Departament de Llenguatges i Sistemes Informatics
UNIVERSITAT POLITÈCNICA DE CATALUNYA

Master in Computing

Master of Science Thesis

ACCURACY ASSESSMENT OF FORECASTING SERVICES

Silverio Martínez-Fernández

Advisors: Dr. Xavier Franch Gutiérrez
Dr. Jesús Bisbal Riera

05/09/2011

Acknowledgements

I would like to thank to all people that have helped me while working in this master thesis.

First of all, I want to thank my advisors, Dr. Xavier Franch and Dr. Jesús Bisbal, for their support and guidance. Our meetings and discussions made this master thesis possible. They gave me the chance of discovering the great world of research. I also want to thank Marc Oriol and Marc Rodríguez for their friendly cooperation while integrating the tool with SALMon.

I devote this master thesis to my family for their love and affection. Especially to my sister, whose wise advises help me in difficult times. They are always there to encourage me and believing in me.

I am especially grateful to Laia for triggering my decision of coming to the UPC and with who I shared all these great days.

I would also like to thank my friends, especially to Jesús, Jovan and Petar, who have given me a hand whenever I needed.

This work has been totally supported by *"la Caixa"* Foundation under the program of master grants in Spain.

Table of contents

Index of figures vii
Index of tables ix
Introduction 1
Chapter 1 Introduction 3
1.1 Motivation 3
1.2 Thesis description 4
1.3 Contribution 5
1.4 Thesis organisation 5
Part I. Systematic literature review 7
Chapter 2 Systematic review definition 9
2.1 Introduction 9
2.2 Background 10
2.3 Review methods 12
Chapter 3 Systematic review results 25
3.1 Service oriented architectures 25
3.2 Testing and monitoring in SOA 33
3.3 Prediction in SOA 39
3.4 Criteria to evaluate predictions 47
Chapter 4 Review discussion and conclusions 51
4.1 Discussion 51
4.2 Conclusion 53
Part II. Architecture 55
Chapter 5 Specification 57
5.1 General functional requirements 57
5.2 Non-functional requirements 60
5.3 Use case diagram 62
5.4 Conceptual modelling of the weather forecast domain 62
Chapter 6 Architecture 65
6.1 Proposed architecture 65
Part III. Tool 71
Chapter 7 Development of the tool 73
7.1 External sources 73
7.2 Database diagram 81
7.3 Forecast verifier 83
7.4 Details about the implementation 84
Chapter 8 Testing 86
8.1 Testing 86
8.2 Problems during the implementation 89

Part IV. Final remarks 91
Chapter 9 Conclusions 93
9.1 Conclusion 93
9.2 Future work 94
Bibliography 95
References 97
Appendix 103
Appendix A: Glossary 105
Appendix B: User manual and demo 107
1. Page of observations 107
2. Page of predictions made in a specified day 109
3. Page of predictions made for a specified day 112
Appendix C: Administrator manual 117

Index of figures

Figure 1.- Systematic literature review process [6]. 9
Figure 2.- The selection strategy for the primary studies. 20
Figure 3.- Number of included studies per year. 21
Figure 4.- Set of technologies for implementing services [21]. 27
Figure 5.- Lookup of services in a SOA [23] 28
Figure 6.- SOC research road map [22]. 31
Figure 7.- High-level view of a runtime monitor [27]. 35
Figure 8.- Categorisation of formal testing approaches for web services. 37
Figure 9.- SALMon Architecture [4]. 38
Figure 10.- Real-time automated forecast model verification system [19]. 40
Figure 11.- SWE Sensor Observation Service as a composition tool [19]. 41
Figure 12.- The network architecure of SMAFS [32]. 43
Figure 13.- The heterogonous data integration platform in SMAFS [32]. 44
Figure 14.- An overview of the R Web service infrastructure [28]. 44
Figure 15.- Activity diagram for MAPS methodology [30]. 46
Figure 16.- Wavelet network schema to forecast the JVM heap memory usage [31]. 47
Figure 17.- Use case diagram. 62
Figure 18.- Conceptual schema. 63
Figure 19.- General architecture for accuracy assessment of forecasting services. 66
Figure 20.- Specific architecture for the domain of weather forecast. 67
Figure 21.- Setup Catalan cities sequence diagram. 68
Figure 22.- Start monitoring sequence diagram. 68
Figure 23.- Start to parser invocations sequence diagram. 69
Figure 24.- See accuracy assessment sequence diagram. 69
Figure 25.- Diagram of ground truth database. 82
Figure 26.- Diagram of forecast data database. 82
Figure 27.- Main screen of the forecast verifier application. 107
Figure 28.- Form of observations menu. 108
Figure 29.- Example of a table with observations. 108
Figure 30.- Example of a graph with observations. 108
Figure 31.- Form of the predictions made in a specified day page. 109
Figure 32.- Example of a table with predictions made a specified day. 109
Figure 33.- Graph with high temperatures in the predictions made a specified day page. 110
Figure 34.- Graph with low temperatures in the predictions made a specified day page. 110
Figure 35.- Errors in the predictions made in a specified day page. 111
Figure 36.- Table with mean squared errors of previous 5 days. 111
Figure 37.- Line chart with mean squared errors of previous 5 days. 112
Figure 38.- Form of the page to consult predictions made for a specified day. 112

Figure 39.- Table with all the forecasts that have been made for a specified day. 113
Figure 40.- Chart with high temperatures in the predictions made for specified day page. 114
Figure 41.- Chart with low temperatures in the predictions made for specified day page... 114
Figure 42.- Errors in the predictions made in for specified day page. 115
Figure 43.- Table with MSEs in the predictions made in for specified day page. 115
Figure 44.- Graph with MSEs in the predictions made in for specified day page. 115

Index of tables

Table 1.- Conferences and Journals in the first iteration of the search process. 14
Table 2.- Electronic databases that were used in the SLR. 15
Table 3.- Journals and Workshops in the second iteration of the search process. 16
Table 4.- Search sources, obtained and included primary studies. 20
Table 5.- Included primary studies 21
Table 6.- Quality assessment form. 22
Table 7.- Quality assessment of the selected studies. 22
Table 8.- Data extraction form. 23
Table 9.- Names and short definitions of three types of goodness [18]. 48
Table 10.- Short definitions for various aspects of forecast quality [18]. 49
Table 11.- Functional requirement #1 57
Table 12.- Functional requirement #1.1 57
Table 13.- Functional requirement #2 57
Table 14.- Functional requirement #2.1 58
Table 15.- Functional requirement #3 58
Table 16.- Functional requirement #4 58
Table 17.- Functional requirement #5 58
Table 18.- Functional requirement #2.2 59
Table 19.- Functional requirement #4.1 59
Table 20.- Functional requirement #6 59
Table 21.- Functional requirement #6.1 59
Table 22.- Non-functional requirement #1. 60
Table 23.- Non-functional requirement #2. 60
Table 24.- Non-functional requirement #3. 60
Table 25.- Non-functional requirement #4. 60
Table 26.- Non-functional requirement #5. 61
Table 27.- Non-functional requirement #5.1. 61
Table 28.- Non-functional requirement #6. 61
Table 29.- Non-functional requirement #7. 61
Table 30.- Proxies' Type of Response. 74
Table 31.- Example of GetLocationListResponse from Weather Bug Web Service 75
Table 32.- Example of GetForecastByCityCode from Weather Bug Web Service 75
Table 33.- Yahoo! Weather RSS Feed Response. 76
Table 34.- Example of XML of county forecast from Meteocat open data. 77
Table 35.- Example of XML file with forecasts from AEMET. 79
Table 36.- Comparative table of information offered by forecasting services. 81
Table 37.- Test Case Planning Template [57]. 87
Table 38.- Test case for AEMET Service. 87

Table 39.- Example of Test Case for the Forecast Verifier web application. 88
Table 40.- Example of Test Case for the forecasting data collector service. 88
Table 41.- Glossary 105
Table 42.- Forecasting data collector service's WSDL. 118
Table 43.- Aemet weather proxy service's WSDL. 121

Introduction

Chapter 1 Introduction

This document represents the master thesis on the Master in Computing, at Barcelona School of Informatics (Facultat d'Informàtica de Barcelona) of the Technical University of Catalonia (Universitat Politècnica de Catalunya). It has been done inside the service monitoring research area of the Group of Software Engineering for Information Systems (GESSI).

1.1 Motivation

A service system is a dynamic configuration of people, technologies, organisations and shared information that create and deliver value to customers and other stakeholders [1]. The following cases are examples of customers receiving a service: taking a bus to go somewhere, or going to a restaurant to have a meal, or for a small IT (information technology) company, contracting a service to a bigger one in order to save costs and time.

Service-oriented architecture (SOA) has become more popular during last years. Basically, this emerging development paradigm allows service providers to offer loosely coupled services. These services are normally only owned by the providers. As a result, the service user or client does not have to worry about the development, maintenance, infrastructure, or any other issue of how the service is working. To sum up, the user just has to find and choose the proper service.

On the one hand, it presents several advantages. Firstly, common functionality can be contracted as a service in order to be able to focus on the own core missions. Secondly, it decreases the cost, since it is cheaper to contract a service than creating it yourself. Thirdly, clients take benefit of provider's latest technologies.

On the other hand, there is one big drawback: lack of trust. When you contract a service, you lose the direct control, the provider has access to your own data, you depend on him, and you experiment delays since your functionality is not working in-home.

That is why the user has to decide previously which service is the most appropriate for his needs. Each client has different needs: quality (it varies among services), reputation (a famous or recommended provider usually gives more confidence), speed (agreements not to break thresholds), security (contract and trust in the provider), personalisation (preferential treatment from the provider), and locality (law is not the same in all countries). Therefore, a customer needs to know about the best service(s).

Among all kind of services, we concentrate on forecasting services. Forecasting services show in advance a condition or occurrence about the future. There are plenty of domains: weather forecasts, stock market prices, results in betting shops, elections…

Let us see a domain which is really familiar to all of us: weather forecast. When we are planning to travel, going somewhere or just deciding what to wear first thing in the morning, we wonder about weather conditions. To make these decisions, we check the weather forecast on TV news, a thermometer, or on a web site. However, sometimes we check several predictions and they do not agree. Which one will be the most accurate?

Our goal in this master thesis is to assess the accuracy of these forecasting services in order to help prospective users to choose the best one according to their needs. To do it, we are going to compare forecast predictions with actual real observations.

1.1.1 Starting point

We do not start from scratch. SALMon [2] [3] [4] is a system for monitoring services. A monitor is a tool which observes the behaviour and predictions of services. The idea is to use SALMon SOA System to monitor forecasting web services. The function of SALMon is to call these forecasting services in a systematic manner, and save their responses. Service's responses enable our system to obtain forecast predictions and real observations.

We are going to extend the architecture of SALMon. SALMon serves to get data from forecasting services and the extension is going to be used for determining their quality and accuracy.

1.2 Thesis description

This master thesis consists of one initial approach to evaluate the accuracy of forecasting services (such as weather forecast, stock market prediction and prediction of results in betting shops). It consists of three main parts which are:

(1) to make a systematic literature review (state-of-the-art) to identify the existing basis of this topic;
(2) to propose an architecture which deals with the accuracy of forecasting services and fits into the current body of knowledge;
(3) to monitor at least two web services of weather forecast, using SALMon SOA System, as a proof-of-concept.

Throughout this master thesis, we will study different domains in which previous research has been carried out.

1.3 Contribution

There are two main contributions in this research oriented master thesis:

First of all, we have thoroughly analysed the literature by means of a systematic literature review (SLR). We can conclude that our topic is still immature since there is little related work. It could be because SOA is a relatively new paradigm.

Secondly, we propose a general service-oriented architecture which monitories several forecasting services and assess them comparing forecast values with real data from observations. Moreover, we have implemented this architecture for the weather forecast domain to prove it.

1.4 Thesis organisation

The rest of the documentation covers the three main parts, which were mentioned in the section 1.2, as follows.

Part I includes the state-of-the-art in accuracy assessment of forecasting services. It is in three chapters. Chapter 2 defines the systematic literature review that has been done to evaluate and interpret all available research relevant to our phenomenon of interest. Chapter 3 includes the results which have been obtained from the primary studies. Finally, Chapter 4 discusses and summarises the findings of the systematic literature review.

Part II presents the architecture which cope with the accuracy assessment of forecasting services. Chapter 5 consists of the specification of the architecture with functional and non-functional requirements. Chapter 6 describes and presents the architecture.

Part III is the follow-up of part II. At this stage, the architecture already presented is instantiated as a proof-of-concept for the weather forecasts domain. Chapter 7 reports in detail the development of the tool. Chapter 8 contains the tool testing.

In order to sum up, Chapter 9 of Part IV concludes the documentation with some final remarks and future work.

Additionally, appendixes contain a glossary, the user manual with a demo and the administrator guide.

Part I. Systematic literature review

Chapter 2 Systematic review definition

2.1 Introduction

We have done a Systematic Literature Review (SLR) following the guidelines of [5]. A SLR is a way to identify, evaluate and interpret all available research to a particular research question, or topic area, or phenomenon of interest. SLR is an adaptation of medical guidelines to the need of software engineering research.

In this master thesis, a SLR is undertaken for the following reasons:

- to summarise the existing evidence concerning our topic,
- to identify gaps in current research that could lead to new areas for further investigation,
- to provide a background in order to appropriately position our research activity.

The importance of SLRs is that they provide scientific value by reviewing thoroughly and fairly the literature.

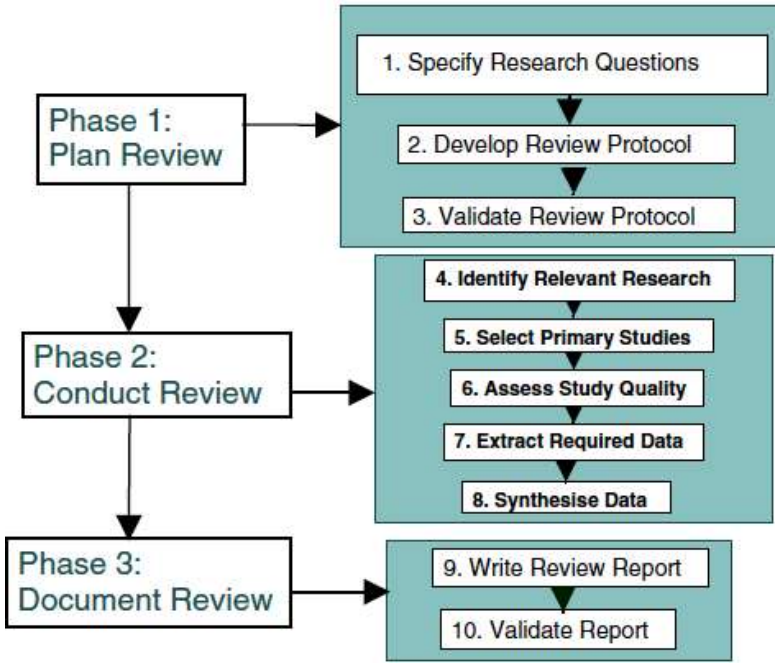

Figure 1.- Systematic literature review process [6].

We have followed the process of Figure 1 during the SLR. It has three main phases [5]:

1. Planning the review. Its stages are:
    a. Identification of the need for a review.

   b. Development of a review protocol.
2. Conducting the review. Its stages are:
   a. Identification of research.
   b. Selection of primary studies.
   c. Study quality assessment.
   d. Data extraction and monitoring.
   e. Data synthesis.
3. Reporting the review. It is a single stage phase.

The process has an iterative nature. Despite the stages listed above may seem to be sequential, many of the stages involve iteration. For instance, we had iteration during the search strategy to gradually improve it.

This SLR is reported with the structure and contents suggested in [5]. In this chapter, we define the SLR. Then, Chapter 3 includes the results. Finally, discussion about the results and conclusions are presented in Chapter 4.

2.2 Background

In this section, the first part of the protocol for the systematic review is identified. This first part consists of the rationale of the survey and the research questions that the review is intended to answer.

2.2.1 Justification of the need for the review

We need a summary with all existing information about our phenomenon of interest in a thorough and unbiased manner. It is the prelude to the research activity in this master thesis. Before undertaking the systematic review, we have to check if it has already been done.

Our subject of study is forecast verification in service oriented architectures. We are going to check if there are any existing state-of-the-art, review (systematic or not) or survey in the literature. Our search is:

"forecast verification" AND "service oriented" AND ("state of the art" OR review OR survey)

Unfortunately, the above search did not find any records in any of the databases described in the section 2.3.1. As a result, we concluded that a systematic literature review is needed.

2.2.2 Summary of previous reviews

There are not previous reviews for this specialised topic.

2.2.3 Review questions

The aim of this review consists of identifying the existing basis for forecasting services (such as weather forecast, stock market prediction and prediction of results in betting shops) in order to measure how accurate their predictions are. This state-of-the-art could lead to a proposal which deals with the accuracy of forecasting services and fits into the current body of knowledge.

The objectives of this review have been defined as explained in [6]. They are:

RQ1. to identify if there are frameworks which measure how accurate forecasting (web) services are;
RQ2. to determine which are the main quality criteria used to evaluate predicting services;
RQ3. to identify the domain in which such frameworks are being applied;
RQ4. to identify the current knowledge about parameters that determine prediction's accuracy of these services.

As we shall see, these objectives are meaningful, important and increase confidence in forecasting services.

Researchers and practitioners would like to know the likelihood that predictions of forecasting services are right. In order to know that, there is a need of a framework that monitors and analyses the predictions. As a result, we can know when a service could have better results than another one, for instance.

We think that the main criterion to assess forecasting services is accuracy. One service is accurate when its predictions are right. However, there might be other criteria to assess these services. Therefore, we should thoroughly bear in mind previous research.

Furthermore, we have to identify the domains (such meteorology, stock market and betting shops) in which previous research has been carried out.

Finally, we need to identify under which parameters these services are more reliable. In the domain of weather forecast, a specific service could make predictions of the weather of one city more precise than predictions of another city one day in advance (locality). In contrast, the same service could make worse predictions from three days in advance (time).

2.3 Review methods

This section completes the protocol for the systematic review. To the background and research questions, which are identified in the previous section, are added the following components: the strategy that will be used to search for primary studies, the study selection criteria and procedures, the study quality assessment checklists and procedures, the data extraction strategy and the synthesis of the extracted data. The student presented the protocol to his supervisors for review and criticism.

2.3.1 Data sources

The following electronic databases are efficient to conduct systematic review in the context of software engineering:

- Google Scholar [7]. Google Scholar provides a simple way to broadly search for scholarly literature. From one place, you can search across many disciplines and sources: articles, theses, books, abstracts and court opinions, from academic publishers, professional societies, online repositories, universities and other web sites.
- ISI Web of Knowledge [8]. It contains a large collection of bibliographic databases, quotations and references to scientific journals in all disciplines of knowledge (scientific, technological, humanistic and sociological) since 1945. It has two performance evaluation tools: Journal Citation Report y Essential Science Indicators.
- Inspec (Engineering Village) [9]. Inspec includes bibliographic citations and indexed abstracts from publications in the fields of physics, electrical and electronic engineering, communications, computer science, control engineering, information technology, manufacturing and mechanical engineering, operations research, material science, oceanography, engineering mathematics, nuclear engineering, environmental science, geophysics, nanotechnology, biomedical technology and biophysics.
- ACM Digital Library [10]. It is a vast collection of citations and full text from ACM journal and newsletter articles and conference proceedings.
- IEEE Xplore [11]. It is a powerful resource for discovering and accessing scientific and technical content published by IEEE (Institute of Electrical and Electronics Engineers) and its publishing partners. It provides Web access to almost 3-million full-text documents.
- ScienceDirect [12]. ScienceDirect is a leading full-text scientific database offering journal articles and book chapters from more than 2,500 peer-reviewed journals and more than 11,000 books. There are currently more than 9.5 million articles/chapters.
- Springer [13]. SpringerLink is an integrated full-text database for journals, books, protocols, eReferences, and book series published by Springer. SpringerLink currently

offers more than 2,500 fully peer-reviewed journals and more than 45,000 books online.

- Scirus [14]. Scirus is the most comprehensive science-specific search engine on the Internet.

The DBLP [15] database was also used to find links to download some papers when they were broken in the above sources. The DBLP server provides bibliographic information on major computer science journals and proceedings.

2.3.2 Search strategy

The final search strategy was defined after three iterations. The following subsections report these iterations. Iterations were needed because we experienced troubles finding studies about our topic in the beginning.

During the selection of the search terms, the most suitable words, synonyms, acronyms or alternative spelling within the research field had been identified according to the three viewpoints (population, intervention and outcomes) recommended in [5]. Population denotes an application area. Interventions refer to software technologies that address specific issues. Outcomes are defined as factors of importance to practitioners such as improved reliability, reduced production costs, and accuracy of predictions.

A First iteration

A.1 Selection of the search terms

The goal was to find previous research about accuracy assessment in forecasting services. There were two points of view. On the one hand, the scope of service oriented architectures (SOA) and web services. On the other hand, everything related to assess the accuracy of predictions. Thus, we had two criteria for population:

Population = Pop. Criterion1 AND Pop. Criterion2

Pop. Criterion1: web services, service oriented, SOA.

Pop. Criterion2: forecast, prediction, predictive, predictability, decision support, weather, meteorology, stock exchange, stock market, betting.

The *Intervention* terms were the following: monitoring, monitor.

The *outcomes* should be the features of the predictions: reliability, accuracy.

A.2 Establishment of the search strategy

A two-phased strategy was selected as the way to perform the search.

