*, 2001, p. 656.

[21] M. Oriol, “Quality of Service (QoS) in SOA Systems. A Systematic Review,” Master thesis, Universitat Politècnica de Catalunya, 2009.

[22] M. Oriol, J. Marco, and X. Franch, "Quality models for web services: A systematic mapping," *Information and Software Technology*, vol. 56, no. 10, pp. 1167–1182, 2014.

[23] D. Ameller and X. Franch, "Service Level Agreement Monitor (SALMon)," in *Proceedings of the 7th International Conference on Composition-Based Software Systems (ICCBSS)*, 2008, pp. 224–227.

[24] M. Oriol, J. Marco, X. Franch, and D. Ameller, "Monitoring Adaptable SOA-Systems using SALMon," in *Proceedings of the 1st International Workshop on Service Monitoring, Adaptation and Beyond (Mona+)*, 2008, pp. 19–28.

[25] O. Cabrera, M. Oriol, L. López, X. Franch, J. Marco, O. Fragoso, and R. Santaolaya, "WeSSQoS: Un sistema SOA para la selección de servicios web según su calidad," in *Proceeding of Jornadas Científico-Técnicas en Servicios Web y SOA (JSWEB)*, 2009, pp. 76–89.

[26] 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," in *arXiv, Technical Report*, 2011.

[27] O. J. Cabrera Bejar, M. Oriol Hilari, X. Franch, J. Marco, L. López, O. Fragoso, and R. Santaolaya, "Open Framework For Web Service Selection Using Multimodal and Configurable Techniques," *Computación y Sistemas*, vol. 18, no. 4, 2014.

[28] A. Kertesz, G. Kecskemeti, Z. Nemeth, M. Oriol, and X. Franch, "A holistic service provisioning solution for Federated Cloud infrastructures," in *1st International Workshop on European Software Services and Systems Research - Results and Challenges (S-Cube)*, 2012, pp. 25–26.

[29] A. Kertesz, G. Kecskemeti, A. Marosi, M. Oriol, X. Franch, and J. Marco, "Integrated Monitoring Approach for Seamless Service Provisioning in Federated Clouds," in *20th Euromicro International Conference on Parallel, Distributed and Network-based Processing*, 2012, pp. 567–574.

[30] A. Kertesz, G. Kecskemeti, M. Oriol, P. Kotcauer, S. Acs, M. Rodríguez, O. Mercè, A. C. Marosi, J. Marco, and X. Franch, "Enhancing Federated Cloud Management with an Integrated Service Monitoring Approach," *Journal of Grid Computing*, vol. 11, no. 4, pp. 699–720, 2013.

[31] C. Müller, M. Resinas, and A. Ruiz-Cortés, "Explaining the Non-compliance between Templates and Agreement Offers in WS-Agreement," in *7th International Joint Conference ICSOC-ServiceWave*, 2009, pp. 237–252.

[32] C. Muller, M. Oriol, M. Rodriguez, X. Franch, J. Marco, M. Resinas, and A. Ruiz-Cortes, "SALMonADA: A platform for monitoring and explaining violations of WS-agreement-compliant documents," in *4th International Workshop on Principles of Engineering Service-Oriented Systems (PESOS)*, 2012, pp. 43–49.

[33] C. Müller, M. Oriol, M. Rodriguez, X. Franch, J. Marco, M. Resinas, and A. Ruiz-Cortés, "SALMonADA: A Platform for Monitoring and Explaining Violations of WS-Agreement-compliant Documents," in *VIII Jornadas de Ciencia e Ingeniería de Servicios (JCIS)*, 2012, pp. 157–160.

[34] B. Burgstaller, D. Dhungana, X. Franch, P. Grunbacher, L. López, J. Marco, M. Oriol, and R. Stockhammer, "Monitoring and Adaptation of Service-oriented Systems with Goal and Variability Models," in *LSI, Technical Report.*, 2009.

[35] X. Franch, P. Grunbacher, M. Oriol, B. Burgstaller, D. Dhungana, L. Lopez, J. Marco, and J. Pimentel, “Goal-Driven Adaptation of Service-Based Systems from Runtime Monitoring Data,” in *35th Annual Computer Software and Applications Conference Workshops*, 2011, pp. 458–463.

[36] O. Sammodi, A. Metzger, X. Franch, M. Oriol, J. Marco, and K. Pohl, “Usage-Based Online Testing for Proactive Adaptation of Service-Based Applications,” in *35th Annual Computer Software and Applications Conference*, 2011, pp. 582–587.

[37] M. Oriol, N. A. Qureshi, X. Franch, A. Perini, and J. Marco, “Requirements Monitoring for adaptive service-based applications,” in *Requirements Engineering: Foundation for Software Quality (REFSQ*, 2012, pp. 280–287.

[38] M. Oriol, X. Franch, and J. Marco, “SALMon: A SOA System for Monitoring Service Level Agreements,” in *LSI, Technical Report.*, 2010.

[39] M. Oriol, O. Franco-Bedoya, X. Franch, and J. Marco, “Assessing Open Source Communities’ Health using Service Oriented Computing Concepts,” in *8th International Conference on Research Challenges in Information Science (RCIS)*, 2014, pp. 1–6.

[40] S. Martínez-Fernández, J. Bisbal, and X. Franch, “QuPreSS: A Service-Oriented Framework for Predictive Services Quality Assessment,” in *7th International Conference on Knowledge Management in Organizations*, 2012, pp. 525–536.

[41] S. Martínez-Fernández, X. Franch, and J. Bisbal, “Verifying Predictive Services’ Quality with Mercury,” in *International Workshop on*

Advanced/Academic Software Development Tools and Techniques (WASDETT), 2013, pp. 1–18.

[42] S. Martínez-Fernández, "Accuracy Assessment of forecasting services," Master Thesis, Universitat Politècnica de Catalunya, 2011.

[43] M. Oriol, X. Franch, and J. Marco, "Monitoring the service-based system lifecycle with SALMon," *Submitted to Expert Systems with Applications*, 2014.

[44] C. Muller, M. Oriol, X. Franch, J. Marco, M. Resinas, A. Ruiz-Cortes, and M. Rodriguez, "Comprehensive Explanation of SLA Violations at Runtime," *IEEE Transactions on Services Computing*, vol. 7, no. 2, pp. 168–193, 2013.

[45] E. Schmieders, A. Micsik, and M. Oriol, "Combining SLA prediction and cross layer adaptation for preventing SLA violations," in *2nd Workshop on Software Services: Cloud Computing and Applications based on Software Services*, 2011, pp. 1–6.

[46] B. Kitchenham and S. Charters, "Guidelines for performing systematic literature reviews in software engineering, Version 2.3," 2007.

[47] M. Oriol, X. Franch, and J. Marco, "Appendix of: quality models for web services, a systematic mapping," 2013. [Online]. Available: http://gessi.lsi.upc.edu/qmodels/.

[48] B. Kitchenham, P. Brereton, M. Turner, M. Niazi, S. Linkman, R. Pretorius, and D. Budgen, "The impact of limited search procedures for systematic literature reviews A participant-observer case study," in

Proceedings of the 3rd International Symposium on Empirical Software Engineering and Measurement, 2009, pp. 336–345.

[49] B. A. Kitchenham, E. Mendes, and G. H. Travassos, “Cross versus Within-Company Cost Estimation Studies: A Systematic Review,” *IEEE Transactions on Software Engineering.Software Eng.*, vol. 33, no. 5, pp. 316–329, 2007.

[50] M. Riaz, E. Mendes, and E. Tempero, “A systematic review of software maintainability prediction and metrics,” in *Proceedings of the 3rd International Symposium on Empirical Software Engineering and Measurement*, 2009, pp. 367–377.