First of all, in order to find a wider range of key words and some clue about where to start, the first iteration of the review protocol began reading the published research papers in 2010 of the conferences and journals showed in Table 1. These conferences and journals deal with services, service oriented architectures and web services.

The title and the abstract of those papers were analysed. Besides, the key words were looked up. The papers which were related with forecasting services or had key words were deeply read. After that, each paper was labelled with a colour (green was proper, yellow was interesting and red was out of scope) and some comments were made about them.

Table 1.- Conferences and Journals in the first iteration of the search process.

Acronym	Source
Conferences	
ICSOC	International Conference on Service Oriented Computing
ICWE	International Conference on Web Engineering
IEEE Services	IEEE World Congress on Services
Journals	
IEEE TSC	IEEE Transactions on Services Computing
Springer SOC&A	Service Oriented Computing and Applications
Springer JoSS	Journal of Service Science
ACM TWEB	ACM Transactions on the Web

The results of this manual search were:

- In the scope of prediction, there was no service oriented architecture which deals with comparing the prediction of a forecasting service with the real results. However, there are some SOA systems which perform predictions such as predictability in service execution for clusters hosting web services, prediction for achieving SLA (when workload varies) or for detecting SLA violations and prediction in domotic systems.
- Some papers refers to useful web pages to find web services, like webservicelist.com, and xmethods.net.
- We found a tool similar to SALMon, which is called VRESCo [16].
- In the references list, we found two journals where to carry on with the search: International Journal of Forecasting and Journal of Forecasting.

Secondly, we used the search string in the electronic databases of Table 2.

Table 2.- Electronic databases that were used in the SLR.

Electronic databases
Google Scholar
ISI Web of Knowledge
Inspec

The results of this systematic search in electronic databases were not promising, as we did not find any tool or framework based on SOA or web services comparing previously done predictions.

B Second iteration

B.1 Selection of the search terms

Since we did not find any tool or framework based on our scope, the goals were to find any software or technology which deal with accuracy assessment of predictions and to find several domains of predictions models and under which parameters these models are more reliable.

Population = Pop. Criterion1 AND Pop. Criterion2

Pop. Criterion1: web services, service oriented, SOA, QoS.

Pop. Criterion2: forecast, foresight, prediction, predictive, predictability, weather, meteorology, stock exchange, stock market, betting.

The *Intervention* terms are the following: monitoring, monitor, service, software, technology.

The *outcomes* should be the features of the predictions: accuracy

B.2 Establishment of the search strategy

Analysing some conferences and journals about services in the first iteration led to find interesting journals about forecasting, and one workshop about forecast verification (see Table 3).

A two-phased strategy was selected as the way to perform the search.

Firstly, a manual search of the journals and workshop of Table 3. The title and the abstract of those papers were analysed. Besides, the key words were looked up. The papers which were related with forecasting services or had key words have been deeply read. After that, each paper was labelled with a colour and some comments were made about them.

Table 3.- Journals and Workshops in the second iteration of the search process.

Sources
Journals
International Journal of Forecasting
Journal of Forecasting
Weather & Forecasting
Workshops
International Verification Methods Workshop

The results of this manual search were:

- Outside of the scope of SOA, Allan Murphy was a pioneer in accuracy assessment for forecasting models. He refers to that as "forecast verification" [17]. The first found paper about assessing the accuracy was [18].
- In the scope of web services, there has been some short work carried out by Ben Domenico [19], although there is no follow-up.
- About domains, these are the prediction markets with more research: sport forecasting, Bayesian forecasting in economics, forecast of macroeconomic variables, election forecasting, time series monitoring and weather forecast. Weather forecast is the domain which has the highest amount of research, probably because in other domains the work is not published.

Secondly, we used new search string in the electronic databases of Table 2.

The results of this systematic search in electronic databases were not promising, as authors were more concerned trying to make better predictions models than comparing previously done predictions.

C Third and last iteration

C.1 Selection of the search terms

In the two previous iterations of the search process, we had seen that there is not enough research about accuracy assessment for forecasting services based on SOA. We had been trying to find SOA systems or web services comparing predictions. However, we had found too little research in SOA Systems or web services which cope with forecast verification. On the one hand, we had found research in SOA systems which make predictions. On the other hand, we had noted that formal methods for forecast verification are a consolidated line of research. We had just found one work combining these two concepts (carried out by Ben Domenico [19]) although there has not been follow-up.

As a consequence, the goal in the third iteration was to find papers about general knowledge in SOA and web services, and specifically SOA systems which make predictions (e.g. prediction in QoS). As a result, we had two criteria for population:

Population = Pop. Criterion1 AND Pop. Criterion2

The population terms were the following:

Pop. Criterion1: web service, service oriented, SOA.

Pop. Criterion2: forecast, forecasting, foresight, foretell, foretelling, forethought, predict, prediction, predictive, predictability, predicting, prognosis, prognosticate, prognostication, prevision, anticipation, outlook.

Regarding the intervention point of view, we did not take into account the words monitoring, monitor and verification because they reduced too much the amount of found papers. We neither took into account accuracy for the outcomes point of view, because there could be other criteria to assess.

The logic formula to perform the search was:

1. ("web service" OR "service oriented" OR SOA) AND (forecast OR forecasting OR foresight OR foretell OR foretelling OR forethought OR predict OR prediction OR predictive OR predictability OR predicting OR prognosis OR prognosticate OR prognostication OR prevision OR anticipation OR outlook)

We can use truncation in some electronic databases, so the formula would be:

- ("web service" OR "service oriented" OR SOA) AND (fore* OR predict* OR prognos* OR prevision OR anticipation OR outlook)

This search did not return enough results, so we decide to add a second logic formula about service oriented architectures in general:

2. (SOA OR "service oriented") AND ("systematic review" OR survey OR "state of the art")

C.2 Establishment of the search strategy

A three-phased strategy was selected as the way to perform the search.

First of all, we reused the interesting papers found in the journals, conferences and workshops of the two previous iterations. The amount was 1 paper. Secondly, we queried our two formulas to perform the search into the databases of Table 2. We used only three electronic databases because adding more just gave more duplicated studies. We got 209

papers. Eventually, during the third phase, we went deeper into the references, and reused resources found in web pages. The expert opinion of my tutors was also considered. This phase returned 3 studies. The top of Figure 2 shows the establishment of the search strategy.

2.3.3 Study selection

Once we have the potentially relevant primary studies, they are going to be selected when they provide direct evidence about any research question.

We have an English title and abstract for all the obtained studies, so there are not exclusions based on the language of the primary study until the selection by abstract.

Knowledge of the authors, institutions, journals and year of publication has not been removed during the study selection process. It has been proved that masking the origin of primary studies does not improve reviews [5]. A complete list of selected and excluded studies has been maintained identifying the reason of inclusion/exclusion.

This process was made by a single researcher but included and excluded papers were discussed with his tutors.

As we split the review in two parts (SOA in general and SOA for prediction), there is a selection criteria for each part. The studies were excluded according to the following exclusion criteria (EC).

In the generic part of SOA and web services, exclusion criteria are:

- Selection by title (EC1): Papers too specific or not related to SOA are excluded.
- Selection by abstract (EC3): Papers which are not relevant to learn the current state-of-the-art of SOA.
- Selection by full text (EC5): Papers which do not aim to do a general review about SOA.

In the part of SOA systems which deals with prediction, studies needed to pass the following filters:

- Selection by title (EC2): Papers which do not cover both: SOA or web services and prediction.
- Selection by abstract (EC4): Papers which cannot answer any research question.
- Selection by full text (EC6): Papers which do not answer any research question.

Besides these exclusion criteria, we excluded another studies if they were: not accessible, not completely written in English (e.g. Chinese) or similar to another included papers of the same author.

To cover deeper two last research questions (to identify the domain in which such frameworks are being applied; to identify the current knowledge about parameters that determine prediction's accuracy of these services), we included a manual search. In this stage, it was considered as a positive point of the paper the possibility to access to its real data set and having a model or service for prediction.

During the three-phased search phase, we found 213 studies. The studies of the manual phases were directly until the selection by full paper. As result of the automatic search, 209 studies were identified. In the second step, 82 studies (39%) were identified as duplicated, so we had 127 non-duplicated studies. In the third step, 23 papers (18%) were excluded applying EC1 and EC3. In the fourth step, abstracts were revised and 86 studies (83%) were excluded. There were only 18 left papers. In the fifth step, 22 studies (1 from 1st search phase, 18 from 2nd search phase and 3 from 3rd search phase) were analysed by full paper. Eventually, 4 papers (18%) did not pass the last criterion and 18 papers were considered as the most relevant to our systematic review. Figure 2 summarises graphically the study selection process.

Table 4 summarises the efficiency of the electronic databases. In this table, the ratio index is the rate between the total of included studies of a database and the total of primary studies obtained. Some of the included studies were returned for more than one electronic database. That is why the column "included" exceeds the total amount of 18 studies. Google Scholar was the most efficient source, since 78% of all included studies were obtained in this source; whereas ISI Web of Knowledge contributed with 50% of the studies.

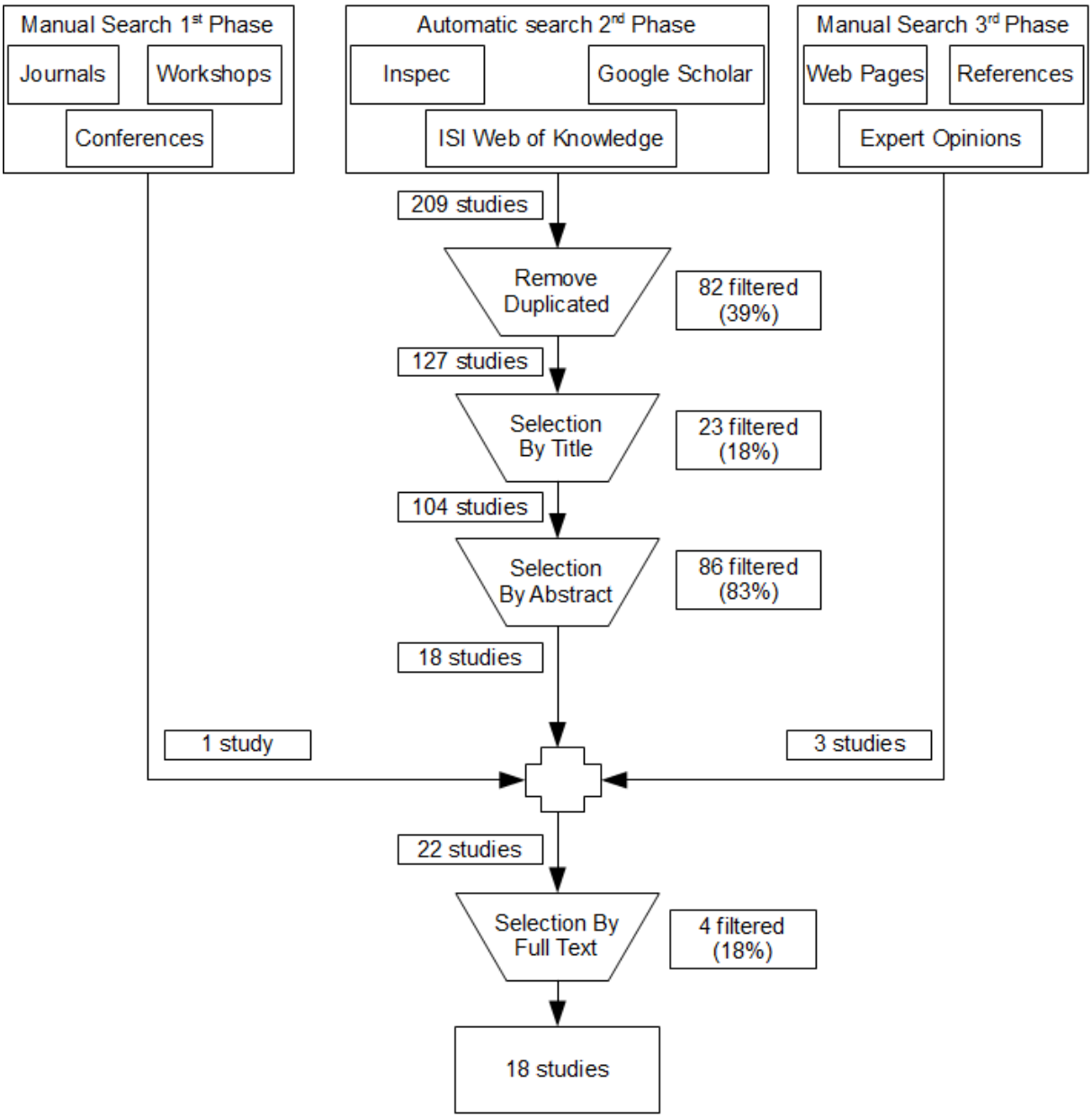

Figure 2.- The selection strategy for the primary studies.

Table 4.- Search sources, obtained and included primary studies.

Database	Returned	Included	Ratio Index	Date
ISI Web of Knowledge	53	9	0.50	12/04/2011
Inspec	67	11 (5 new and 6 duplicated)	0.61	14/04/2011
Google Scholar	89	14 (4 new and 10 duplicated)	0.78	15/04/2011

All 18 included studies are presented in Table 5. The column type is the paper's topic. It could be SOA, monitoring or prediction. The column document type indicates if the paper is a journal article (JA), a conference paper (CP), a technical report (TR), a master thesis (MT), or a book chapter (BC).

Table 5.- Included primary studies

Study	Authors	Year	Type	Doc. Type	Ref.
S1	L.B.R. de Oliveira et al.	2010	SOA	JA	[20]
S2	M.H. Valipour et al.	2009	SOA	CP	[21]
S3	M.P. Papazoglou et al.	2007	SOA	JA	[22]
S4	U. Zdun et al.	2006	SOA	JA	[23]
S5	G. Canfora et al.	2009	Monitoring	CP	[24]
S6	M. Oriol	2009	Monitoring	MT	[3]
S7	A.T. Endo et al.	2010	Monitoring	TR	[25]
S8	A. Bertolino	2007	Monitoring	CP	[26]
S9	N. Delgado et al.	2004	Monitoring	JA	[27]
S10	R. Guha	2008	Prediction	JA	[28]
S11	S. Punitha et al.	2008	Prediction	CP	[29]
S12	M. Marzolla et al.	2007	Prediction	CP	[30]
S13	H.N. Meng et al.	2007	Prediction	CP	[31]
S14	D.M. Hang et al.	2006	Prediction	CP	[32]
S15	C.B.C. Latha et al.	2010	Prediction	JA	[33]
S16	N. Xiao et al.	2008	Prediction	JA	[34]
S17	A.H. Murphy	1993	Prediction	JA	[18]
S18	B. Domenico	2007	Prediction	TR	[19]

Figure 3 shows the time distribution of the studies over the years. We can say that all of them are recent because they have been published in the last 7 years, with the exception of one paper about forecast verification that was published in 1993.

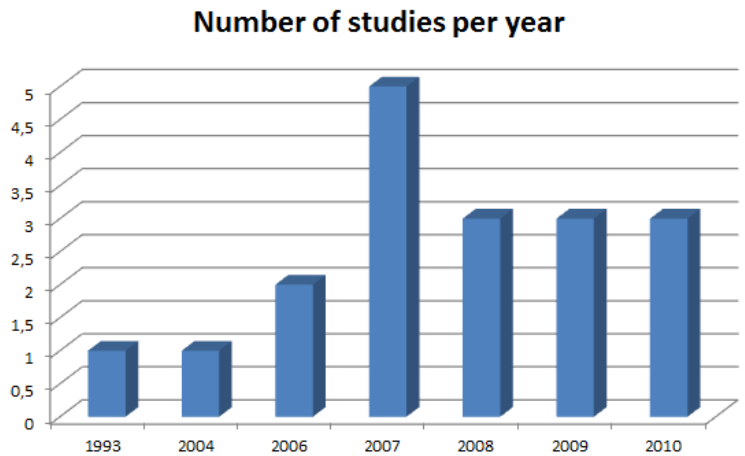

Figure 3.- Number of included studies per year.

2.3.4 Study quality assessment

Although there is no agreed definition of quality, it might be seen as the extent to which a study minimises bias and maximises internal and external validity. To assess the quality of primary studies, the following filters in Table 6 were used. Some of them were extracted from other systematic literature reviews [35] [36]. The rest were created by us.

Table 6.- Quality assessment form.

Quality assessment form	
QA1	Is there a clear and understandable statement of the aim(s) of the research?
QA2	Does the paper include an introduction or review about related work?
QA3	Does the study have a thorough and adequate research design to address the aim of research?
QA4	Are there clearly stated findings with credible results and justified conclusions?
QA5	Does the study provide value for research and further work?
QA6	Are there any discussions about limitations or validity of the approach?
QA7	Have been the method applied to some examples (e.g. tiny, real…)?
*	If the work is a general review about SOA or testing, do not continue with the quality assessment.
QA8	Have the predictions been assessed with an appropriate method?*

Table 7.- Quality assessment of the selected studies.

	QA1	QA2	QA3	QA4	QA5	QA6	QA7	QA8	Total
S1	✓	✓	✓	✓	✓	✓	×	-	6
S2	✓	×	✓	✓	✓	✓	×	-	5
S3	✓	✓	✓	✓	✓	✓	×	-	6
S4	✓	✓	✓	✓	✓	✓	×	-	6
S5	✓	✓	✓	✓	✓	✓	✓	-	7
S6	✓	✓	✓	✓	✓	✓	✓	-	7
S7	✓	✓	✓	✓	✓	✓	✓	-	7
S8	✓	✓	✓	✓	✓	✓	×	-	6
S9	✓	✓	✓	✓	✓	✓	✓	-	7
S10	✓	✓	✓	✓	✓	✓	✓	×	7
S11	✓	✓	✓	✓	✓	✓	✓	✓	8
S12	✓	✓	✓	✓	✓	✓	✓	×	7
S13	✓	×	✓	✓	✓	×	✓	✓	6
S14	✓	×	✓	✓	✓	✓	✓	✓	7
S15	✓	×	✓	×	✓	✓	✓	×	5
S16	✓	✓	✓	✓	✓	✓	✓	✓	8
S17	✓	×	✓	✓	✓	✓	✓	×	6
S18	✓	×	✓	✓	✓	✓	✓	✓	7
Total	18	12	18	17	18	17	13	5/9	

Each of these criteria has been graded on a dichotomous ("Yes" or "No") scale [36] whether the primary studies covered them or not. Table 7 shows the results of applying the quality criteria to each primary study. Despite there are studies which do not fulfil all of the quality criteria, we decided to include all the studies within this review. This decision was taken bearing in mind that this master thesis covers a recent research topic so there are not many studies that address it.

2.3.5 Data extraction

The aim of this section is to design data extraction forms to accurately record obtained information from the primary studies. Therefore, they must collect all the information needed to answer the review questions, the study quality criteria and the synthesis strategy. Our data extraction form, which is shown in Table 8, was created following the examples of other SLRs [35] [36].

Table 8.- Data extraction form.

ID	Field	Description	Research Question
Internal information			
1	Study identifier	Unique identifier for the primary study	
2	Date	Date of data extraction	
Reference information			
3	Title	Title of the study	
4	Author(s)	Author or authors of the study	
5	Year	Year of publication	
6	Type of study	Journal article, conference paper, workshop paper, book section	
7	Name of the journal /conf. /works. /book	Name of the journal, conference, workshop, book where the study was published	
8	Publication details	Rest of reference details	
Content Information			
9	Abstract	Original abstract	
10	Objectives	What are the objectives of the study?	
11	Framework	Are there monitoring systems? If so, do they assess the predictions?	RQ1
12*	Quality criteria	By means of which criteria do they assess the predictions?	RQ2
13*	Domain	What are the domains being studied?	RQ3
14*	Quality parameters	Are there parameters which affect the quality of the predictions?	RQ4
15	Conclusions	Original conclusions of the study	
16	Reviewer description	Reviewer description	
17	Additional notes	Space for additional notes	
Note	If the work is a general review about SOA or testing, do not reply questions with an (*).		

2.3.6 Data synthesis

The results of the included primary studies are collated and summarised in Chapter 3. Once the data was extracted, it was synthesised in a manner suitable for answering the review questions. Chapter 3 has been split into different topics which were synthesised in a descriptive way (non-quantitative).

Chapter 3 Systematic review results

The results of the systematic literature review are divided into the following four sections:

- 3.1: Service oriented architectures.
- 3.2: Testing and monitoring in SOA.
- 3.3: Prediction in SOA.
- 3.4: Criteria to evaluate predictions.

Sections 3.1 and 3.2 introduce the concepts of service-oriented architectures and monitoring respectively. We have written these two sections for two reasons: they are a good general introduction to the topic and there is not much work in our specific topic. As a result, we decided to take general papers about SOA and monitoring. At the end of section 3.2, SALMon is summarised.

The main contribution of this master thesis is in the sections 3.3 and 3.4. The former includes previous work about accuracy assessment for forecasting services, prediction models that are implemented with SOA principles, and performance prediction of web services. The latter contains all types of predictions' goodness in order to evaluate whether forecasting services make good or bad predictions.

3.1 Service oriented architectures

The contents of this section are organised as follows. Initially, an introduction defines what a SOA is, its benefits and how they can be implemented. Next, characteristics of SOA are showed. Afterwards, a detail review of recent reference models and architectures based on SOA is presented. Finally, future research lines of SOA are outlined.

3.1.1 Introduction

Services are autonomous, platform-independent, self-contained and well-defined modules which perform software functionalities. They can be described, published, discovered and loosely coupled in novel ways. Since they are network-available, any service can be reused [22]. They are the basis to compose more complex service-oriented systems.