[51] K. Petersen, R. Feldt, S. Mujtaba, and M. Mattsson, “Systematic mapping studies in software engineering,” in *Proceedings of the 12th International conference on Evaluation and Assessment in Software Engineering (EASE)*, 2008, pp. 68–77.

[52] A. Mani and A. Nagarajan, “Understanding quality of service for Web services,” 2002. [Online]. Available: http://www-106.ibm.com/developerworks/library/ws-quality.html.

[53] A. Gehlert and A. Metzger (eds.), “CD-JRA-1.3.2 Quality Reference Model for SBA,” S-Cube deliverable, 2009.

[54] OASIS, “Web Services Quality Factors Version 1.0,” 2012. [Online]. Available: http://docs.oasis-open.org/wsqm/wsqf/v1.0/WS-Quality-Factors.pdf.

[55] OASIS, “OASIS WS quality model TC - Quality model for Web services.,” 2005. [Online]. Available: http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=wsqm.

[56] W. Abramowicz, R. Hofman, W. Suryn, and D. Zyskowski, “SQuaRE based Web Services Quality Model,” in *Proceedings of the International MultiConference of Engineers and Computer Scientists 2008, Vol I IMECS*, 2008, pp. 827–835.

[57] B. Yin, H. Yang, P. Fu, and X. Chen, “A semantic web services discovery algorithm based on QoS ontology,” in *Proceedings of the 6th International conference on Active media technology*, 2010, pp. 166–173.

[58] O. Cabrera and X. Franch, “A quality model for analysing web service monitoring tools,” in *6th International Conference on Research Challenges in Information Science (RCIS)*, 2012, pp. 1–12.

[59] P. Nadanam and R. Rajmohan, “QoS evaluation for web services in cloud computing,” in *3rd International Conference on Computing, Communication and Networking Technologies (ICCCNT)*, 2012, pp. 1–8.

[60] S. Ran, “A model for web services discovery with QoS,” *ACM SIGecom Exchanges*, vol. 4, no. 1, pp. 1–10, 2003.

[61] E. M. Maximilien and M. P. Singh, “A framework and ontology for dynamic Web services selection,” *IEEE Internet Computing*, vol. 8, no. 5, pp. 84–93, 2004.

[62] K. Lee, J. Jeon, W. Lee, S. Jeong, and S. Park, “QoS for Web Services: Requirements and Possible Approaches,” *W3C Working group Note 25*,

2003. [Online]. Available: http://www.w3c.or.kr/kr-office/TR/2003/ws-qos/.

[63] G. Dobson, R. Lock, and I. Sommerville, "QoSOnt: a QoS ontology for service-centric systems," in *Proceedings of the 31st EUROMICRO Conference on Software Engineering and Advanced Applications*, 2005, pp. 80–87.

[64] S. Jiang and F. Aagesen, "An Approach to Integrated Semantic Service Discovery," in *Proceedings of the 1st IFIP TC6 international conference on Autonomic Networking*, 2006, pp. 159–171.

[65] V. X. Tran, H. Tsuji, and R. Masuda, "A new QoS ontology and its QoS-based ranking algorithm for Web services," *Dependable Service-Orientated Computing Systems*, vol. 17, no. 8, pp. 1378–1398, 2009.

[66] V. X. Tran, "WS-QoSOnto: A QoS Ontology for Web Services," in *IEEE International Symposium on Service-Oriented System Engineering (SOSE)*, 2008, pp. 233–238.

[67] S. Hanna and A. Alalawneh, "An Ontology for the Quality Attributes of Web Services," in *13th IBIMA Conference*, 2009.

[68] A. Avizienis, J.-C. Laprie, B. Randell, and C. Landwehr, "Basic concepts and taxonomy of dependable and secure computing," *IEEE Transactions on Dependable and Secure Computing*, vol. 1, no. 1, pp. 11–33, 2004.