A service-oriented architecture (SOA) is essentially a collection of services that are able to communicate with each other [23]. One can coordinate this collection of services to create business processes, combining them in several ways to accomplish some task [21]. OASIS (the Organization for the Advancement of Structured Information Standards [37]) defines SOA as: "A paradigm for organizing and utilizing distributed capabilities that may be under the control of different ownership domains. It provides a uniform means to offer, discover,

and interact with the use capabilities to produce desire effects consistent with measurable preconditions and expectations".

A SOA consists of the software's coarse-grained structure, where each service is the endpoint of a connection, which can be used to access the service and interconnect different services. Services are abstract modules of software deployed as a unit.

In another perspective, software architecture has received increasing attention in last years [20] because of increasing complexity of software. SOA is a descendant of the logical evolution of the software modularization techniques that go back more than 50 years (when structured programming was introduced).

In the two following sections, we will see the benefits of SOA and technologies to implement them.

A Benefits of SOA

Using services to support the development of applications has several benefits. It allows developing rapid, low-cost, interoperable, evolvable, scalable, manageable and massively distributed applications. It makes easier to manage growth of large-scale enterprise systems and to reduce costs by reusing services. Services could be provided to either end-user applications or other services distributed in a network. A well-constructed, standards-based SOA can empower an organization with a flexible infrastructure and processing environment provisioning independent, reusable applications functions as services and providing a robust foundation for leveraging these services.

SOA gives more flexibility in choosing the implementation of technologies and locations for service providers and consumers. The benefits of SOA resulted primarily from a single feature: the stability of the interface service. Both suppliers and consumers evolve independently as long as the interfaces remain stable. This stability isolates service to consumers in the development of implementation services. This isolation limits the scope of change and the cost of subsequent amendments.

B Technologies used to developed SOA systems

There are several technologies that can be used to implement service oriented architectures. Web services [38] and Jini [39] are the most common ones.

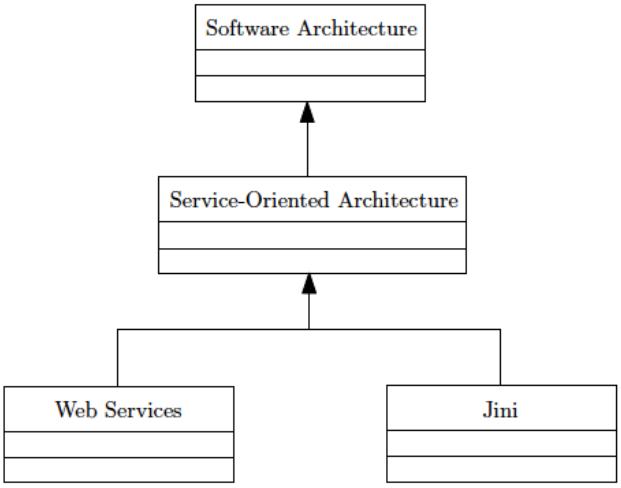

Figure 4.- Set of technologies for implementing services [21].

So we cannot say that SOA and Web services are the same. Web services are, instead, specialized SOA implementations that embody the core aspects of a service-oriented approach to architecture. Therefore, Web services just indicate a collection of technologies, such as SOAP and XML. Although web services provide support for many of the concepts of SOA, they do not implement all of them. For instance, they do not currently support the notion of contract lease neither its official specification provides QoS levels for a service.

In any event, web services are currently the most promising technology for service-oriented computing (SOC) [22]. They use the Internet as the communication medium and open Internet-based standards such as SOAP (Simple Object Access Protocol) for transmitting data, WSDL (Web Services Description Language) for defining services and BPEL4WS (Business Process Execution Language for Web Services) for orchestrating services.

3.1.2 Characteristics

Each system's software architecture follows a set of principles, depending on the designers' decisions. Service-oriented software architecture has these main characteristics [21]:

A Discoverable and dynamically bound

Based on a set of criteria that fulfil the customer's needs, a service consumer discovers what service to reuse. This can even happen at runtime. This could be done since SOA supports the concept of service discovery.

Services have an interface description. The interface description contains all interface details about the service. In other words, it describes how operations could be accessed remotely and explains its functionality. With this interface description, services can be offered. The service advertises itself at the lookup service.

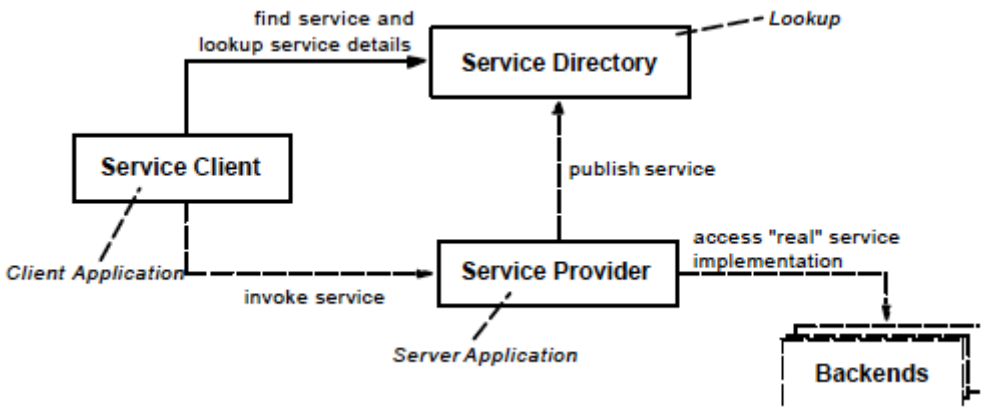

Figure 5.- Lookup of services in a SOA [23]

Figure 5 illustrates how service discovery works. Firstly, a service provider publishes a service in a service directory. The service provider often uses backends such as ERP systems, legacy systems [40], or databases. Although it is optional, flexible integration of heterogeneous backend systems is a central goal of a SOA. Secondly, clients look services up in order to find the software functionality that they need. To avoid problems of service identification, local service identifiers are extended including location information (like host name and port).

B Self-contained and modular

Modularity is one of the most important concepts in SOA. The following criteria apply equally well when determining whether a service is sufficiently modular.

- Modular decomposability. It is sometimes called “top-down” design. It consists of big problems which are iteratively decomposed into smaller ones. Each module is responsible for a single distinct function. The aim for service design is to identify the smallest unit of software that can be reused in different contexts.
- Modular composability. This is sometimes called as “bottom-up” design. It refers to the production of services that may be freely combined as a whole with other services to produce new systems.
- Modular understandability. It is the ability of a person to understand the function of a service without having any knowledge of other services.
- Modular continuity. Every service must hide information about its internal design. Modular continuity is achieved when changes in a service does not require a change in other services or in the consumers of the service. A service that exposes its internal design, limits its modular continuity. When changes are applied, it could lead to a domino effect.
- Modular protection. The modular protection of a service is sufficient when an irregular condition in the service does not cascade to other services or consumers.

C Interoperability of services

Interoperability is the ability of systems using different platforms and languages to communicate with each other. Services reflect a contract between the service provider and service clients. Service contracts are not just an interface description, but they also define the interaction between service client and service provider. Service contracts includes the following information about a service: communication protocols, message types, operations, operation parameters, exceptions, message formats, encodings, payload protocols, pre- and post-conditions, side-effects, operational behaviour, legal obligations, service-level agreements, directory service. In this contract, we can found an interface that can be invoked through a connector type. Interoperability is achieved by supporting the protocol and data format of the service and clients, which should be define in the service contract.

D Loose coupling

Coupling refers to the number of dependencies between modules. There are two different types of coupling: loosely coupled modules have a few well know dependencies whereas tightly coupled modules have many unknown dependencies. Obviously, SOA promote loose coupling between service consumers and service providers.

E Location transparency

Since the lookup and dynamic binding to a service can happen at runtime, the service implementation can change its location without the client's awareness. With the help of a load balancer which forwards requests to multiple service instances, a greater availability and performance can be achieved.

F Composability

There are three ways to compose a service: application composition, service federations and service orchestration.

- An application is an assembly of components, services and application logic that is made for a specific purpose.
- A service federation is a collection of services managed together in a larger service domain.
- Service orchestrations are executions of a single transaction that impacts one or many services in an organization. They are sometimes called business processes. A business process consists of a number of service invocations. If any of the service invocations fails, the entire transaction should be rolled back to the previous state. Thus, they do not perform data commits themselves.

G Self-healing

A self-healing system has the ability to recover from errors without human intervention. This characteristic is important because of the complexity and size of modern distributed applications.

3.1.3 Reference models and reference architectures

Software architecture is a set of plans which guide the selection of architecture elements, their interactions, and the constraints of these interactions. It can also be seen as a box-and-line schema of the system that is created to solve the problems that are defined in the specifications [21]. In this schema, a box is an element or part of the system and a line defines the interaction between the elements.

Software architecture works as a bridge between requirements and implementation. The architecture of a system describes its gross structure. The importance of software architecture lies in its benefits during the software development. It makes easier to understand large problems by presenting them at a high-level design, reuses other components, and is a first blueprint of the system's components.

Ruas de Oliveira et al. present a detail review of recent reference models and architectures based on SOA [20]:

- They identified that reference architectures and reference models deal with some of the following SOA characteristics: service publication, quality of service, policies and governance, service composition and enterprise service bus.
- These models and architectures provide a common basis that facilitates the development of systems. Their main benefits are: inter-operability, better comprehension of the domain, establishment of a common vocabulary, architectural reuse, consistence in the system's representation, and a better time-to-market.
- Some models and architectures are domain-dependent whereas others attempt to be generic. In the one hand, the most investigated applications domain are governmental systems, e-learning and collaborative work. In spite of these domains, more effort is needed on other domains. On the other hand, there are two initiatives widely known which are used as basis of other reference architecture and models. They are S3 reference architecture [41] and OASIS reference model [42].
- In most of the cases, this architectures and models present an instantiation or implementation based on them.
- In order to define these models and architectures, creators used existing systems, knowledge of the domain expert or other architecture which are not based on SOA.

3.1.4 Research road map

Papazoglou et al. [22] present a research road map for service-oriented computing (SOC). As Figure 6 shows, they split future research in three planes: service foundations at the bottom, service composition in the middle and service management and monitoring on top. There is another area for research which cut across previous three planes including semantics, non-functional service properties and quality of service. These areas are described next.

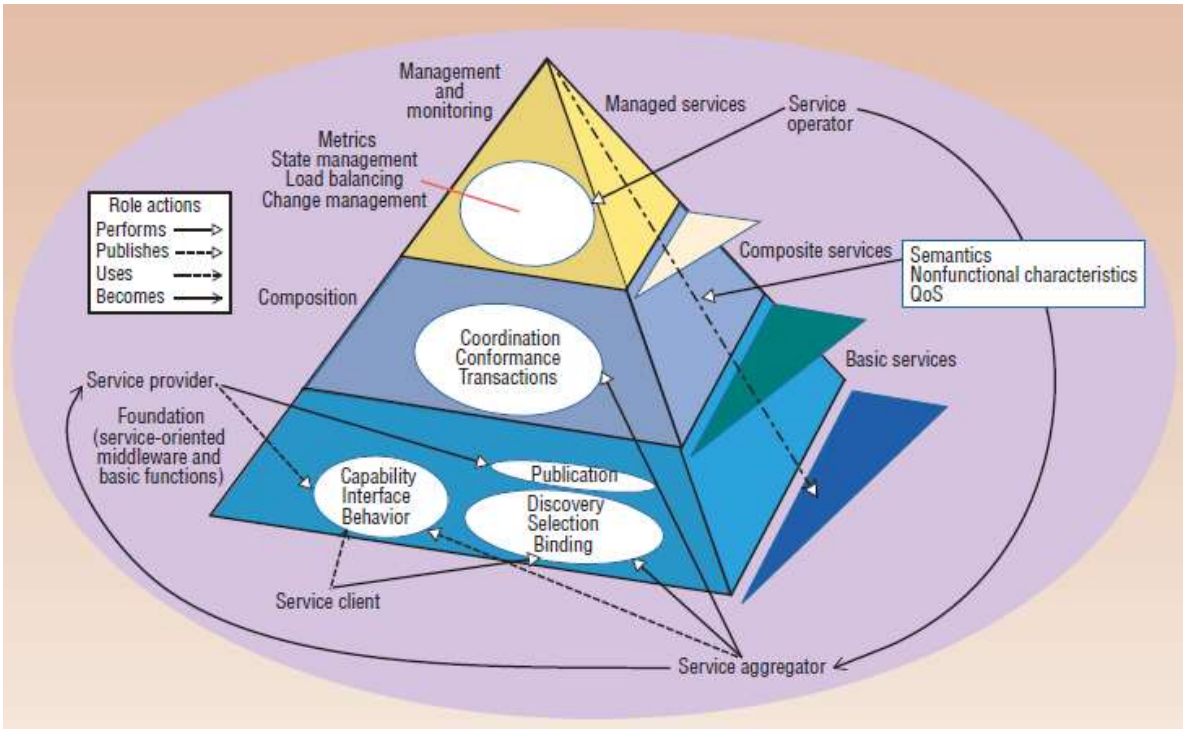

Figure 6.- SOC research road map [22].

A Service foundations

Service foundations consist of a service-oriented middleware backbone that realizes the runtime SOA infrastructure. Some of the most notable research challenges in near future include:

- Dynamically reconfigurable runtime architectures. In terms of efficiency and availability, the best service should be chosen at runtime to create an optimal architectural configuration. The goal of this auto configuration at runtime is to accomplish user's requirements.
- End-to-end security solutions. It requires a full system approach to test end-to-end security solutions in the application level and the network.
- Infrastructure support for data and process integration. Even when the architecture is changing at runtime, all data should be accessed by any requester or its characteristics (data format, source or location).

- Semantically enhanced service discovery. It consists of using automated ways to discover right services with minimal user involvement.

B Service composition

It refers to aggregate multiple services into a single composite service. This composite service could be used as a basic one. Thus, service aggregators are also service providers. Major research challenges in the near future are:

- Composability analysis for replaceable, compatibility and process conformance. Service conformance imposes semantic constraints on the components services and data to guarantee that services exchanges are satisfied.
- Dynamic and adaptive process. The challenge is to create techniques that support dynamic service compositions.
- QoS-aware service compositions. This means that service compositions should be auto managed by means of QoS. In other words, coordinate and control collaborative services when QoS do not fulfil policies, performance levels, service-level agreements, security requirements and so on.
- Business-driven automated compositions. There should be an abstraction from the logic at the application level. This abstraction may enable the composition of distributed business processes and transactions.

C Service management and monitoring

In order to manage the architecture, the status of the system should be assessed. Service management is responsible of several activities, such as installation and configuration, collection of metrics, tuning and to ensure responsive service execution. To perform all these activities, there is a need for information. Information is collected by means of service monitoring.

Major research challenges in the near future include an evolutionary approach in which autonomic capabilities anticipate runtime system requirements and resolve problems without human intervention. These challenges are self-configuring (service's auto configuration to be adapted to the current environment), self-adapting (service's adaptation to changes in the environment), self-healing (service's reaction to disruptions), self-optimizing (auto tuning to meet end-user needs) and self–protecting (protection against threats) management services.

D Service design and development

SOAs must rely on an evolutionary software engineering approach to provision reusable, independent automated services and provide a robust foundation for leveraging these services. The most prominent research challenges are:

- A new service-oriented engineering methodology is required for service applications.
- Flexible gap-analysis techniques. Current abstract service/process interfaces are not enough. More implementation details should be included in them.
- Service versioning and adaptability. Techniques to analyse business processes instantaneously are needed.
- Service governance. The potential composition into business processes through organisational boundaries can work properly only if services are effectively governed for compliance with QoS.

3.2 Testing and monitoring in SOA

In recent years, monitoring and testing has attracted increasing interest from researches for the following reasons:

- The cost and inadequacy of testing. Testing can consume fifty per cent, or even more, of the development costs [26].
- The increasing complexity and ubiquitous nature of software systems [27].
- It is the way to provide users and system integrators the means to build confidence that a service in service-oriented architectures (SOAs), which is used and not owned and runs on a machine out of the user's control, delivers a function with the expected quality of service (QoS) [24].

Testing consists of observing the execution of a software system to validate whether it behaves as intended and identify potential malfunctions. Testing is widely used in industry for quality assurance: indeed, by directly scrutinizing the software in execution, it provides a realistic feedback of its behaviour and as such it remains the inescapable complement to other analysis techniques [26].

Bertolino sees software testing as a broad term encompassing a wide spectrum of different activities, from the testing of a small piece of code by the developer (unit testing), to the customer validation of a large information system (acceptance testing), to the monitoring at run-time of a network-centric service-oriented application. Therefore, monitoring can be considered as one of the activities of testing.

Monitoring tools provide evidence that program behaviour complies or does not comply with specified properties during program execution. The main difference between testing and monitoring is their aim. Testing aim to ensure universal correctness of programs, whereas runtime software-fault monitoring is to determine whether the current execution preserves specified properties. Thus, monitoring can be used to provide additional defence against catastrophic failure and to support testing by exposing state information.

Service monitoring can play an important role in cutting testing costs. With the help of monitoring we can succeed in useful tasks such as:

- Preventing failures – for instance, by replacing an unavailable service with another equivalent.
- Verifying that a service invocation meets given pre- and post-conditions.
- Triggering recovery actions when needed.
- Offering access to monitored data for testing tools in a corporate environment.

However, testing a service-centric system requires the invocation of actual services on the provider's machine. If there are a huge number of invocations, we have a massive testing, which could cause a denial-of-service phenomenon for service providers.

In the following subsections we will see the basic structure of a monitor, the testability of services, and a framework to monitor services and test the accomplishment of SLAs which is called SALMon.

3.2.1 Structure of a monitor

A monitor is a tool that observes the behaviour of a system and determines if it is consistent with a given specification [43]. Before seeing a typical structure of a monitor, let us see some definitions to understand how they work:

- Software properties are relations within and among states of a computation. They answer the question: what relations about states of a computation lead to acceptable external behaviour? A monitor uses properties to discover faults prior to them becoming failures.
- The state $\sum p$ of an executing program is the state of the store and the individual threads executing in that program.

A monitor typically attempts to verify a software property while the software system is executing. A software property often has the form $\mu \rightarrow \alpha$, where μ is some condition on $\sum p^i$ that identifies the states in which α must hold. This means that in any state where μ is true in $\sum p^i$, then α must also be true in $\sum p^i$. If α evaluates to false, then the current execution has reached an unwanted state.

A monitor is composed of two parts: an observer that evaluates μ and an analyser that evaluates α. The Figure 7 shows this structure.

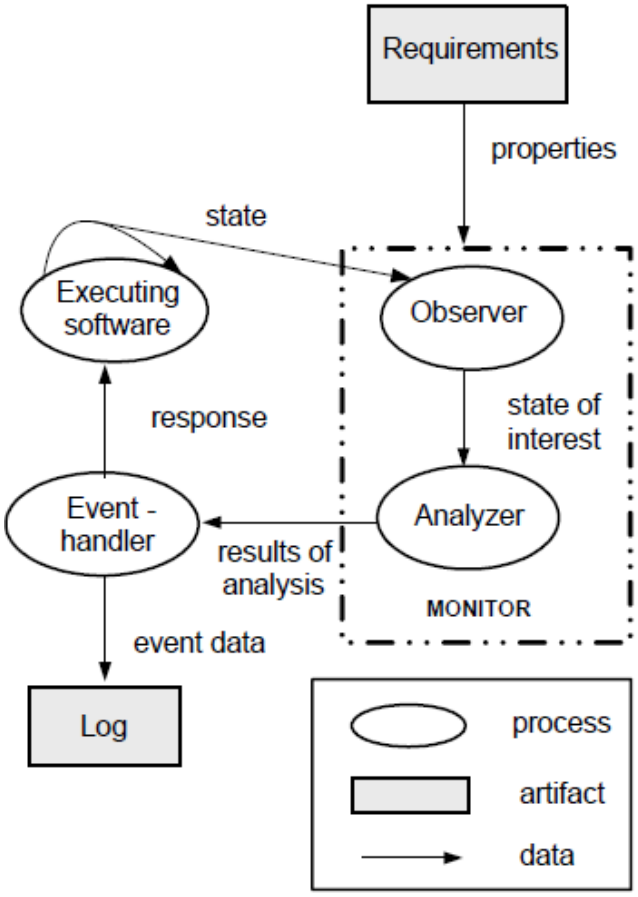

Figure 7.- High-level view of a runtime monitor [27].

In order to understand this structure, let us see an example from [27]. Suppose a program must never allow two processes access to a critical region simultaneously. This property may be specified for a monitor as: when a process enters the critical region, the number of processes in the critical region must be exactly one. The observer checks the program state to determine if a process has entered the critical region. When it detects this event, the analyser tests the proposition that the number of processes in the critical region is exactly one.

When a violation of a property is detected by the analyser, the monitoring system must respond in some fashion. The response could require the system to initiate an action such as halting the program, entering a recovery routine, or sending event data to a log. The event-handler is the mechanism that captures and communicates the monitoring results to the system or user and possibly responds to a violation.

3.2.2 Testing services

Testing within the emerging development paradigm is a research challenge to achieve 100% automatic testing [26]. Nowadays, the paradigm of development that promises to release higher quality and less costly software is service-oriented computing [24].

Service-oriented architecture (SOA) testing has similarities to commercial off-the shelf (COTS) testing: the provider can test a component only independently of the applications in which it will be used, and the system integrator is not able to access the source code to analyse and retest it. However, there is a clear difference: COTS are integrated into the user's system deployment infrastructure whereas services live in a foreign infrastructure.