[69] H. Truong, R. Samborski, and T. Fahringer, "Towards a Framework for Monitoring and Analyzing QoS Metrics of Grid Services," in *2nd IEEE International Conference on e-Science and Grid Computing (e-Science)*, 2006, pp. 65–65.

[70] H. Muñoz Frutos, I. Kotsiopoulos, L. M. Vaquero Gonzalez, and L. Rodero Merino, “Enhancing Service Selection by Semantic QoS,” in *Proceedings of the 6th European Semantic Web Conference on The Semantic Web: Research and Applications*, 2009, pp. 565–577.

[71] H. Muñoz Frutos, A. Micsik, D. Fellows, J. Leukel, R. Woitsch, H. Eichner, B. Cantalupo, and A. Giuseppe, “Final Report on BREIN-Core Ontologies D3.2.5,” 2008. [Online]. Available: http://www.eu-brein.com/index.php?option=com_docman&task=doc_view&gid=57.

[72] X. Wang, T. Vitvar, M. Kerrigan, and I. Toma, “A qos-aware selection model for semantic web services,” in *Proceedings of the 4th International conference on Service-Oriented Computing*, 2006, pp. 390–401.

[73] M. Comuzzi and B. Pernici, “A framework for QoS-based Web service contracting,” *ACM Transactions on the Web*, vol. 3, no. 3, pp. 10:1–10:52, 2009.

[74] G. Yeom, T. Yun, and D. Min, “QoS Model and Testing Mechanism for Quality-driven Web Services Selection,” in *Proceedings of the 4th IEEE Workshop on Software Technologies for Future Embedded and Ubiquitous Systems, and the 2nd International Workshop on Collaborative Computing, Integration, and Assurance (SEUS-WCCIA)*, 2006, pp. 199–204.

[75] D. T. Tsesmetzis, I. G. Roussaki, I. V Papaioannou, and M. E. Anagnostou, “QoS awareness support in Web-Service semantics,” in *International Conference on Internet and Web Applications and Services/Advanced International Conference onTelecommunications. AICT-ICIW*, 2006, p. 128.

[76] I. V Papaioannou, D. T. Tsesmetzis, I. G. Roussaki, and M. E. Anagnostou, “A qos ontology language for web-services,” in *Proceedings of the 20th International Conference on Advanced Information Networking and Applications*, 2006.

[77] D. Z. G. Garcia and M. B. F. de Toledo, “Semantics-enriched QoS policies for web service interactions,” in *Proceedings of the 12th Brazilian Symposium on Multimedia and the web*, 2006, pp. 35–44.

[78] A. Soomro and W. Song, “Developing and Managing SLAs for Business Applications in Information Systems,” in *Emerging Trends and Applications in Information Communication Technologies*, 2012, pp. 489–500.

[79] J. Brujin, C. Bussler, J. Domingue, D. Fensel, M. Hepp, M. Kifer, B. König-Ries, J. Kopecky, R. Lara, E. Oren, A. Polleres, J. Scicluna, and M. Stollberg, “D2v1.4. Web Service Modeling Ontology (WSMO),” *WSMO Working Draft*, 2007. [Online]. Available: http://www.wsmo.org/TR/d2/v1.4/.

[80] J. Brujin, C. Bussler, J. Domingue, D. Fensel, M. Hepp, U. Keller, M. Kifer, B. König-Ries, J. Kopecky, R. Lara, H. Lausen, E. Oren, A. Polleres, D. Roman, J. Scicluna, and M. Stollberg, “Web service modeling ontology (WSMO),” *W3C Member Submission*. 2005.

[81] C. Patel, K. Supekar, and Y. Lee, “Provisioning resilient, adaptive Web services-based workflow: a semantic modeling approach,” in *Proceedings of IEEE International Conference on Web Services*, 2004, pp. 480–487.

[82] C. Patel, K. Supekar, and Y. Lee, “A QoS Oriented Framework for Adaptive Management of Web Service based Workflows,” in *Proceeding of Database and Expert Systems Conference*, 2003, pp. 826–835.