There are some key issues which limit the testability of SOA Systems. Therefore, it is needed to develop new strategies or to adapt existing testing approaches for other paradigms (like traditional monolithic systems, distributed systems, component-based systems and Web applications). These key issues are: white-box testing is not possible since service's code and structure is unavailable, run-time discovery and ultra-late binding do not allowed to determine the components invoked in a given call-site, user loses control because the service may evolve without prior notice, a provider could lie providing incorrect description of a service's behaviour, and testing calls to a service are also charged to the user (per-use basis) [24].

SOA Testing is discussed across two dimensions in [24]:

- Testing perspectives. Various stakeholders, such as service providers and end users, have different needs and raise different testing requirements. There are five perspectives:
    - Service developers aim to release a highly reliable service and look for the maximum possible number of failures.
    - Service providers need to ensure that a service can guarantee the requirements stipulated in the SLAs with customers.
    - Service integrators test a service in order to gain confidence that this service fits all assumptions made at design time and can be bound in any composition of services.
    - Third-party certifiers assess service's fault proneness and recommend or not a service to the end-user.
    - End-users are just interested in the proper work of their applications.
- Testing levels. There are four levels:
    - unit testing of atomic service or service compositions,
    - integration and interoperability testing (which is crucial because of the dynamic binding nature in SOA),

- regression testing (which consists of retesting a piece of software after changes to ensure that these changes do not adversely affect the delivered service),
- and testing of non-functional properties (this is done through checking the SLAs between providers and users by means of QoS).

There are formal approaches to test web services. They can be categorised depending on their application context and the model used [25]. Figure 8 shows this categorisation.

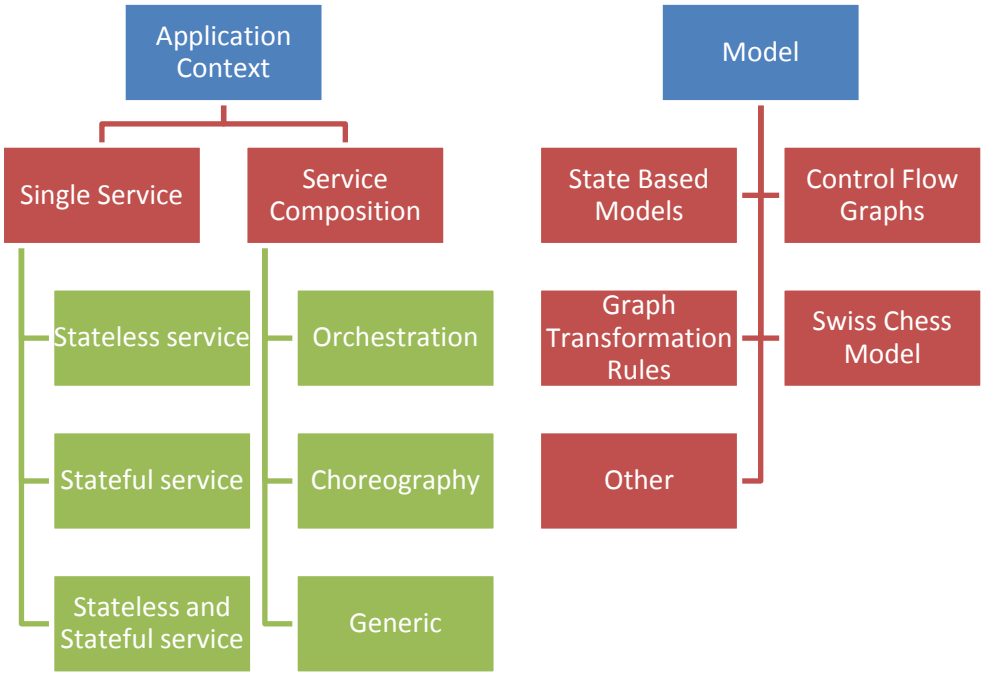

Figure 8.- Categorisation of formal testing approaches for web services.

There are two application contexts when coping with testing web services: single service testing and service composition testing. The former consists of assuring the reliability of an isolated service. A service can keep its current state varying the output of an operation according to the state. If a service keeps state information of the current state, it is a stateful service. Otherwise, it is a stateless service, because it is not necessary a context (previous operation invocations) to invoke a service operation. The latter focuses on the integration and execution of a set of services contained in a composition. There is research for testing two paradigms of composition: orchestration and choreography. The main models used to support the formal testing of web services are: state based models, control flow graph and extensions, graph transformation rules, swiss chess model and others [25].

3.2.3 SALMon, a tool for monitoring web services

SALMon [3] [4] is a SOA System that has two services: the monitor service and the analyser service. The monitor service measures the values of dynamic quality attributes (such as response time and availability) in order to retrieve QoS of a service-oriented system (SOS). The analyser service detects whether a SLA is being or going to be violated. The analyser needs a set of dynamic quality attributes for services to compare them with obtained QoS. Figure 9 shows the architecture of SALMon.

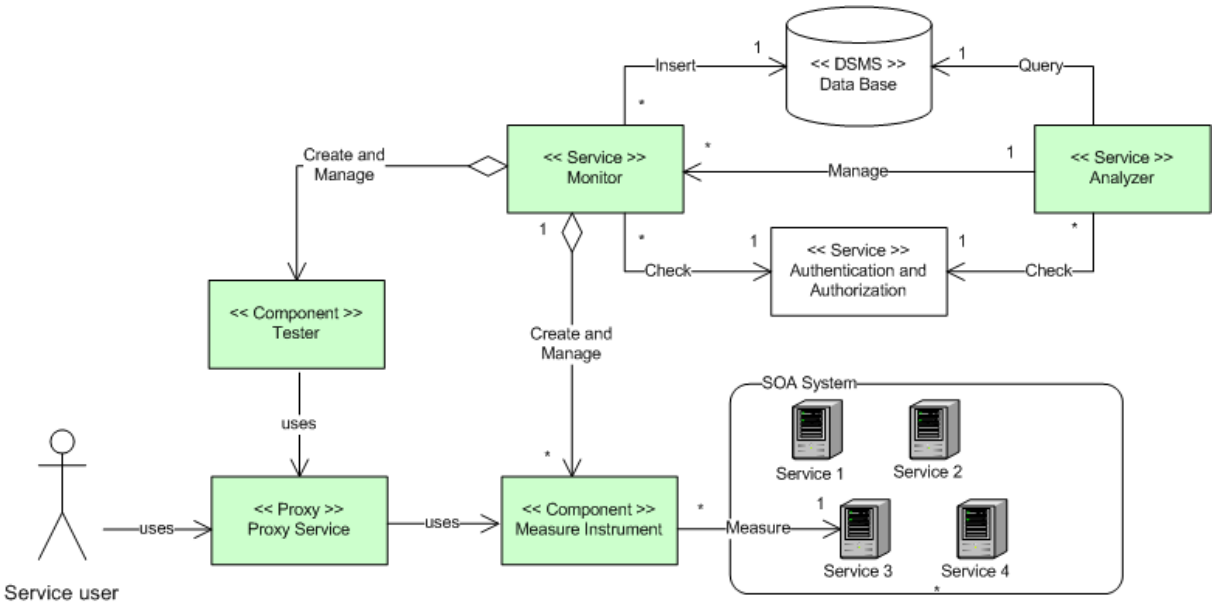

Figure 9.- SALMon Architecture [4].

In this master thesis, we are only interested in the monitoring service. We will use SALMon to monitor web services because it supports passive monitoring and testing, it supports any kind of service technology, it can be integrated in our architecture and new ways to measure QoS can be added. There are more contributions in the literature to obtain real-time QoS in SOS, but no one of them has all previous advantages [4].

The monitor service has two approaches to retrieve QoS of a service-oriented system: passive monitoring and active monitoring. Passive monitoring collects data from the interaction between the provider and the client whereas active monitoring (also known as testing) invokes the service in a systematic manner. On the one hand, active monitoring is adequate to detect a failure of the service before the client but it overloads the system and control structure and frequency of invocations. Moreover, there are certain operations that cannot be tested because they cannot be rollback. On the other hand, passive monitoring does not overload the system but loses the control of the invocations.

Measure instruments (see Figure 9) are used to get the value of a specific basic quality metric of either a service or an operation. Examples of quality metrics are current response

time, current availability and current round trip time. Measure instruments are technologically dependent on monitored service. This way, the monitor service is generic and use measure instruments when they are needed. Measure instruments provide high extensibility to the monitor.

In passive monitoring, the service user calls service through a proxy. This is a man-in-the-middle approach, since the proxy is between the user and the service. In active monitoring or testing, a tester component needed to be created. The tester invokes the services in a systematic manner. As measure instruments, the tester is technologically dependant.

3.3 Prediction in SOA

This section cover three topics: forecast verification of services, prediction models based on SOA and performance predictions of services.

3.3.1 Assessing forecasting web services

In the literature, the closest work to our work has been done by Ben Domenico [19]. He had the same idea as us: “compare a forecast for a given phenomenon or property (surface temperature, pressure or rain for example) with the actual observed values at specific places and times”. Additionally, he used geosciences web services in order to provide a framework and tools (important cyber infrastructure components) that enable experts in the field to compose their own system. In other words, he used geosciences web services to integrate all sources of data, to easy the replacement of different components in the system and standardise their interface specifications. With this base, one does not have to create all the processing steps and gather all the data locally to get started.

On the other hand, our approach has the same initial idea, but its final goal is to rank forecasting services according to their accuracy instead of integrating forecasting services to take data from all of them.

By means of GIS (geographic information system) technologies, Domenico differentiates three data types depending on their nature:

- Point: it refers to observations from sensors at meteorological observing stations. The measurements are taken continually at fixed points in space.
- Grid: it is related to the output of forecast models, which are estimated by numerical simulation. A grid refers to a zone/area in the space, so its feature type is coverage.
- Digital elevation models (DEM): to determine the location of an observing station in the Earth’s surface.

The architecture that allows comparing forecast with actual observations can be seen in Figure 10, where:

- Web Feature Services (WFS) provide access to point feature data collections.
- Web Coverage Services (WCS) cope with forecast model output data sets.
- Sensor Observation Service (SOS) manages the weather station observations.
- Sensor Web Enablement (SWE) integrates technology of other services, like WFS and WCS.
- NAM is the NCEP North American Model. It assumes that the earth is spherical.
- DEM is the before mentioned digital elevation model.
- WPS performs the transformation from NAM to a spheroid.

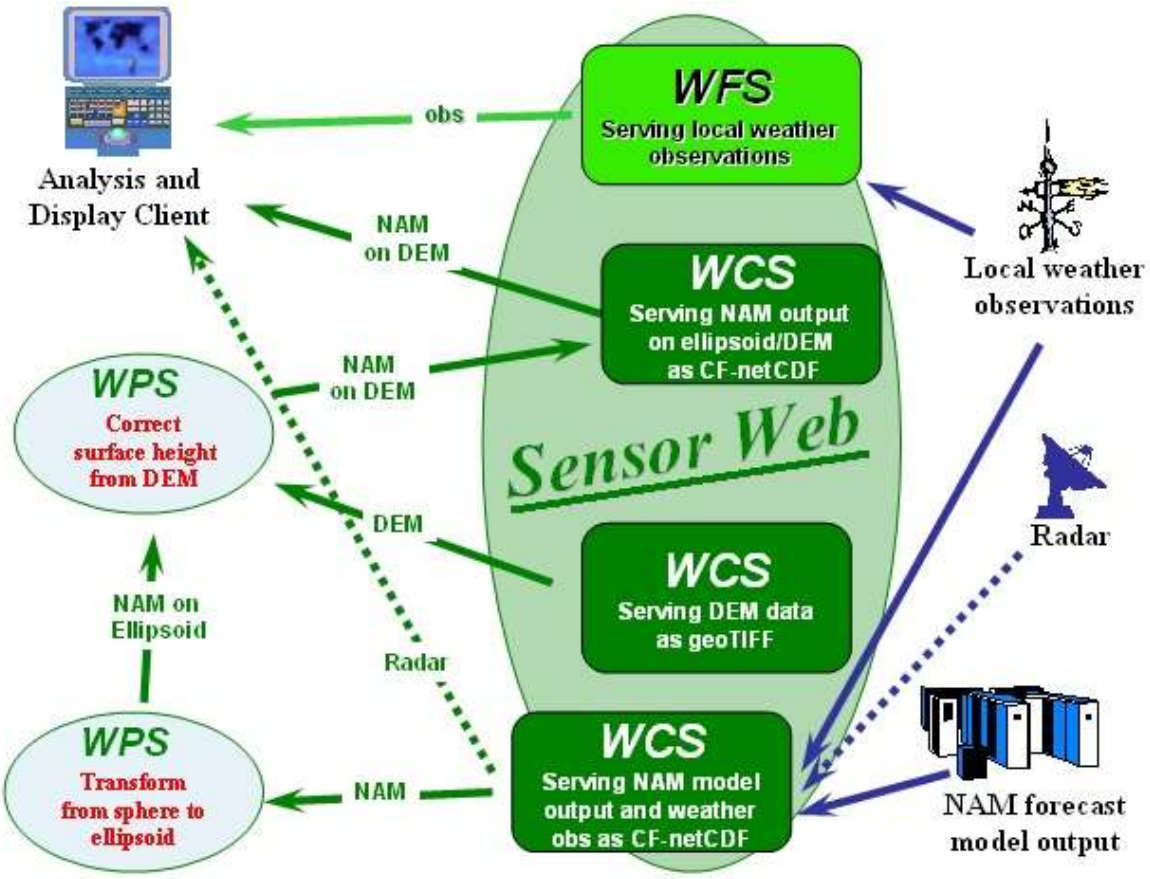

Figure 10.- Real-time automated forecast model verification system [19].

As we can see in Figure 10, the blue arrows on the right side indicate that the real-time is delivered automatically to a set of standards-based data servers: WFS and WCS servers. WPS performs the transformation from NAM models to a spheroid, which is a more realistic approximation to the mean sea level surface of the Earth. The location of the observations is redefined by DEM. Finally, the data source radar is included in the diagram to show that other different data sources can be introduced in the system.

In Figure 11, we can see a very general view of how SWE technologies can be used as a mechanism for integrating services.

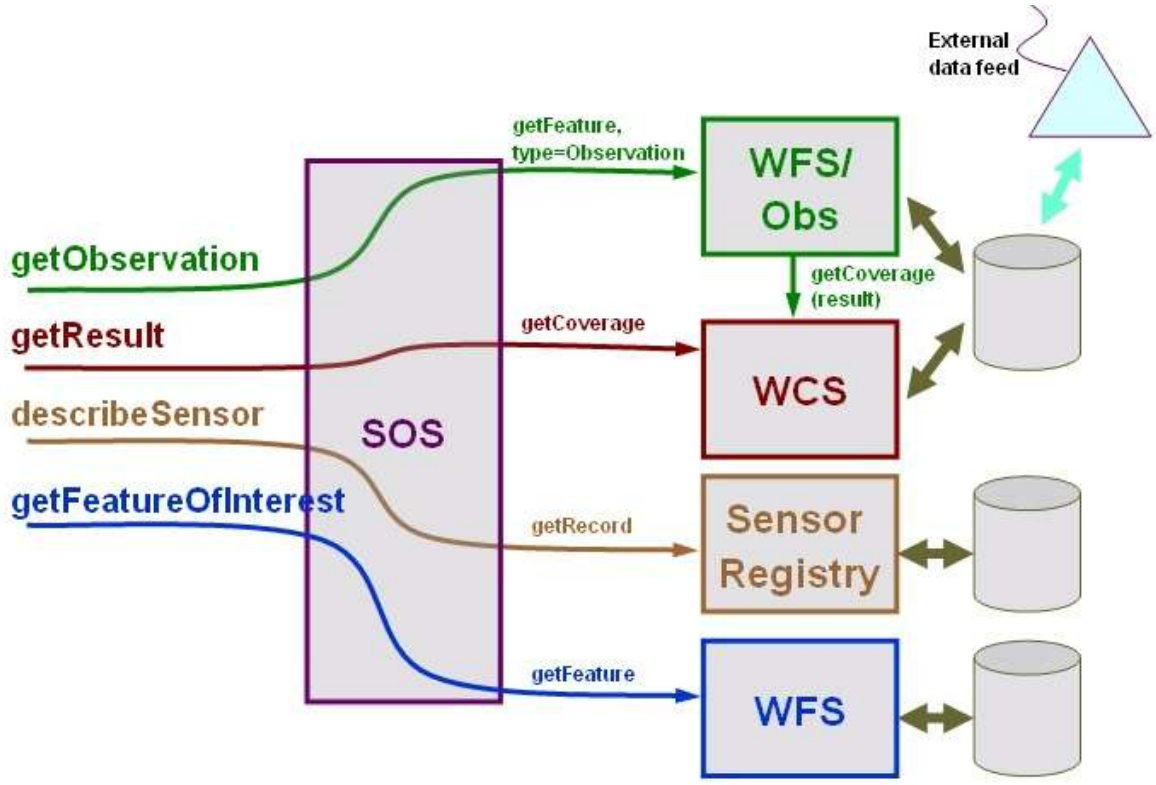

Figure 11.- SWE Sensor Observation Service as a composition tool [19].

Although this work is published on the Internet, there is neither published scientific paper nor follow-up.

3.3.2 Prediction models based on SOA

Nowadays, the increasing demand for computing power and the need of integrating and reusing different resources in prediction applications, have led to use service-oriented architectures. When prediction models are independently implemented and based on conventional architectures, they present obstructions for model reuse, data integration and system replanting. For instance, prediction applications need to be up-to-date in order to find new explanations and rules. With traditional technologies, they can only be accessed within local systems or LAN and do not provide interfaces for external applications. Moreover, large amounts of data are needed to be processed. These data usually lie in various regional governments or companies that might employ different operational systems and kind of databases. Thus, it is difficult to recollect and integrate the data.

However, SOA architectures provide the following advantages, which are not present in stand-alone prediction model applications:

- Reusable models. Complex services, which have the functionality of prediction models and data processing, can be assembled by low-level services that can be reused by external resources.
- Easier maintenance. There is a loose coupling relationship between supplier and user that assure an easier maintenance of the system.
- General solution for remote access. Heterogeneity of platforms and languages is not a problem in SOA. There is no restriction in the manner in which services can be used. It presents several strong points:
    - the user does not need to install or manage any specific software,
    - integration of distributed resources is possible,
    - pervasive computing is supported since services are available anywhere, at any time and on any device.

Therefore, service-oriented infrastructures provide an interesting and potentially general solution to the deployment of predictive statistical models.

We have found in the literature three prediction models which are based on SOA. These are in the domain of weather forecasting, macroeconomic analysis and drug design, respectively.

In [33] the authors develop a weather information system based on SOA. This information system forecasts weather conditions by means of data mining techniques. In their work, they explain the basics of web services and then propose their approach to forecast weather conditions.

They defend that there are two methods to forecast weather: the empirical approach and the dynamical approach. The former is based on the occurrence of analogues and is also called as analogue forecasting. It is useful to predict local scale weather when recorded cases are abundant. The latter, computer modelling, is based upon equations and forward simulations of the atmosphere. They use a combination of both techniques.

Basically, their infrastructure consists of one web service, which is developed in .NET, to get weather data from the Internet. They had been retrieving data through the web service for a year. This data is the input of their models. Several weather parameters were recorded at 1-hour interval, but only daily maximum temperature is used in their work. In our proof-of-concept we work with both maximum and minimum daily temperature. On the other hand, they use the WEKA toolkit to obtain the weather forecasts. During the process of data mining, they use Support Vector Machine (SVM) algorithm and Support Vector Regression (SVR). The maximum temperature is predicted based on the maximum temperature of previous n days, where n is the optimal length of the span (n is obtained by experimental iterations).

Han et al. propose a SOA-based Macroeconomic Analysis and Forecasting System, which is called SMAFS, that simulates and forecasts macroeconomic cycle trend by analysing the macroeconomic data [32]. SMAFS has three tasks, which are done by several subsystems: analysing/simulating real economic activities trend (macroeconomic cycle index sub-system), preventing the departure from normal path during crisis (early-warning sub-system) and forecasting the results that would influence main macroeconomic index in the future (economy forecast sub-system).

Besides, there are two more subsystems: heterogeneous data integration platform and user interface. The former draws data from various distinct databases and transforms it into the required format by models' calculation. In this sense, it has the same functionality as the forecast data collector service of our architecture (see Figure 19). The latter one queries data on user's request and feedback the result to users with the form of either table or charts. In our architecture, this is done by the forecast verifier service.

Its implementation provides clients with varied modules and methods of macroeconomic analysis and forecasting in the form of web services. Web services (econometric models) are the core of the system and use data which come from the heterogeneous data integration platform. In this architecture (shown in Figure 12), web services do not implement directly the functionality. However, their tasks are to authenticate and authorize incoming service request and then rely on backend models components and its workflows.

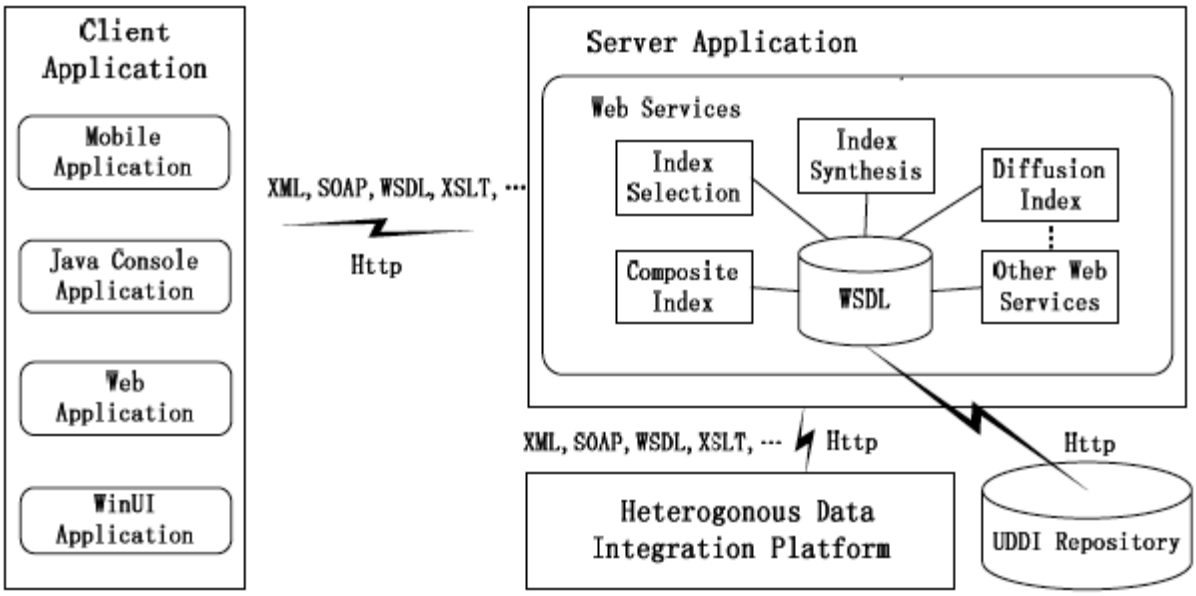

Figure 12.- The network architecure of SMAFS [32].

SMAFS needs huge numbers of economics data series. In order to effectively collect and use all data and to integrate all heterogeneous databases, a uniform data platform was built. It has four layers. The first layer consists of an accessible interface for the external data

consumers. Secondly, the data integration layer records data services and manages users' access to the data sources. The third layer refers to the service layer and is the connection between the second layer and the database. Finally, the fourth layer consists of various databases with original structures. It can be seen in Figure 13.

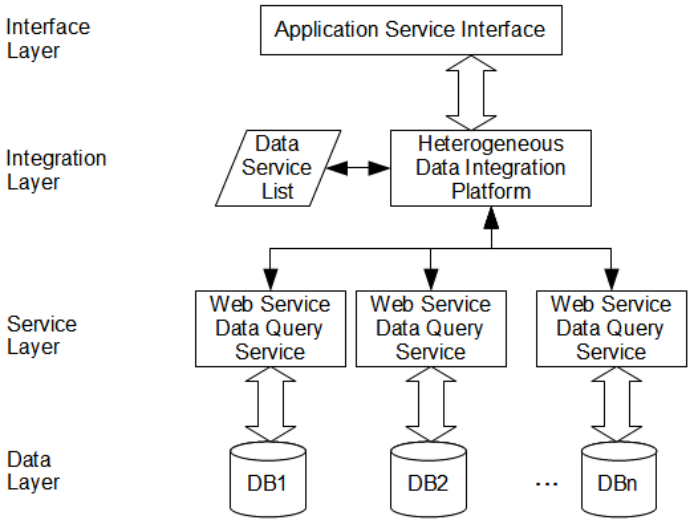

Figure 13.- The heterogonous data integration platform in SMAFS [32].

In [28] Guha presents a flexible and generalizable approach to the deployment of predictive models in the field of drug design, based on a web service infrastructure for the deployment of models developed in R [44]. An overview of the R Web service infrastructure is shown in Figure 14. R is a programming language and software environment for statistical computing and graphics. The R language has become a de facto standard among statisticians for developing statistical software, and is widely used for statistical software development and data analysis [45].

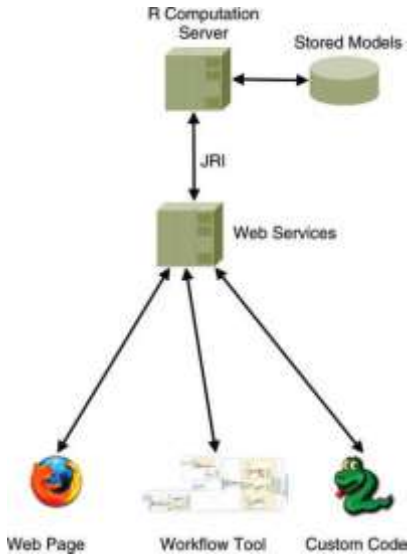

Figure 14.- An overview of the R Web service infrastructure [28].

JRI [46] is a Java library that allows web services to communicate with a remote R server. Models are stored in the file system instead of into databases. In his work, Guha allows to perform the following functionalities over the models: linear regression, neural network regression, random forest, LDA, k-means feature selection, model generation, sampling distributions and plots. As we can see, web services do not constraint users in any fashion, and the models can accessed from a web page, a workflow tool, or custom code.

3.3.3 Performance prediction of services

The final goal of this master thesis is to recommend a forecasting service from a set of forecasting services because it presents the most accurate predictions. However, accuracy is not the unique crucial non-functional requirement. When developing business applications from independently services (not necessary forecasting services), performance is one of the most crucial non-functional requirements. The idea is the same: providers offer similar competing services but they differ significantly in some QoS attributes (such as performance, availability and response time). Then, prospective users of services choose the best offering for their purposes.

M. Marzolla et al. [30] envisage a new approach called Multi-views Approach for Performance analysis of web Services (MAPS). They bear in mind that users and providers have different viewpoints. Users want to experience the required performance whereas providers want to reach the maximum number of users, so that their incomes are maximized. As a result they distinguish two different levels. On the one hand, the description of the application level behaviour is described as a BPEL (Business Process Execution Language) workflow in the user level. On the other hand, the provider level describes the physical resources to deploy services. Those two levels are combined and they calculate performance bounds based on operational laws of Queuing Network (QN) analysis.

Figure 15 shows an UML activity diagram with the main steps of MAPS.

In the user side, first of all, the user details functional and non-functional requirements of the application he intends to realise. According to his requirements, he discovers services and builds his application. Then, the QN model of MAPS computes performance bounds on throughput and response time by means of BPEL specifications. Finally, with the help of computed performance bounds, the user chooses among all available services those that better fulfil the performance requirements.

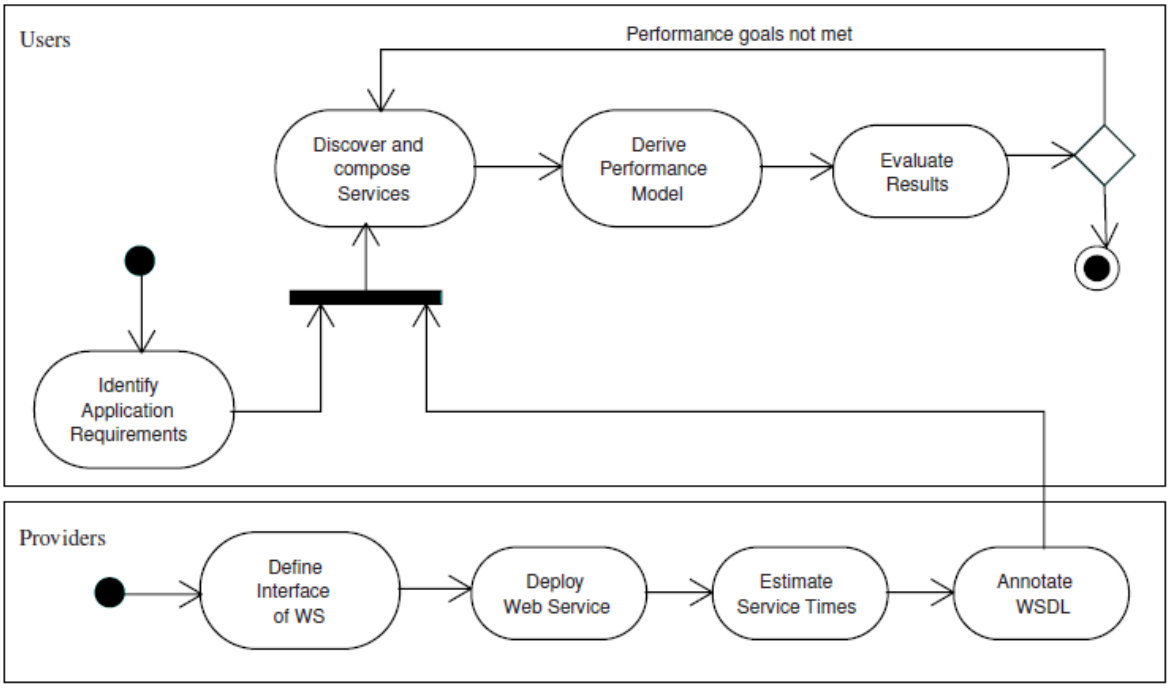

Figure 15.- Activity diagram for MAPS methodology [30].

In the provider side, providers start to carefully define the interface (WSDL) of the services they offer. After that, hardware resources are defined to deploy web services. Then, response times of each operation of each web service are saved by means of monitoring. Eventually, all these performance annotations are collected in the WSDL of the services. Performance annotations and BPEL workflows are used to compute the performance bounds.

With this methodology, a SOA system can also be reconfigured at run-time because either changes if users requirements or low performance of a single service.

S. Punitha et al. [29] also present a prototype to predict performance of service oriented applications using Queuing Network (QN) model. In order to do that, they use the AcmeStudio plugin of Eclipse. After predicting the performance, one is able to predict how far the requirements are met with the design architecture. Then, bottlenecks could be identified and reduce them by changing components of the service and calculating how much the response time becomes closer to the expected value.

In [34], a novel service-oriented monitoring system, which is called GridEye, is presented. A time-sequence-based forecasting algorithm is provided for performance prediction. In contrast with the two previous approaches, they do not use the QN model. They claimed that there are two classical methods for forecasting: k-moving average MA(k) and the exponential smoothing model ExS(α). They construct a hybrid model using both of them.

In [31], M. H. Ning et al. try to avoid performance degradation, crash failure or other unexpected effects in service-oriented applications. These problems sometimes arise because of software aging. In order to enhance reliability of systems, a maintenance technique which is called software rejuvenation is introduced. It consists of detect software aging and forecast the time when the resource exhaustion reaches the critical level. A wavelet network is used to forecast the JVM heap memory usage, which is the most important resource parameter in a service-oriented application.

The idea is to monitor a web service to get its response time and its throughput amount. These are the input parameters (x1 and x2) for the wavelet network. Then, the JVM heap memory usage is predicted (y). Figure 16 shows the basic design of the wavelet network. Detecting software aging and when a service is about to exceed the threshold of memory usage, can avoid a crash of the service and improves its reliability.

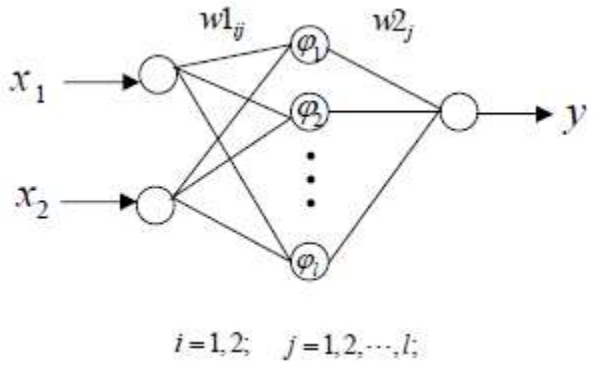

Figure 16.- Wavelet network schema to forecast the JVM heap memory usage [31].

3.4 Criteria to evaluate predictions

Murphy [18] coped with the lack of clarity and ambiguity concerning the goodness in weather forecast. For instance, the statement: “that was a good / bad forecast” is often heard, but its meaning is seldom clear. From the forecaster’s point of view, the goodness of a forecast is generally related to the degree of similarity between the forecast conditions and the observed conditions. However, users are mainly concerned with whether or not a forecast leads to beneficial outcomes while solving decision-making problems.

As a result, without an unambiguous definition of what constitutes a good forecast, the establishment of well-defined goals for any project to ensure its forecasting performance is cumbersome.

Murphy [18] identified three different ways in which a forecast can be either good or bad, as shown in Table 9.

Table 9.- Names and short definitions of three types of goodness [18].

Type	Name	Definition
1	Consistency	Correspondence between forecasts and judgments
2	Quality	Correspondence between forecasts and observations
3	Value	Incremental benefits of forecasts to users

Type 2 and, to a lesser extent, type 3 of goodness are familiar. However, many forecasters may not be familiar with the type 1 of goodness and/or the relationships among all of them.

In the following subsections those three types of goodness and their relationships are described.

3.4.1 Type 1 of goodness: consistency

A forecaster does not only derive its forecasts concerning future weather conditions from a knowledge base. A forecasting process is culminated by the formulation of forecaster's judgments regarding the occurrence/nonoccurrence of future weather events or future values of weather variables.

There exists a difference between judgments (forecaster's internal assessment which are only recorded in his mind) and forecasts (forecaster's external spoken/written statements regarding future weather conditions). Forecaster's judgments are the result of a rational process of assimilation and distillation of the information contained in his knowledge base. That is why it is reasonable to require that the forecasts, which represent the external manifestation of the judgments, correspond to the judgments.

Thus, consistency is high when a requisite forecast corresponds to a forecaster's best judgment. This is called the basic maxim of forecasting. A requisite forecast contains all of the information that potential users require to act optimally while solving decision-making problems. It varies from user to user. Since forecasters' judgments contain an element of uncertainty by definition, forecasts must be expressed in probabilistic terms.

Very high levels of consistency can be achieved by forecasters making their forecasts correspond to their judgments. In this point, type 1 goodness is distinct to types 2 and 3.

3.4.2 Type 2 of goodness: quality

Quality, or goodness in the type 2 sense, relates to the degree of correspondence between forecasts and observations. This is measured by means of forecast verification. Forecast verification consists of the computation of measures of the overall correspondence between forecasts and observations. Examples of such measures are the mean absolute error, the

mean-squared error, and various skill scores. There are two ways to calculate quality: measured-oriented approach and distributions-oriented approach. The former focuses on one or two overall aspects of forecast quality (e.g. accuracy and skill) whereas the latter avoids many of the pitfalls in the measures-oriented approach and constitutes forecast quality in its fullest sense. For example, the distributions-oriented approach allows comparing two or more sets of forecasts.

When we defined the second research question about quality criteria (see section 2.2.3), we thought that accuracy was the unique way to assess a forecast. However, accuracy is just one aspect of accuracy. Furthermore, a forecast may be assessed by means of consistency, quality and value as we have seen in Table 9.

Forecast quality is inherently multifaceted in nature. Table 10 shows various aspects of forecast quality. To see their relevant distributions, the reader is referred to [18].

Table 10.- Short definitions for various aspects of forecast quality [18].

Aspect	Definition
Bias	Correspondence between mean forecast and mean observation
Association	Overall strength of linear relationship between individual pairs of forecast and observations
Accuracy	Average correspondence between individual pairs of forecast and observations
Skill	Accuracy of forecasts of interest relative to accuracy of forecasts produced by standard of reference
Reliability	Correspondence between conditional mean observation and conditioning forecast, averaged over all forecasts
Resolution	Difference between conditional mean observation and unconditional mean observation, averaged over all forecasts
Sharpness	Variability of forecasts as described by distribution of forecasts
Discrimination 1	Correspondence between conditional mean forecast and conditioning observation, averaged over all observations
Discrimination 2	Difference between conditional mean forecast and unconditional mean forecast, averaged over all observations
Uncertainty	Variability of observations as described by distribution of observations

We can say that a forecast *f* is better in all respects than a forecast *g* when *f*'s forecasts can be sufficient for *g*'s forecasts. Then, *f* has a greater type 2 of goodness than *g*.

Type 2 of goodness is not completely under the control of the forecaster as type 1.

3.4.3 Type 3 of goodness: value

Type 3 of goodness or value relates to the benefits realized, or expenses incurred, by individuals or organisations that use forecasts to guide their choices among alternative courses of action.

Forecasts have no intrinsic value. They acquire value through their ability to help users in the context of decision-making problems. This value may be measured. For instance, it may be measured in terms of monetary benefits or expenses or in terms of nonmonetary gains or losses (such as lives saved or lost). There exist two approaches to assess value-of-information. On the one hand, the ex-post approach is concerned with determining the actual value of the forecasts after the forecasts and observations has become available. Therefore, its value relates to the actual value of a set of forecasts that have been made in the past. On the other hand, the ex-ante approach consists of determining the expected value of the forecasts before the forecasts and observations have become available. Thus, its value relates to the expected value of a set of forecasts that may be made in the future.

Obviously, as type 2, type 3 of goodness is not under the forecaster's control.

There are relationships among all three types of goodness. Consistency directly influences both quality and value. Moreover, forecast quality impacts the forecast value.

Chapter 4 Review discussion and conclusions

4.1 Discussion

4.1.1 Principal findings

After reporting all findings in Chapter 3, we will try to answer the research questions (see 2.2.3).

RQ1. to identify if there are frameworks which measure how accurate forecasting (web) services are;

Up to our knowledge, we have only found one service-oriented framework in the literature which measures how accurate forecasting web services are [19]. Our architecture (see section 6.1) presents the main advantage that is scalable (it can grow and monitor plenty of forecasting services) and really flexible since it is implemented with SOA principles.

Nevertheless, we have seen other service-oriented frameworks related with either monitoring or prediction.

We saw that there are numerous monitoring frameworks to get QoS of web services. Some of these monitoring frameworks have their own prediction models to predict the performance of web services in order to balance the workload in a composite SOA. Moreover, there are also loads of frameworks which alert when a SLA is going to be violated with the help of the previously recollected QoS attributes (to see them, the reader is referred to the related work section of [4]).

Other studies predict the performance of services either to help users during the selection process of a provider [30] or to improve the reliability of services using these performance predictions (e.g. detecting bottlenecks at design time [29], to promote efficiency [34] or to avoid a service crash because of exceeding memory usage [31]).

There is another group of service-oriented frameworks that we discussed and is not related to monitoring. Due to the benefits of SOA, predictions models which required huge amounts of data from different sources, and vast computing power are starting to use this paradigm. We saw examples in the weather forecast domain [33], macroeconomic domain [32] and drug design domain [28].

RQ2. to determine which are the main quality criteria used to evaluate predicting services;

In the beginning we thought that the unique criterion to evaluate predicting services was accuracy. However, as we saw in Table 9 and Table 10 (section 3.4) there are three types of goodness: consistency, quality and value [18]. What is more, accuracy is one aspect of forecast quality. Consistency is the correspondence between forecasts and judgments. Quality is the correspondence between forecasts and observations. Value refers to the incremental benefits of forecasts to users.

All these criteria are inside the forecast verification research field. By accuracy assessment in the title of this master thesis we actually mean forecast verification. Let us see the definition of forecast verification from [17]: "If we take the term forecast to mean a prediction of the future state (of the weather, stock market prices, or whatever), then forecast verification is the process of assessing the quality of a forecast. The forecast is compared, or verified, against a corresponding observation of what actually occurred, or some good estimate of the true outcome. The verification can be qualitative ("does it look right?") or quantitative ("how accurate was it?"). In either case it should give you information about the nature of the forecast errors."

RQ3. to identify the domain in which such frameworks are being applied;

The framework that has a similar functionality to ours (and measure how accurate forecasting services are) is being applied in the domain of weather forecast [19].

Moreover, we saw other SOA frameworks which do not measure how accurate forecasting (web) services are, but perform prediction. The domains were: weather forecast [33], macroeconomic analysis [32], and drug design [28].

Despite of that, we found plenty of forecasting web services in other domains: weather forecasts (e.g. weather bug [47]), results in betting shops (e.g. Betfair Sport API [48]) and so on. In the excluded studies, there were web services to predict flight delays, to predict protein's issues in medicine, and to predict QoS (with the final goal of improving composition of services and service's reliability).

RQ4. to identify the current knowledge about parameters that determine prediction's accuracy of these services.

In the SLR, we have not gone in depth in prediction models. In any event, we think that in the weather forecast domain, two parameters that determine prediction's accuracy of forecasting services are locality and distance of time. As on-going and further work, to discover those parameters we will use data mining tools over our current database with both real observations and weather forecasts.

4.1.2 Strengths and weaknesses

The main strength of this work is that have been done following the guidelines of B. Kitchenham [5] for systematic literatures reviews. A SLR provides scientific value by reviewing thoroughly and fairly the literature. The review protocol was defined in Chapter 2 and all the changes have been reported.

However, our subject of study is still immature. Because of that, there are not similar reviews which cover this topic. This has led to cover a wider range of topics in the SLR, although it was not the initial intention of this review.

4.1.3 Meaning of findings

Researchers and practitioners would like to know the likelihood that predictions of forecasting services are right.

For research, the review shows a clear need of a framework that monitors and analyses the predictions. A striking finding is that only one study has combined SOA with forecast verification. With the goal of developing a framework with these functionalities, parts II and III of this document are done. Monitoring the predictions we can know how accurate forecasts are and if they are improving over time. Furthermore, if we discover what the prediction models are doing wrong, they can be improved.

For practitioners, this review shows the importance of knowing to what extent one forecasting service gives better forecasts than another, and in what ways that service is better.

4.2 Conclusion

The first contribution of this master thesis is to present a state-of-the-art about forecast verification for forecasting services. For this, we have conducted a systematic review. We defined a review protocol to answer the research questions. The review protocol consisted of the data sources, the search strategy, the study selection, the study quality assessment, the data extraction and the data synthesis. We rigorously identified 213 studies from the literature with the search strategy, of which 18 passed the selection criteria and quality assessment. They have a satisfactory rigour and relevance. Fifteen of the 18 studies identified were primary studies (contributors to systematic reviews), whereas three were secondary studies (systematic reviews). Due to specialized nature of the subject of study, it can be considered as a significant number of studies. However, 50% of the studies cover the topic in a generic way. The studies covered three topics: SOA, monitoring and prediction in SOA Systems. The results were reported according to these three topics.