[83] N. Looker, M. Munro, and J. Xu, "Assessing Web Service Quality of Service with Fault Injection," in *Workshop on Quality of Service for Application Servers*, 2004.

[84] K. Ren, J. Chen, T. Chen, J. Song, and N. Xiao, "Grid-Based Semantic Web Service Discovery Model with QoS Constraints," in *3rd International Conference on Semantics, Knowledge and Grid*, 2007, pp. 479–482.

[85] W. D. Yu, R. B. Radhakrishna, S. Pingali, and V. Kolluri, "Modeling the Measurements of QoS Requirements in Web Service Systems," *Simulation*, vol. 83, no. 1, pp. 75–91, 2007.

[86] Y. Kang, "Extended Model Design for Quality Factor Based Web Service Management," *Future Generation Communication and Networking (FGCN)*, vol. 2, pp. 484–487, 2007.

[87] N. Artaiam and T. Senivongse, "Enhancing Service-Side QoS Monitoring for Web Services," in *9th ACIS International Conference on Software Engineering, Artificial Intelligence, Networking, and Parallel/Distributed Computing (SNPD)*, 2008, pp. 765–770.

[88] E. Giallonardo and E. Zimeo, "More semantics in qos matching," in *IEEE International Conference on Service-Oriented Computing and Applications (SOCA)*, 2007, pp. 163–171.

[89] G. Damiano, E. Giallonardo, and E. Zimeo, "onQoS-QL: A Query Language for QoS-Based Service Selection and Ranking," in *Service-Oriented Computing-ICSOC 2007 Workshops*, 2009, p. 127.

[90] H. Chang and K. Lee, “Quality-Driven Web Service Composition for Ubiquitous Computing Environment,” in *International Conference on New Trends in Information and Service Science (NISS)*, 2009, pp. 156–161.

[91] E. Al-Masri and Q. H. Mahmoud, “Understanding web service discovery goals,” in *IEEE International Conference on Systems, Man and Cybernetics (SMC)*, 2009, pp. 3714–3719.

[92] E. Al-Masri and Q. H. Mahmoud, “Identifying Client Goals for Web Service Discovery,” in *IEEE International Conference on Services Computing*, 2009, pp. 202–209.

[93] H. Tong, J. Cao, S. Zhang, and Y. Mou, “A fuzzy evaluation system for web services selection using extended QoS model,” *Kybernetes*, vol. 38, no. 3/4, pp. 513–521, 2009.

[94] Z. Balfagih and M. F. Hassan, “Quality Model for Web Services from Multi-stakeholders’ Perspective,” in *Proceedings of the International Conference on Information Management and Engineering*, 2009, pp. 287–291.

[95] S. Li and J. Zhou, “The WSMO-QoS Semantic Web Service Discovery Framework,” in *International Conference on Computational Intelligence and Software Engineering*, 2009, pp. 1–5.

[96] K. K. Reddy, K. Maralla, G. Raj Kumar, and M. Thirumaran, “A greedy approach with criteria factors for QoS based web service discovery,” in *Proceedings of the 2nd Bangalore Annual Compute Conference - COMPUTE*, 2009, pp. 12:1–12:5.

[97] R. Mohanty, V. Ravi, and M. R. Patra, “Web-services classification using intelligent techniques,” *Expert Systems with Applications*, vol. 37, no. 7, pp. 5484–5490, 2010.

[98] H. Yang, X. Chen, and S. Liu, “Research and Implementation on QoS Ontology of Web Service-Oriented Composition,” in *2nd International Symposium on Information Engineering and Electronic Commerce*, 2010, pp. 1–4.

[99] Y. Baocai, Y. Huirong, F. Pengbin, G. Liheng, and L. Mingli, “A framework and QoS based web services discovery,” in *IEEE International Conference on Software Engineering and Service Sciences*, 2010, pp. 755–758.