As a main result, we can conclude that forecast verification for forecasting services does not have an enough amount of research and deserves more attention. More effort is necessary to integrate methods for forecast verification in SOA monitoring frameworks. That is why we start with the following parts of this master thesis.

Future work is discussed in section 9.2.

Part II. Architecture

Chapter 5 Specification

5.1 General functional requirements

The following tables show the functional requirements of our architecture. It must be noted that functional requirements of this section refer to the general architecture, without taking into account the domain.

Table 11.- Functional requirement #1.

F. Req #1	The platform shall be able to compare the data predicted by forecasting services with real data.
Description and Rationale:	Comparing real data to predictions, the platform assesses the accuracy of the web services. As a result, we can know under which parameters each service is better and make a ranking.
Fit Criteria:	The platform assesses the accuracy of the prediction of a web service.

Table 12.- Functional requirement #1.1.

F. Req #1.1	The platform shall make a ranking of the forecasting services based on the accuracy of their predictions.
Description and Rationale:	By means of comparing real data to predictions, the platform indicates quantitatively how accurate is a forecasting service.
Fit Criteria:	The platform indicates quantitatively how accurate is a forecasting service.

Table 13.- Functional requirement #2.

F. Req #2	The platform shall read prediction data from several prediction sources of the forecasting domain (i.e., forecasting services).
Description and Rationale:	The platform connects to forecasting services to get their predictions.
Fit Criteria:	The platform reads forecasting data from several forecasting services.

Table 14.- Functional requirement #2.1.

F. Req #2.1	The platform shall manage prediction data that may not come from web services.
Description and Rationale:	There are backends which are not accessible as a web service. To integrate them with the platform, new web services need to be created.
Fit Criteria:	The platform is able to manage prediction data which may come from both web services and backends (e.g. xml files).

Table 15.- Functional requirement #3.

F. Req #3	The platform shall parser and save prediction data from several prediction sources of the forecasting domain (i.e., forecasting services).
Description and Rationale:	Once the web services give their response, the platform parsers it to save the desired data.
Fit Criteria:	The platform is able to save forecasting data into the database.

Table 16.- Functional requirement #4.

F. Req #4	The platform shall save real data coming from trusted sources to assess the predictions.
Description and Rationale:	In order to assess the predictions of the web services, the platform needs real data from a trusted source (also known as ground truth).
Fit Criteria:	The platform saves real data into the database.

Table 17.- Functional requirement #5.

F. Req #5	The platform shall be able to offer data to external systems.
Description and Rationale:	Information stored by the platform, could be accessible to authorized external systems (e.g. data mining systems to assess services).
Fit Criteria:	External systems are able to connect to the database of the platform.

5.1.1 Functional requirements for weather forecast

We will implement a proof-of-concept to demonstrate that the general architecture works. The chosen domain is weather forecast. We have chosen this domain because we can recollect weather data from different institutions (like government) for free. We add to the above list of general functional requirements the following ones, which are domain specific:

Table 18.- Functional requirement #2.2.

F. Req #2.2	The forecasting services currently considered are: RSS Yahoo! Weather, Open data Meteocat, AEMET.
Description and Rationale:	Data is taken from RSS Yahoo Weather, open data of Meteocat, and AEMET
Fit Criteria:	Those three services are monitored.

Table 19.- Functional requirement #4.1.

F. Req #4.1	The trusted source (also known as ground truth) currently considered is: State Meteorological Agency of Spain (AEMET).
Description and Rationale:	They offer csv files with diary summaries. These summaries give trusted information for several Spanish cities.
Fit Criteria:	The platform saves real data into the database.

Table 20.- Functional requirement #6.

F. Req #6	The platform shall give current weather forecast from all the sources for a specified city.
Description and Rationale:	All predictions of a city from external sources are gathered and show to the user.
Fit Criteria:	Users can consult the weather forecasts of a specified city.

Table 21.- Functional requirement #6.1.

F. Req #6.1	The cities currently considered are: Barcelona, Girona, Lleida, Tarragona.
Description and Rationale:	We have chosen Catalan cities because they are in all services. Yahoo! Is worldwide, AEMET covers Spanish cities and Meteocat Catalan cities.
Fit Criteria:	Forecast data and ground truth is recollected every day for these four cities.

5.2 Non-functional requirements

Our system is an extension of SALMon service oriented system [4]. That is why we accomplish most of its original non-functional requirements (from 1 to 4). In addition, we add new ones (from 5 to 7). The first 6 non-functional requirements refer to the software's architecture. The rest are related with software's behaviour.

Table 22.- Non-functional requirement #1.

NF. Req #1	The platform shall be extensible.
Description and Rationale:	Since it will be a component-based platform, we need enough extensibility capabilities for accepting new components.
Fit Criteria:	The platform is able to accept new components.

Table 23.- Non-functional requirement #2.

NF. Req #2	The platform shall be developed as a service.
Description and Rationale:	Developing the platform as a service would facilitate its integration into existing SOA Systems.
Fit Criteria:	The platform is developed as a service under the paradigm of SOA.

Table 24.- Non-functional requirement #3.

NF. Req #3	The platform should be adhered to standards.
Description and Rationale:	In order to be easily incorporated with other frameworks and accepted for the community, the development of the platform should prioritize the use of the standards.
Fit Criteria:	The platform should be compliant with current standards such as WSDL and SOAP.

Table 25.- Non-functional requirement #4.

NF. Req #4	The platform shall be able to work with continuous data flows.
Description and Rationale:	Because of the nature of the information treated, the platform should be able to work with continuous data flows.
Fit Criteria:	The platform is able to obtain and process the data continuously.

Table 26.- Non-functional requirement #5.

NF. Req #5	The platform shall use existing components when possible.
Description and Rationale:	If there are web services or service-oriented platforms which already fulfil some requirements, they should be reused.
Fit Criteria:	The platform uses existing web services or service-oriented components.

Table 27.- Non-functional requirement #5.1.

NF. Req #5.1	The platform shall be able to connect to SALMon to monitor web services.
Description and Rationale:	The web services are monitored with SALMon. SALMon saves the output of the reply, which will be used by the platform.
Fit Criteria:	The platform is able to monitor web services via SALMon.

Table 28.- Non-functional requirement #6.

NF. Req #6	The platform shall obey legal statements from external web services.
Description and Rationale:	All web services which are used are free.
Fit Criteria:	The platform only uses free web services.

Table 29.- Non-functional requirement #7.

NF. Req #7	The platform shall work when external sources are unavailable.
Description and Rationale:	There are sources which are distributed over the network and even are developed and hosted by different organisations. Thus, some of these sources may be unavailable for a period of time for reasons out of our control. Although data will not be collected from unavailable sources, the platform has to continue recollecting data from the rest of available sources.
Fit Criteria:	The platform works properly when any external source is unavailable.

We discard other non-functional requirements because of the nature of the platform. For instance, the data that we are collecting is public, so it is not necessary to handle security problems like unauthorized data to sensible data. Since the platform collects the data just once every day, efficiency is not a mandatory requirement.

5.3 Use case diagram

Although the majority of functional requirements are performed by the system, there are two actors: the service user and the service administrator. The service administrator manages the *forecasting data collector* service. After configuring this service, the system is set up to: read prediction data from several prediction sources (functional requirements #2, #2.1 and #2.2), parser and save predictions (functional requirement #3), and parser and save observations (functional requirements #4 and #4.1). On the other hand, the service user consults the *forecast verifier* service in order to see observations, predictions and services' accuracy assessment. This service verifies service's accuracy by means of comparing predictions with observations (functional requirements #1 and #1.1), offers data to external systems (functional requirement #5), and gives weather forecasts (functional requirements #6 and #6.1). Figure 17 shows the use case diagram.

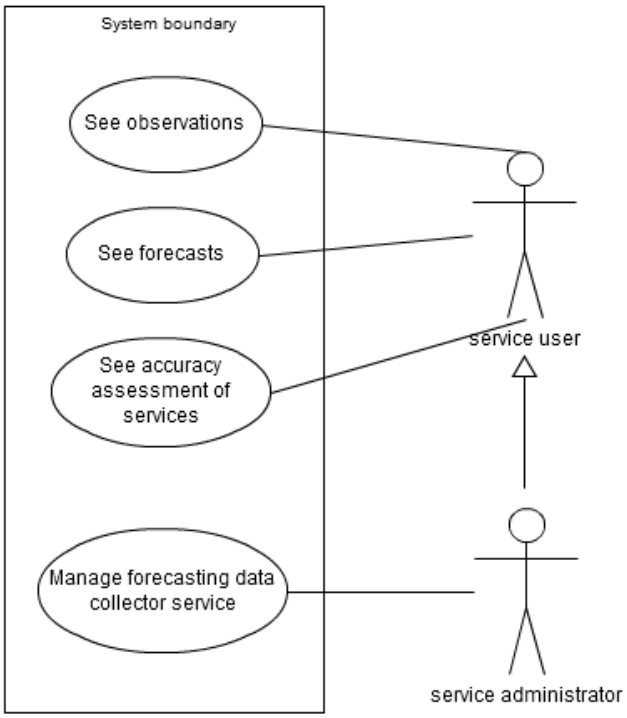

Figure 17.- Use case diagram.

5.4 Conceptual modelling of the weather forecast domain

In the weather forecast domain we deal with two concepts: observations and predictions. On the one hand, observations refer to information that is collected "on location". It is also called ground truth. We consider this information reliable, real and true, since it has been gathered by sensors. On the other hand, forecasts or predictions are statements about

future states of the atmosphere for a given location. Our purpose is to compare observations with predictions to verify forecasts. To compare observations with predictions, we take into account the attributes that they have in common (so they can be compared). "*WeatherData*" entity has these common attributes (see section 7.1.6). *Observation* and *forecast* are specialisations of the entity type "*WeatherData*". These specialisations are disjoint and complete.

Information is gathered by a *service*. There are two kinds of services: "*GroundTruthServices*" and "*WeatherForecastServices*". These relationships are disjoint and complete. The former gathers *observations* whereas the latter provides *forecasts*. Information is always related to a *city*. *Observation* is the relationship between a *city* and a *ground truth service* whereas *forecast* is the relationship between a *city*, a *weather forecast service*, and an *invocation*. An *invocation* indicates the date when a *forecast* was performed. Every *weather forecast service* realises an *operation* with its own internal city code to provide the weather forecast of a *city*.

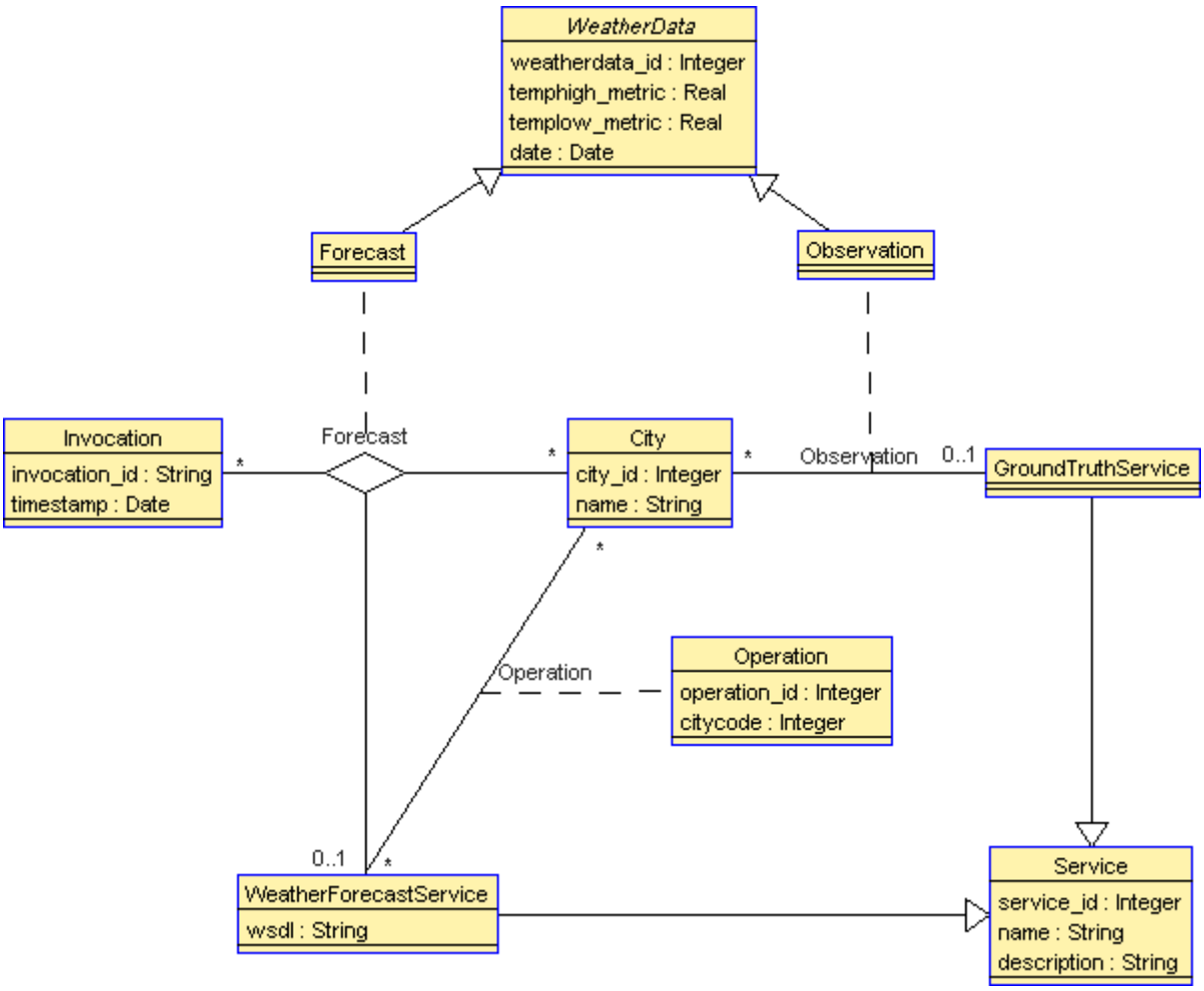

Figure 18.- Conceptual schema.

Chapter 6 Architecture

6.1 Proposed architecture

In order to fulfil the previous requirements, the architecture presented in Figure 19 was designed. This architecture extends and specialises from SALMon's architecture [4]. There are two big parts: SALMon and new resources that allow assessing the accuracy of forecasting services. These new services are: external sources, a group of forecasting web services, a service to collect all data, two databases and a web application which shows the results.

There are two kinds of external sources, the observations and the predictions. The former consists of the ground truth, that is to say observations that show real facts. The data taken from this type of external source is going to be used to verify the accuracy of the predictions. The latter contains predictions. They could come from either web services or other technologies.

The aim of *forecasting web services* is to provide forecasts. They are *external sources*. If an external source is not exposed as a web service, a proxy is developed which works as a web service. Web services have to provide forecasts following a pre-defined document format to be easily integrated.

SALMon is responsible to monitor the forecasting web services. It saves the QoS of each service and the response for every request to the web service.

A web service called *forecasting data collector*, collects both ground truth and predictions. To do so, it manages a set of parsers. The service administrator manages this web service. To save observations into the database, only a parser which called a reliable external source is needed. On the other hand, to collect predictions, the forecasting data collector uses SALMon.

There are two databases: one in charge of saving observations (*ground truth*) and another one to save predictions (*forecast data*).

Eventually, a web application verifies the accuracy of the predictions. This web application, which is called *forecast verifier*, can be used by users.

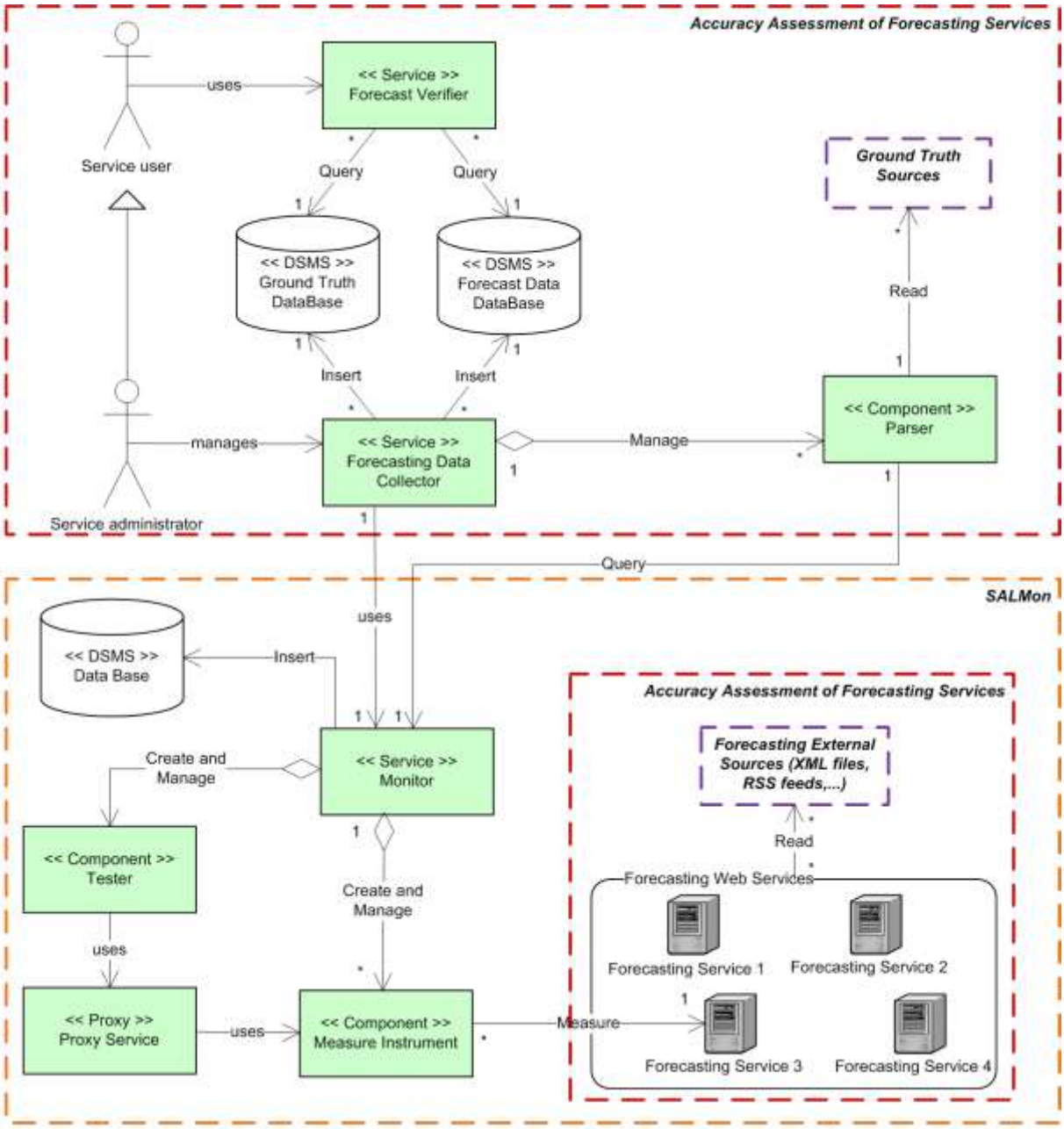

Figure 19.- General architecture for accuracy assessment of forecasting services.

We have instantiated this general architecture for the weather forecast domain. It can be seen in Figure 20.

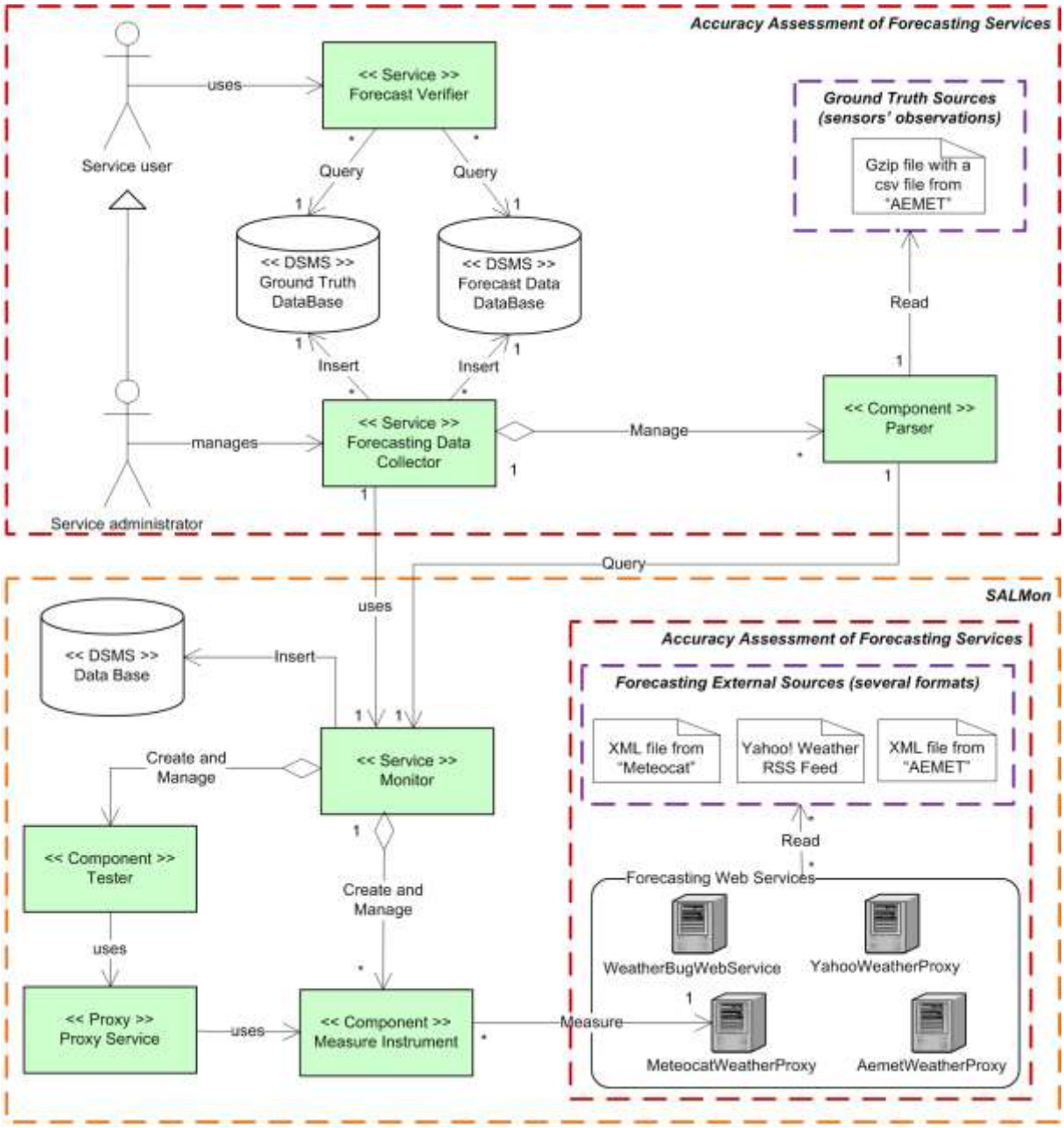

Figure 20.- Specific architecture for the domain of weather forecast.