[100] Z. Pan and J. Baik, “A QOS enhanced framework and trust model for effective web services selection,” *Journal of Web Engineering*, vol. 9, no. 2, pp. 186–204, 2010.

[101] Z. Shu and S. Meina, “An architecture design of life cycle based SLA management,” in *12th International Conference on Advanced Communication Technology (ICACT)*, 2010, pp. 1351–1355.

[102] Y. Lee, “Quality Control for Business Collaboration Based on SOA Framework,” in *International Conference on Convergence Information Technology (ICCIT)*, 2007, pp. 1963–1968.

[103] Y. Lee and G. Yeom, “A Quality Chain Modeling Methodology for Ternary Web Services Quality View,” in *5th ACIS International Conference on Software Engineering Research, Management & Applications (SERA)*, 2007, pp. 91–97.

[104] Y. Lee and G. Yeom, “A Research for Web Service Quality Presentation Methodology for SOA Framework,” in *6th International Conference on Advanced Language Processing and Web Information Technology (ALPIT)*, 2007, pp. 434–439.

[105] Y. Lee and G. Yeom, “A Research for Quality Preservation Methodology for SOA Environment,” in *8th ACIS International Conference on Software Engineering, Artificial Intelligence, Networking, and Parallel/Distributed Computing (SNPD)*, 2007, vol. 2, pp. 429–434.

[106] Youngkon Lee, “QoS metrics for service level measurement for SOA environment,” in *6th International Conference on Advanced Information Management and Service (IMS)*, 2010, pp. 509–514.

[107] P. Bocciarelli and A. D’Ambrogio, “A model-driven method for describing and predicting the reliability of composite services,” *Software & Systems Modeling*, vol. 10, no. 2, pp. 265–280, 2011.

[108] A. D’Ambrogio, “A Model-driven WSDL Extension for Describing the QoS ofWeb Services,” in *International Conference on Web Services (ICWS)*, 2006, pp. 789–796.

[109] J. Qiu and F. Yu, “Research on Semantic Dynamic Service Combination Module,” in *International Conference on Internet Technology and Applications*, 2011, pp. 1–4.

[110] N. Debnath, P. Martellotto, M. Daniele, D. Riesco, and G. Montejano, “A method to evaluate QoS of web services required by a workflow,” in *11th International Conference on ITS Telecommunications*, 2011, pp. 640–645.

[111] F. Rosenberg, C. Platzer, and S. Dustdar, “Bootstrapping Performance and Dependability Attributes ofWeb Services,” in *International Conference on Web Services (ICWS)*, 2006, pp. 205–212.

[112] O. Moser, F. Rosenberg, and S. Dustdar, “Domain-Specific Service Selection for Composite Services,” *IEEE Transactions on Software Engineering*, vol. 38, no. 4, pp. 828–843, 2012.

[113] R. Phalnikar and P. A. Khutade, “Survey of QoS based web service discovery,” in *World Congress on Information and Communication Technologies*, 2012, pp. 657–661.

[114] W. N. Wan Ab. Rahman and F. Meziane, “The Awareness of QoS Consideration for Web Services,” in *International Business Information Management Association (IBIMA)*, 2012, pp. 509–516.

[115] J. Pablo Carvallo, X. Franch, and C. Quer, “Managing Non-Technical Requirements in COTS Components Selection,” in *Proceedings of the 14th IEEE International Requirements Engineering Conference*, 2006, pp. 316–321.

[116] M. Oriol, X. Franch, and J. Marco, “Details on SALMon - State of the Art,” 2014. [Online]. Available: http://gessi.lsi.upc.edu/salmon/slr.

[117] Z. Zheng, Y. Zhang, and M. R. Lyu, “Distributed QoS Evaluation for Real-World Web Services,” in *IEEE International Conference on Web Services*, 2010, pp. 83–90.

[118] G. Kang, J. Liu, M. Tang, X. (Frank) Liu, and K. K. Fletcher, “Web Service Selection for Resolving Conflicting Service Requests,” in *IEEE International Conference on Web Services (ICWS)*, 2011, pp. 387–394.

[119] S. Wang, Q. Sun, and F. Yang, “Quality of service measure approach of web service for service selection,” *IET Software*, vol. 6, no. 2, p. 148, 2012.

[120] A. Michlmayr, F. Rosenberg, P. Leitner, and S. Dustdar, “End-to-End Support for QoS-Aware Service Selection, Binding, and Mediation in VRESCo,” *IEEE Transactions on Services Computing*, vol. 3, no. 3, pp. 193–205, Jul. 2010.

[121] E. Di Nitto, C. Ghezzi, A. Metzger, M. Papazoglou, and K. Pohl, “A journey to highly dynamic, self-adaptive service-based applications,” *Automated Software Engineering*, vol. 15, no. 3–4, pp. 313–341, Sep. 2008.

[122] D. Michell Smith, G. Petri, Y. V. Natis, D. Scott, M. Warrilow, J. Heiser, A. Bona, D. Toombs, M. Yeates, T. Bova, and B. J. Lheureux, “Predicts 2014: Cloud Computing Affects All Aspects of IT,” Gartner document, 2013.

[123] W. Gentzsch, L. Grandinetti, G. Joubert, L. Ricci, R. Baraglia, J. L. Lucas-Simarro, R. Moreno-Vozmediano, R. S. Montero, and I. M. Llorente, “Scheduling strategies for optimal service deployment across multiple clouds,” *Future Generation Computer Systems*, vol. 29, no. 6, pp. 1431–1441, 2013.

[124] A. C. Marosi, G. Kecskemeti, A. Kertesz, and P. Kacsuk, “FCM: an Architecture for Integrating IaaS Cloud Systems,” in *2nd International Conference on Cloud Computing, GRIDs, and Virtualization*, 2011, pp. 7–12.

[125] Q. Zhang, L. Cheng, and R. Boutaba, “Cloud computing: state-of-the-art and research challenges,” *Journal of Internet Services and Applications*, vol. 1, no. 1, pp. 7–18, 2010.

[126] L. Baresi, S. Guinea, M. Pistore, and M. Trainotti, “Dynamo + Astro: An Integrated Approach for BPEL Monitoring,” in *IEEE International Conference on Web Services (ICWS)*, 2009, pp. 230–237.

[127] S. Guinea and G. Kecskemeti, “Multi-layered monitoring and adaptation,” *9th International Conference on Service-Oriented Computing (ICSOC)*, pp. 359–373, 2011.

[128] O. Moser, F. Rosenberg, and S. Dustdar, “Non-intrusive monitoring and service adaptation for WS-BPEL,” *Proceeding of the 17th international conference on World Wide Web - WWW ’08*, p. 815, 2008.

[129] O. Moser, F. Rosenberg, and S. Dustdar, “Event Driven Monitoring for Service Composition Infrastructures,” in *Web Information Systems Engineering – WISE 2010*, 2010, pp. 38–51.

[130] L. Baresi, S. Guinea, O. Nano, and G. Spanoudakis, “Comprehensive Monitoring of BPEL Processes,” *IEEE Internet Computing*, vol. 14, no. 3, pp. 50–57, 2010.

[131] L. Baresi and S. Guinea, “Self-Supervising BPEL Processes,” *IEEE Transactions on Software Engineering*, vol. 37, no. 2, pp. 247–263, 2011.

[132] K.-J. Lin, M. Panahi, Y. Zhang, J. Zhang, and S.-H. Chang, “Building Accountability Middleware to Support Dependable SOA,” *IEEE Internet Computing*, vol. 13, no. 2, pp. 16–25, 2009.

[133] M. Comuzzi, C. Kotsokalis, G. Spanoudakis, and R. Yahyapour, “Establishing and Monitoring SLAs in Complex Service Based Systems,” in *IEEE International Conference on Web Services (ICWS)*, 2009, pp. 783–790.