6.1.1 Sequence diagrams

In order to show how services operate with each other and in what order, the following sequence diagrams includes their most representative interactions.

To setup the cities that have to be monitored, the administrator calls the operation *setupCatalanCities* of the *forecasting data collector* service. This operation interacts with the *monitor* service of SALMon. Firstly, the services that will be used are configured by sending information about the service like the end point (*SetService*). Then, the soap messages to get the forecasts of different cities are conducted by *SetOperation*. This is shown in Figure 21.

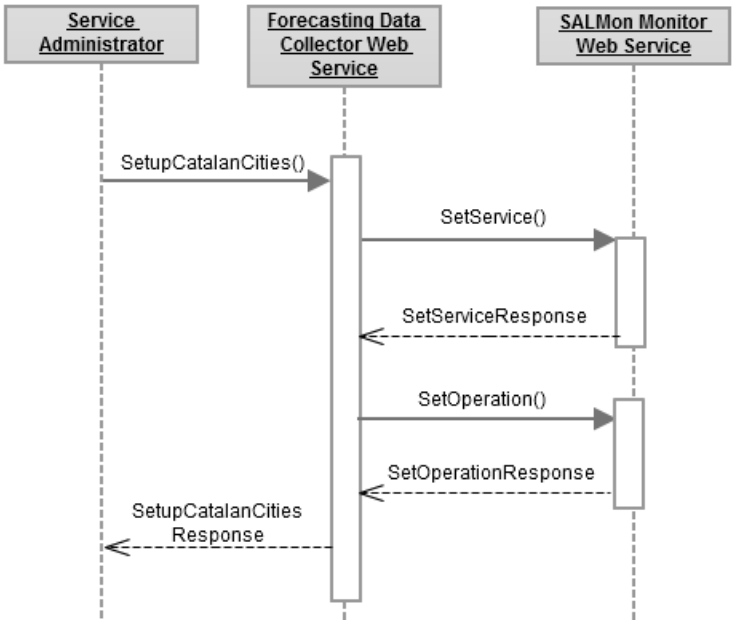

Figure 21.- Setup Catalan cities sequence diagram.

The operation *startMonitoring* of the *forecasting data collector* service interacts with the *monitor* service. The properties are sent with the operation *SetServiceProperty*, which contains the service to be monitored, the operation to be done by the service, the monitoring interval, the time out and the QoS to be retrieved. Once this operation is done, SALMon internally manages how to call periodically the services to get QoS and their output.

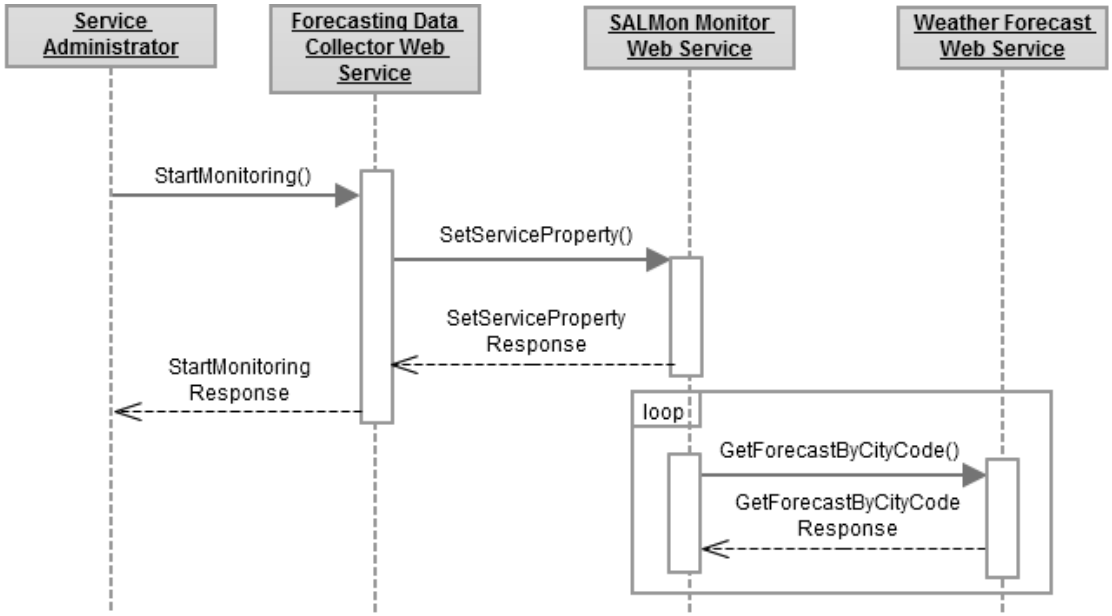

Figure 22.- Start monitoring sequence diagram.

After monitoring services, all data is in the database of SALMon. To retrieve it, the *forecasting data collector* service retrieve it by means of *GetAllInvocationInformation()*

operation. These invocations are parsed to get forecast data. The first time that the administrator wants to get invocations information, he needs to call this operation. After that, this process is automatically done.

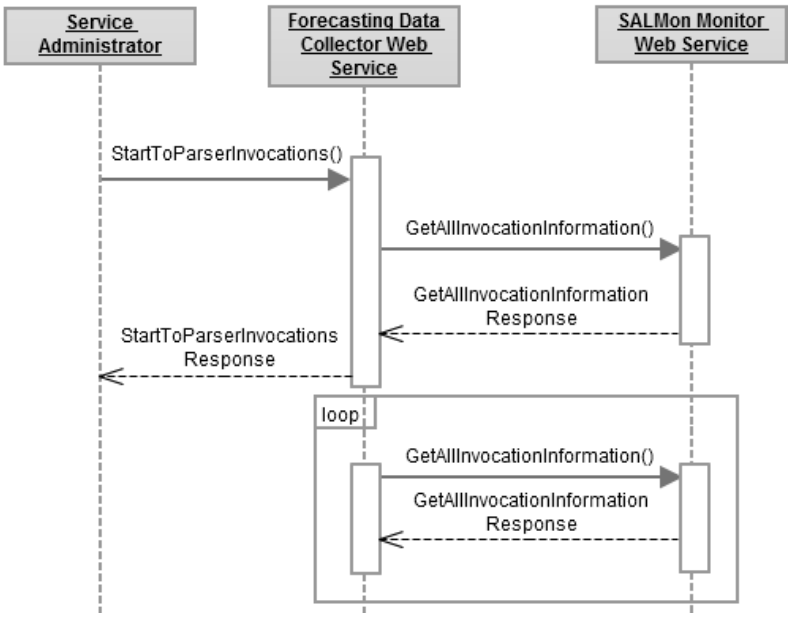

Figure 23.- Start to parser invocations sequence diagram.

When a user queries the *forecast verifier* service, forecast verification with observations is performed to provide an analysis to him. Prior to that, this service obtains information from the databases through the persistence framework.

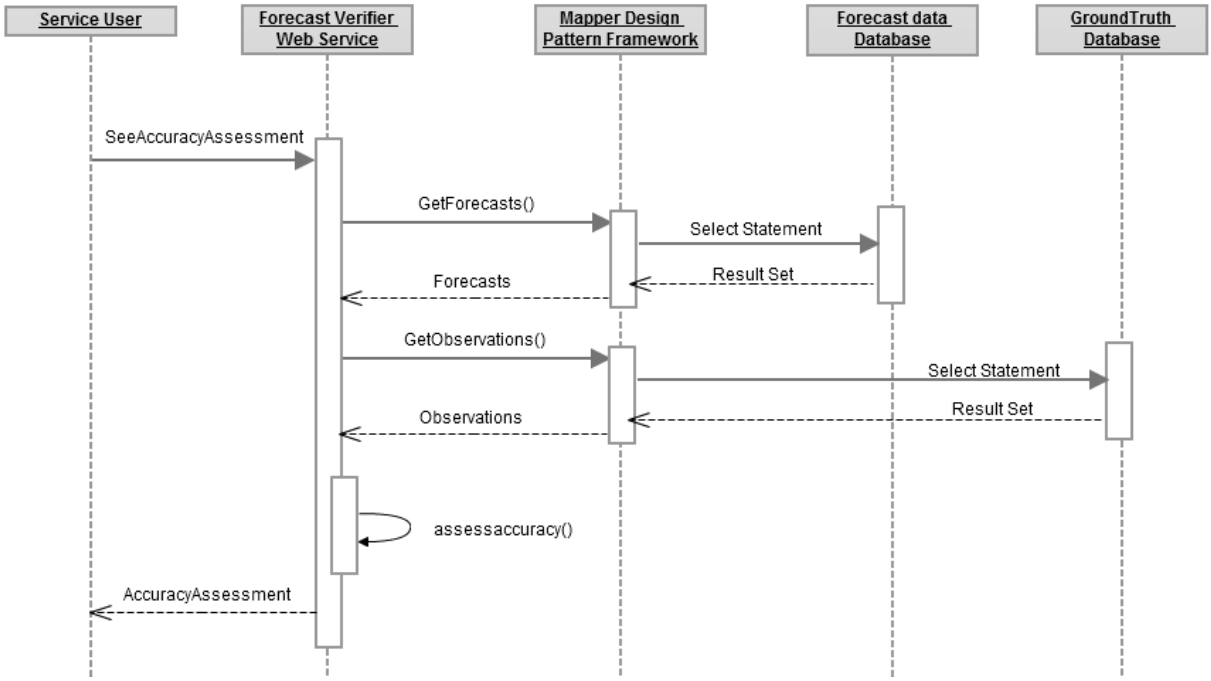

Figure 24.- See accuracy assessment sequence diagram.

6.1.2 Considerations about the architecture

As our architecture is service-oriented, it is scalable and easy to integrate with other systems and services. Our architecture is fully integrated with the monitor service of SALMon. Indeed, the current architecture is integrated with three web services of weather forecast (via SALMon), but it is scalable and new web services can be easily added. If the sources are not web services, they can be also integrated by creating a new web service as a proxy. Likewise, our databases could be accessed from external resources with another web service which would include a complete set of queries.

Integration with another system is currently on-going. This system is in charge of use data-mining techniques to extract interesting knowledge from the data and redirect users to the best service for their needs.

The platform has been developed as a service-oriented architecture. The integration with SALMon, the forecasting data collector service, is a web service which complies with the standards SOAP. It can be accessed through a WSDL and runs under Tomcat server and engine Axis2. It receives continually data from external web services, and other external services, like the ftp server of AEMET. It obeys all the legal statements of the used external sources (since our tool is non-commercial, the main legal statement is basically mentioning from where we get the data).

Part III of the present document show how this architecture has been developed and tested.

Part III. Tool

Chapter 7 Development of the tool

In this chapter, we will report the development of our tool. It should fulfil the requirements defined in Chapter 5.

The objective of the third part of this master thesis was to monitor two web services of weather forecast, using "SALMon" SOA System, as a proof of concept. Finally, we monitor three of them. All of them share the same response document format to make easy the integration of new web services. Besides, observations have to be saved. We explain how all external sources work in the section 7.1.

7.1 External sources

The web services of weather forecast should have the following characteristics:

- To provide predictions for several cities in Spain.
- To be reliable (good predictions).
- To be free.
- Not to have any restrictions when calling them.

A search to find web services with those characteristic was realised in March 2011 on the following web sites:

- http://webservices.seekda.com/
- http://webservicelist.com/
- http://xmethods.net/
- http://www.webservicex.net
- http://www.soatrader.com/
- http://www.remotemethods.com/
- http://www.wsindex.org/

Unfortunately, on the one hand, most of them just provided weather forecast for cities of United States, since their government make public this information through a web service. On the other hand, web services which provide forecasts for cities all over the world were not free. We only found one web service with the previous characteristics, although it is free for non-commercial purposes and you need to register in its web portal to be allowed to use it. It is called weather bug web service [47].

However, there were sources, without being web services, providing weather forecasts. Thus, we decided to make web services proxies which use them as a backend. In order to make as easier as possible to integrate new web services, the type of the response exposed

by the proxies is the same for all them. In this manner, the parser of the forecasting data collector service manages always the same type of output. This output contains an element with a complex type *ArrayOfApiForecastData*. This type includes several elements of complex type *ApiForecastData*. An *ApiForecastData* element consists of a set of weather forecast parameters for a city. We will see examples in the following subsections. The document format used in this response is shown in Table 30.

Table 30.- Proxies' Type of Response.

Proxies' Type of Response

```
<xsd:complexType name="ArrayOfApiForecastData">
      <xsd:sequence>
            <xsd:element minOccurs="0" maxOccurs="unbounded" name="anyType"
                  type="tns:ApiForecastData" nillable="true" />
      </xsd:sequence>
</xsd:complexType>

<xsd:complexType name="ApiForecastData">
      <xsd:sequence>
            <xsd:element minOccurs="0" maxOccurs="1" name="ConditionID"
                        type="xsd:string" />
            <xsd:element minOccurs="0" maxOccurs="1" name="Description"
                        type="xsd:string" />
            <xsd:element minOccurs="0" maxOccurs="1" name="Icon"
                        type="xsd:string" />
            <xsd:element minOccurs="0" maxOccurs="1" name="Image"
                        type="xsd:string" />
            <xsd:element minOccurs="0" maxOccurs="1" name="IsNight"
                        type="xsd:boolean" />
            <xsd:element minOccurs="0" maxOccurs="1" name="Prediction"
                        type="xsd:string" />
            <xsd:element minOccurs="0" maxOccurs="1" name="ShortPrediction"
                        type="xsd:string" />
            <xsd:element minOccurs="0" maxOccurs="1" name="ShortTitle"
                        type="xsd:string" />
            <xsd:element minOccurs="0" maxOccurs="1" name="TempHigh"
                        type="xsd:string" />
            <xsd:element minOccurs="0" maxOccurs="1" name="TempLow"
                        type="xsd:string" />
            <xsd:element minOccurs="0" maxOccurs="1" name="TempUnit"
                        type="xsd:string" />
            <xsd:element minOccurs="0" maxOccurs="1" name="Title"
                        type="xsd:string" />
            <xsd:element minOccurs="0" maxOccurs="1" name="WebUrl"
                        type="xsd:string" />
      </xsd:sequence>
</xsd:complexType>
```

We found and decided to monitor the services which are described in next subsections. Although the weather bug web service is not currently being monitored (see section 8.2).

7.1.1 Weather bug web service

Its WSDL is available on [49]. Basically, we use two operations of this web service: *GetLocationList* and *GetForecastByCityCode*. To call both of them, we need the account code obtained after the registration. The former returns the city code of the searched city, and several characteristics of the city such as country, latitude, and longitude (see Table 31). The latter needs the previously obtained city code and the unit type (english or metric). It returns a list of next seven days' forecasts (see Table 32).

Table 31.- Example of GetLocationListResponse from Weather Bug Web Service

Example of GetLocationListResponse from Weather Bug Web Service

```
<anyType xmlns="http://api.wxbug.net/"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:type="ApiLocationData">
  <City>Barcelona</City>
  <CityCode>61456</CityCode>
  <CityType>1</CityType>
  <Country>Spain</Country>
  <Latitude>41.38</Latitude>
  <Longitude>2.17</Longitude>
  <State/>
  <ZipCode/>
</anyType>
```

Table 32.- Example of GetForecastByCityCode from Weather Bug Web Service

Example of GetForecastByCityCode from Weather Bug Web Service

```
<anyType xmlns="http://api.wxbug.net/"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:type="ApiForecastData">
  <ConditionID>3</ConditionID>
  <Description>Partly Cloudy</Description>
  <Icon>cond003.gif</Icon>
  <Image>http://deskwx.weatherbug.com/images/Forecast/icons/cond003.gif
</Image>
  <IsNight>false</IsNight>
  <Prediction> Partly cloudy.  Temperature of 14&amp;amp;deg;C.   Winds SE
16km/h.  Humidity will be 64% with a dewpoint of 7&amp;amp;deg; and feels-like
temperature of 14&amp;amp;deg;C. </Prediction>
  <ShortPrediction>Partly Cloudy</ShortPrediction>
  <ShortTitle>MON</ShortTitle>
  <TempHigh>14</TempHigh>
  <TempLow>9</TempLow>
  <TempUnit>°C</TempUnit>
  <Title>Monday</Title>
  <WebUrl>http://weather.weatherbug.com/Spain/Barcelona-weather/local-
forecast/7-day-forecast.html</WebUrl>
</anyType>

[…] 6 more anyType elements (one for each day of the week)
```

7.1.2 Yahoo! weather RSS feed

It is available on [50]. The base URL for Yahoo!'s Weather RSS feed is http://weather.yahooapis.com/forecastrss. For this RSS Feed there are two parameters:

- w for WOEID. A WOEID (Where on Earth IDentifier) is a unique reference identifier assigned by Yahoo! to identify any feature on Earth. To find the WOEID of a city, one browses or searches the city on http://weather.yahoo.com/. The WOEID is in the URL for the forecast page for that city. For instance, if one searches for Barcelona, the forecast page for that city is http://weather.yahoo.com/spain/catalonia/barcelona-753692/. Thus, its WOEID is 753692.
- u for degrees units (Fahrenheit or Celsius). It could be either 'f' (Fahrenheit) or 'c' (Celsius).

So, Barcelona's RSS feed is: http://weather.yahooapis.com/forecastrss?w=753692&u=c

Its response includes several elements containing metadata about the feed itself, information about location, units, wind, atmosphere, astronomy and condition, but we are just interested on *yweather:forecast* elements for today and tomorrow (in bold in Table 33). Its attributes are: day of the week to which this forecast applies, the date to which this forecast applies, the forecasted low temperature for this day (in the units specified by the *yweather:units* element), the forecasted high temperature for this day, a textual description of conditions and the internal condition code for this forecast.

The web service YahooWeatherProxy reads this RSS feed to return an *ArrayOfApiForecastData*, as it was defined in Table 30.

Table 33.- Yahoo! Weather RSS Feed Response.

Yahoo Response

```
<?xml version="1.0" encoding="UTF-8" standalone="yes" ?>
<rss version="2.0"
xmlns:yweather="http://xml.weather.yahoo.com/ns/rss/1.0"
xmlns:geo="http://www.w3.org/2003/01/geo/wgs84_pos#">
<channel>
<title>Yahoo! Weather - Barcelona, ES</title>
[…] More information
<item>
<title>Conditions for Barcelona, ES at 8:58 am CEST</title>
[…] More information
<yweather:forecast day="Fri" date="17 Jun 2011" low="20" high="25"
text="Partly Cloudy" code="30" />
<yweather:forecast day="Sat" date="18 Jun 2011" low="18" high="24"
text="Mostly Sunny" code="34" />
<guid isPermaLink="false">SPXX0015_2011_06_17_8_58_CEST</guid>
</item>
</channel>
</rss>
```

7.1.3 Open data Meteocat

Open data is a philosophy and a practice promoted by the Catalan government [51]. It requires that certain data are of free access to all, with no technical or legal limitations. In the public sector, having access to data from the administration guarantees transparency, efficiency and equal opportunities, and also creates value.

Since March 2011, with the collaboration of Meteocat (the Catalan meteorology service), they provide a XML file with forecasts for all 41 Catalan counties. The XML file is updated every day with forecasts for next 2 days. The prediction is given by the "prediccio" element. It contains a "variable" element with attributes: the date to which the forecast applies, a number which indicates the distance from today, two names of the images which represent the forecast in a graphical way during the morning and afternoon respectively, the forecasted high temperature for this day, the forecasted low temperature for this day, the probability of storm in the morning (from 1 to 4), the probability of storm during the afternoon, the probability of hail in the morning, the probability of hail during the afternoon.

Table 34.- Example of XML of county forecast from Meteocat open data.