[134] G. Ortiz and B. Bordbar, “Aspect-Oriented Quality of Service for Web Services: A Model-Driven Approach,” in *IEEE International Conference on Web Services*, 2009, pp. 559–566.

[135] L. Fei, Y. Fangchun, S. Kai, and S. Sen, “A Policy-Driven Distributed Framework for Monitoring Quality of Web Services,” in *IEEE International Conference on Web Services (ICWS)*, 2008, pp. 708–715.

[136] K. Mahbub, G. Spanoudakis, and A. Zisman, “A monitoring approach for runtime service discovery,” *Automated Software Engineering*, vol. 18, no. 2, pp. 117–161, 2010.

[137] M. Psiuk, T. Bujok, and K. Zieliski, “Enterprise Service Bus Monitoring Framework for SOA Systems,” *IEEE Transactions on Services Computing*, vol. 5, no. 3, pp. 450–466, 2012.

[138] Q. Wang, J. Shao, F. Deng, Y. Liu, M. Li, J. Han, and H. Mei, “An Online Monitoring Approach for Web Service Requirements,” *IEEE Transactions on Services Computing*, vol. 2, no. 4, pp. 338–351, 2009.

[139] A. Benharref, R. Dssouli, M. A. Serhani, and R. Glitho, “Efficient traces’ collection mechanisms for passive testing of Web Services,” *Information and Software Technology*, vol. 51, no. 2, pp. 362–374, 2009.

[140] F. Raimondi, J. Skene, and W. Emmerich, “Efficient online monitoring of web-service SLAs,” *Proceedings of the 16th ACM SIGSOFT International Symposium on Foundations of software engineering - SIGSOFT ’08/FSE-16*, pp. 170–180, 2008.

[141] G. Katsaros, R. Kübert, and G. Gallizo, "Building a Service-Oriented Monitoring Framework with REST and Nagios," in *IEEE International Conference on Services Computing*, 2011, pp. 426–431.

[142] G. Copil, D. Moldovan, H.-L. Truong, and S. Dustdar, "SYBL+MELA: Specifying, Monitoring, and Controlling Elasticity of Cloud Services," in *Service-Oriented Computing*, 2013, vol. 8274, pp. 679–682.

[143] A. Bertolino, A. Calabrò, and G. De Angelis, "A generative approach for the adaptive monitoring of SLA in service choreographies," in *Proceedings of the 13th international conference on Web Engineering*, 2013, vol. 7977, pp. 408–415.

[144] A. Andrieux, K. Czajkowski, A. Dan, K. Keahey, H. Ludwig, J. Pruyne, J. Rofrano, S. Tuecke, and M. Xu, "Web services agreement specification (WS-Agreement)," in *Global Grid Forum*, 2004.

[145] R. de Lemos, H. Giese, H. Müller, M. Shaw, and et al., "Software Engineering for Self-Adaptive Systems: A second Research Roadmap," in *Software Engineering for Self-Adaptive Systems*, Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2011, pp. 1–32.

[146] SAP, "Standardized Technical Architecture Modeling, Conceptual and Design Level," 2007. [Online]. Available: http://www.fmc-modeling.org/download/fmc-and-tam/SAP-TAM_Standard.pdf. [Accessed: 27-Jan-2014].

[147] S. H. Krzysztof Czarnecki, "Classification of Model Transformation Approaches," in *2nd OOPSLA'03 Workshop on Generative Techniques in the Context of MDA*, 2003, pp. 1–17.

[148] M. Oriol, J. Marco, and X. Franch, “SALMon evaluation,” 2014. [Online]. Available: http://gessi.upc.edu/salmon/evaluation.

[149] D. M. Hawkins, *Identification of Outliers*, 1st ed. Dordrecht: Springer Netherlands, 1980.

[150] R. Ratcliff, “Methods for dealing with reaction time outliers,” *Psychological Bulletin*, vol. 114, no. 3, pp. 510–532, 1993.