XML of county forecast from Meteocat open data.
Available on: http://www.meteo.cat/servmet/opendata/ctermini_comarcal.xml

```
<smc datacreacio="16-06-2011 12:07:39">
<comarca id="1" nomCOMARCA="L'Alt Camp" nomCAPITALCO="Valls"/>
<comarca id="2" nomCOMARCA="L'Alt Empordà" nomCAPITALCO="Figueres"/>

[…] 39 more comarca elements (one for each Catalan county)

<simbol id="1" nomsimbol="sol"/>
<simbol id="2" nomsimbol="sol i núvols alts"/>
<simbol id="3" nomsimbol="mig ennuvolat"/>
<simbol id="4" nomsimbol="cobert"/>
<simbol id="5" nomsimbol="plugim"/>
<simbol id="6" nomsimbol="pluja"/>
<simbol id="7" nomsimbol="xàfec"/>
<simbol id="8" nomsimbol="tempesta"/>
<simbol id="9" nomsimbol="calamarsa"/>
<simbol id="10" nomsimbol="neu"/>
<simbol id="11" nomsimbol="boira"/>
<simbol id="12" nomsimbol="boirina"/>
<simbol id="13" nomsimbol="xàfec de neu"/>
<tempesta id="1" nomprobtempmati="Baix" nomprobtemptarda="Baix"/>
<tempesta id="2" nomprobtempmati="Moderat" nomprobtemptarda="Moderat"/>
<tempesta id="3" nomprobtempmati="Alt" nomprobtemptarda="Alt"/>
<tempesta id="4" nomprobtempmati="Molt alt" nomprobtemptarda="Molt alt"/>
<calamarsa id="1" nomprobcalamati="Baix" nomprobcalatarda="Baix"/>
<calamarsa id="2" nomprobcalamati="Moderat" nomprobcalatarda="Moderat"/>
<calamarsa id="3" nomprobcalamati="Alt" nomprobcalatarda="Alt"/>
<calamarsa id="4" nomprobcalamati="Molt alt" nomprobcalatarda="Molt alt"/>
<!--PREDICCIO PER COMARQUES -->
<prediccio idcomarca="1">
      <variable data="17-06-2011" dia="1" simbolmati="1.png"
```

```
      simboltarda="1.png" tempmax="27" tempmin="15" probtempmati="1"
      probtemptarda="1" probcalamati="1" probcalatarda="1"/>
      <variable data="18-06-2011" dia="2" simbolmati="1.png"
      simboltarda="1.png" tempmax="26" tempmin="13" probtempmati="1"
      probtemptarda="1" probcalamati="1" probcalatarda="1"/>
</prediccio>
<prediccio idcomarca="2">
      <variable data="17-06-2011" dia="1" simbolmati="3.png"
      simboltarda="1.png" tempmax="28" tempmin="19" probtempmati="1"
      probtemptarda="1" probcalamati="1" probcalatarda="1"/>
      <variable data="18-06-2011" dia="2" simbolmati="1.png"
      simboltarda="1.png" tempmax="27" tempmin="18" probtempmati="1"
      probtemptarda="1" probcalamati="1" probcalatarda="1"/>
</prediccio>

[…] 39 more prediccio elements (one for each Catalan county)

</smc>
```

7.1.4 AEMET forecasts

A XML file with weather forecast of a locality can be downloaded from AEMET web site since 30th May 2011. As Meteocat did through the open data project, AEMET has changed its data policy improving and increasing the given data to the public [52].

When one looks a city up on their web site, the URL contain an identifier of the city. For example, if we search Barcelona, we can know that its id is 08019 from its URL: http://www.aemet.es/es/eltiempo/prediccion/municipios/barcelona-id08019. The XML file with the prediction can be retrieved using the city code in the following base URL: http://www.aemet.es/xml/municipios/localidad_. For example, Barcelona's XML is on http://www.aemet.es/xml/municipios/localidad_08019.xml, where, as we showed, 08019 is the code of the city.

Prediction includes: rainfall probability, snow level (m), sky conditions, wind, highest wind gust, temperatures (Celsius), real feel (Celsius), relative humidity (%) and maximum UV rate. For the first two days data is given in 6-hours intervals whereas for the two following days it is given in 12-hours intervals. Last three days predictions have just one interval (24-hours interval).

Table 35.- Example of XML file with forecasts from AEMET.

XML file of Barcelona from AEMET

```
<root id="08019" version="1.0"
xsi:noNamespaceSchemaLocation="http://www.aemet.es/xsd/localidades.xsd">
<origen>[…] Information about AEMET</origen>
<elaborado>2011-06-17T09:25:10</elaborado>
<nombre>Barcelona</nombre><provincia>Barcelona</provincia>
<prediccion>
      <dia fecha="2011-06-17">
            <prob_precipitacion periodo="00-12">30</prob_precipitacion>
            <prob_precipitacion periodo="12-24">25</prob_precipitacion>
            <prob_precipitacion periodo="00-06">5</prob_precipitacion>
            <prob_precipitacion periodo="06-12">30</prob_precipitacion>
            <prob_precipitacion periodo="12-18">25</prob_precipitacion>
            <prob_precipitacion periodo="18-24">25</prob_precipitacion>
            <cota_nieve_prov periodo="00-12" />
            […] 5 more cota_nieve_prov elements (one for each interval)
            <estado_cielo periodo="00-12" descripcion="Intervalos
      nubosos">13</estado_cielo>
            […] 5 more estado cielo elements (one for each interval)
            <viento periodo="00-12">
                  <direccion>E</direccion>
                  <velocidad>15</velocidad>
            </viento>
            […] 5 more viento elements (one for each interval)
            <racha_max periodo="00-12" />
            […] 5 more cota_nieve_prov elements (one for each interval)
            <temperatura>
                  <maxima>26</maxima>
                  <minima>19</minima>
                  <dato hora="06">21</dato>
                  <dato hora="12">24</dato>
                  <dato hora="18">24</dato>
                  <dato hora="24">21</dato>
            </temperatura>
            <sens_termica>
                  <maxima>26</maxima>
                  <minima>19</minima>
                  <dato hora="06">21</dato>
                  <dato hora="12">24</dato>
                  <dato hora="18">24</dato>
                  <dato hora="24">21</dato>
            </sens_termica>
            <humedad_relativa>
                  <maxima>85</maxima>
                  <minima>65</minima>
                  <dato hora="06">80</dato>
                  <dato hora="12">65</dato>
                  <dato hora="18">65</dato>
                  <dato hora="24">80</dato>
            </humedad_relativa>
            <uv_max>10</uv_max>
      </dia>
      […] 6 more dia elements (one for each day of week)
      </prediccion>
</root>
```

7.1.5 Ground truth source

Among several institutions, such as European Centre for Medium-Range Weather Forecasts, AEMET and Meteocat, we decided to take the observations from AEMET, since they provide the data in a more adequate way for us. AEMET has more than 700 stations with sensors measuring observations.

They offer daily summaries [53] which includes: identifier of the station, name of the station, province, altitude of the station (m), maximum temperature (Celsius), time of maximum temperature (in format hh:mm), minimum temperature (Celsius), time of minimum temperature (in format hh:mm), average temperature (Celsius), maximum wind gust (m/sec), time of maximum wind gust, maximum wind speed in 10-minutes intervals, time of maximum wind speed (in format hh:mm), daily rainfall (mm), rainfall in 6-hours intervals. The times are always referenced in Coordinated Universal Time (UTC).

The information is contained in CSV files and compressed with gzip. The filenames follow the pattern: YYYYMMDD_resudia.csv.gz, where "YYYY" is the year, "MM" is the month and "DD" is the day. Only CSV files of last week can be obtained. The base URL is: ftp://ftpdatos.aemet.es/datos_observacion/resumenes_diarios/. If we want to get the observations of the day 09/06/2011, they can be obtained (only next 7 days) on ftp://ftpdatos.aemet.es/datos_observacion/resumenes_diarios/20110609_resudia.csv.gz.

7.1.6 Comparative table and decisions

We compare information given by the previous sources in Table 36. The "greatest common divisor" of the information provided by the 4 forecasting web services and the ground truth source is:

- Maximum temperature
- Minimum temperature
- Unit of temperature
- Date of forecast/observation
- Location of forecast/observation

This is the information saved into the database to compare web services of weather forecast. However, this information is not given in the same format by all external sources. Maximum and minimum temperatures are given as a natural number by forecasting sources whereas they are given as a real number with one decimal of precision by ground truth source. For this reason, forecasts could be mistaken even when they are right. What is more, observation may be taken in several places (centre, airport…) of a city whereas predictions refer to a region or area. In order to compare them, we take the most centric place of the observations.

Table 36.- Comparative table of information offered by forecasting services.

	Weather Bug	Yahoo!	Meteocat	AEMET forecasts	AEMET observations	Common information
Condition Identifier	×		×	×		
Textual Description	×	×	×	×		
Icon or Image	×	×	×	×		
Temp High	×	×	×	×	×	✓
Temp Low	×	×	×	×	×	✓
Temp Unit	×	×	×	×	×	✓
Date	×	×	×	×	×	✓
External URL	×	×				
Language		×		×		
Location	×	×	×	×	×	✓
Wind				×	×	
Atmosphere (e.g. humidity)		×		×		
Astronomy (e.g. sunrise)		×				
Rainfall probability			×	×		
Hail probability			×			
Snow level				×		
Real feel		×		×		
Time of events					×	
Daily rainfall					×	

7.2 Database diagram

As we have already seen in the Figure 20, there are two databases: ground truth database, which saves real observations taken from sensors, and forecast data database, which keeps record of the forecast made by web services. In Figure 18 we made a conceptual schema for the weather forecast domain. In this section, we explain the schema of the databases.

The ground truth database, whose diagram is shown in Figure 25, consists of the following tables: observation and city. One city has many observations. Observation table has the following attributes: an auto generated primary key, maximum temperature in Celsius, minimum temperature in Celsius and the date of this temperature.

The forecast data database consists of the following tables: city, forecast, invocation, operation and web service. The forecast table has the same attributes as the observation table of the previous database. A forecast for a city is given by a web service, and it is done in the date of its invocation. The city table includes all the city identifiers of external sources and a name. Web service table consists of its primary key (this identifier has the same value

as “service_id” of the service table of SALMon), its name, wsdl and description. Invocation table has a primary key and the date of the invocation (made by SALMon). With the help of the operation table we can know the operation_id (this identifier has the same value as operation_id of the operation table of SALMon) which uses SALMon to request to a web service the weather forecast of a specific city.

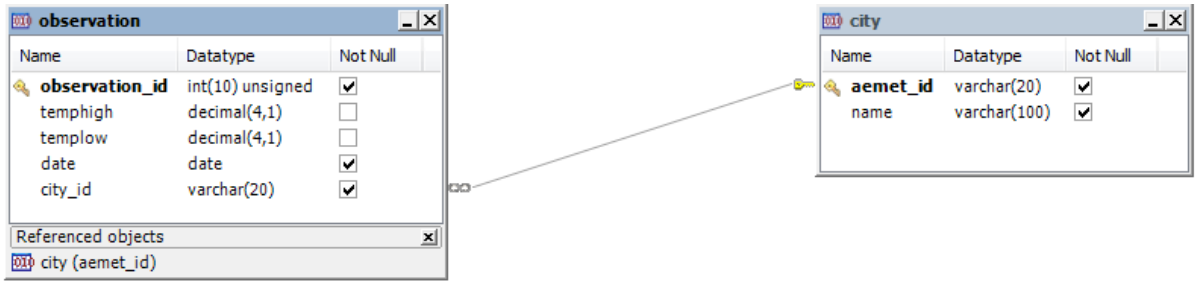

Figure 25.- Diagram of ground truth database.

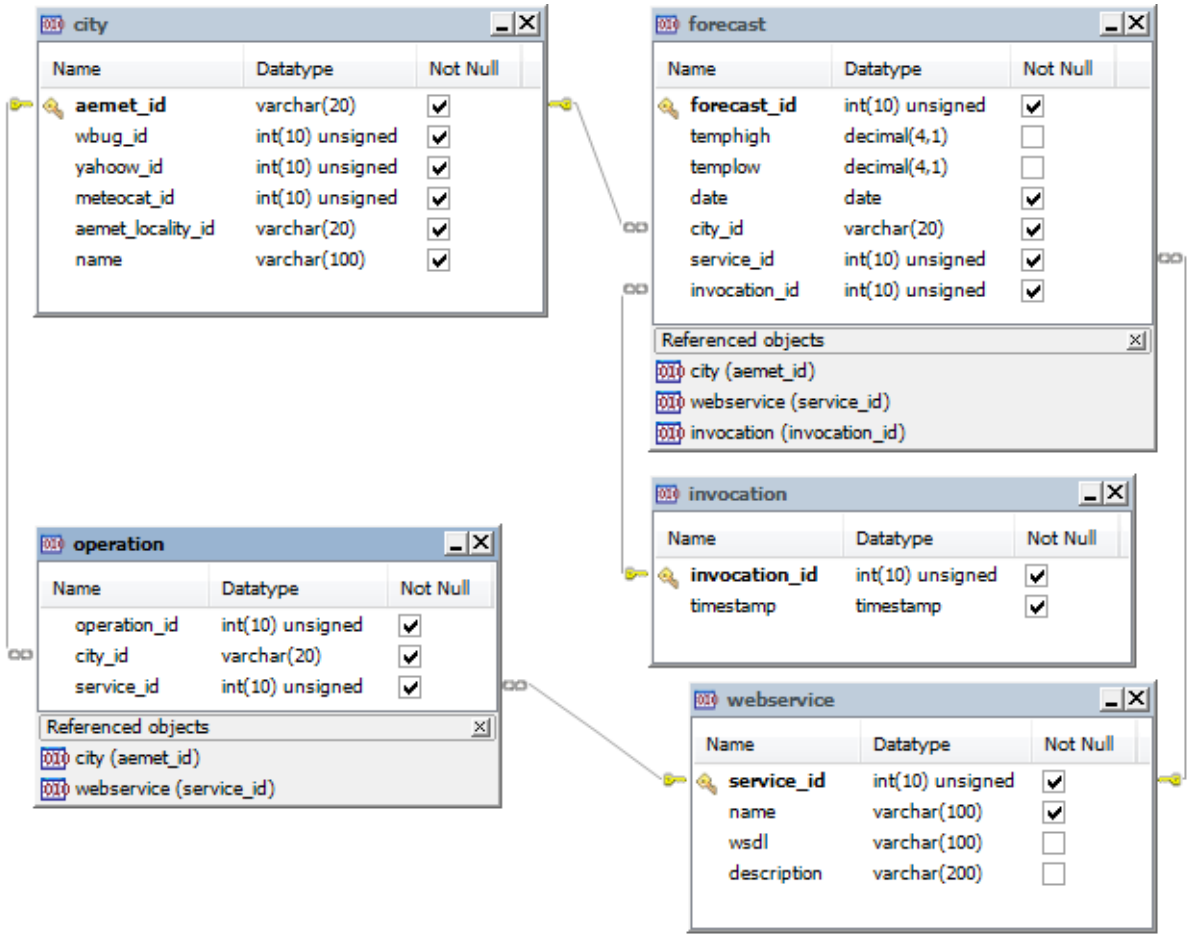

Figure 26.- Diagram of forecast data database.

7.3 Forecast verifier

To verify the correctness of forecasting services, we use the mean squared error and the approximation error. The mean squared error quantifies the difference between values implied by an estimator and the true values. Approximation errors are typically calculated in laboratories, where errors can occur because either the measurement of the data is not precise (due to the instruments), or approximations are used instead of the real data (e.g., 3.14 instead of π). In our case, the true values are observations and the estimations are forecasts.

7.3.1 Mean squared error

In statistics, the mean squared error (MSE) [54] of an estimator is one of many ways to quantify the difference between values implied by an estimator and the true values of the quantity being estimated. MSE is a risk function, corresponding to the expected value of the squared error loss or quadratic loss. MSE measures the average of the squares of the "errors". The error is the amount by which the value implied by the estimator differs from the quantity to be estimated. The difference occurs because of randomness or because the estimator does not account for information that could produce a more accurate estimate. The formula to calculate MSE is:

$$MSE_{WS} = \frac{\sqrt{\sum_{i=1}^{n}(\mu_T - T_i)^2}}{n}$$

where *WS* is web service, μ_T is the average of temperatures (high or low) of the observations, T_i is one predicted temperature (high or low) of the web service, *i* is an index (which refers to a date), and *n* is the total amount of temperatures of the web service.

This formula is applied separately to high and low temperatures.

7.3.2 Approximation error

The approximation error [55] in some data is the discrepancy between an exact value and some approximation to it. The formula is:

$$Approximation\ Error_{WS} = \frac{\sum_{i=1}^{n}|T_{GT_i} - T_{WS_i}|}{\sum_{i=1}^{n}|T_{GT_i}|}$$

where *WS* is web service, *T* indicates temperature (high or low), *GT* refers to ground truth, *i* is an index (which refers to a date), and *n* is the total amount of temperatures of the web service.

We do not take the absolute value of each temperature because they can be positive or negative. This formula is applied separately to high and low temperatures.

7.4 Details about the implementation

7.4.1 Generally used technologies

The general technologies (programming languages and development tools) that have been used for the development process and the reasons for their selection are briefly presented in this section.

In general, we have been using (when possible) the same technologies as the ones that were used to the development of SALMon [3]. In this sense, our tool has been implemented in Java (jdk1.6.0_23) too. We also used the Java EE version of Eclipse. This integrated development environment (IDE) has a web service explorer and web service tools, which make easier working with web services. We used Tomcat and Axis2, that can be integrated with Eclipse. Tomcat is an application server. Axis2 is a web service engine for the Java programming language.

Regarding to the storage of data, MySQL has been the database chosen. We used the software application Toad for MySQL to administrate the database.

To do the diagrams of the architecture we used Microsoft Visio. For the conceptual schema of the database, Use has been used.

The forecast verifier web application uses jQuery and highcharts javascript libraries to draw graphics.

7.4.2 Technical details

To completely integrate our tool with SALMon, SALMon has added a new operation in its monitor service. Moreover, we have implemented the interaction of the database and the schedule of parser operations as SALMon [3].

- The monitor service of SALMon added for the purposes of this master thesis one method (GetAllInvocationInformation) which responses is analogous to GetAllInputInformationFromService, but uses the type InvocationOutput instead of Invocation. Besides, a new field “output” in the table “operation_invocation” was added. This field saves the whole response of the service which is monitored.
- The architecture chosen for mapping the information into a database has been the Data Mapper pattern. In this pattern, the classes in OO-paradigm have a correspondent table in the database, and each object can be stored (in a row of the table) or accessed from the database through the usage of a mapper that have operations to create, edit, load and remove them.

- To schedule forecast data collector's operations (like parser observations and forecast data), we have been implemented them as threads. Multithreading allows scheduling operations in different time intervals without being affected between them. Hence, each operation being called is implemented as an independent thread that performs its activity on the given time interval. Therefore, if an operation crashes, it is the only one affected.

Chapter 8 Testing

8.1 Testing

In this section, we test the tool that has been created for the accuracy assessment for forecasting services. Testing consists of observing the execution of a software system to validate whether it behaves as intended and identify potential malfunctions.

As it is discussed in [24], there are four levels of testing in SOA:

- Unit testing of atomic services and service compositions. Unit testing methods aim to test individual units of source code in order to determine if they meet the functional requirements and are fit for use. A unit is the smallest testable part of an application.
- Integration or interoperability testing. During this phase individual software modules are combined and tested as a group. Integration testing usually takes as its input modules that have already been unit tested. Integration testing is more complex in SOA than in monolithical applications. This is because services are distributed over the network, developed and hosted by different organisations, and cooperating together in a totally distributed environment that even may dynamically change. Due to this complexity, the Web Services Interoperability Organization (WS-I) [56] was created to promote interoperability amongst the stack of web services specifications.
- Regression testing. It seeks to uncover new errors, or regressions, in existing functionality after changes have been made to a system (such as functional enhancements, patches or configuration changes). In service-centric systems this kind of testing is really important because integrators have a lack of control over the used services.
- Testing of non-functional properties. This kind of testing is crucial in SOA for the following reasons. SLAs between service providers and consumer must be checked. The lack of service robustness for unexpected behaviours can cause undesired effects. Services are often exposed over the Internet, so they should be protected from security attacks.

Depending on how we perform the testing inside those levels, we use either black-box or white-box testing. On the one hand, white-box testing tests internal structures or workings of an application, as opposed to its functionality. The tester requires specific knowledge of the application's code. On the other hand, black-box testing tests the functionality of an application as opposed to its internal structures or workings. Knowledge about internal structure is not required. There is a hybrid mode of testing: grey box testing. It involves having knowledge of internal data structures and algorithms for purposes of designing the test cases, but testing at the user, or black-box level. In all levels, we used grey box testing.

We perform the testing with black-box methods but since the tester was the same person as the developer, he had the knowledge of the internal infrastructure. In other words, we took into account the input and output of the software consulting the internal code of the program when it was necessary.

To design the test case, we use a template from [57]. Table 37 shows the test case planning template. First, each test case has a unique identifier. This identifier is recorded in the first column. Next, in the second column of the table, the set of steps and/or input for the particular condition one wants to test need to be described. The third column is the expected results for an input/output data – what is expected to come out of the "black box" based upon the input (as described in the "description"). In the last column, the actual results are recorded after the tests are run. If a test passes, the actual results will indicate "pass". If a test fails, results will record "fail" and a description of the failure.

Table 37.- Test Case Planning Template [57].

Test ID	Description	Expected Results	Actual Results

We started the process of software testing at the lowest level: unit testing. However, all of our units are integrated with another unit.

For the forecasting services that we developed, when we insert a valid input (correct city code and unit of measurement for temperature), we can have two results: either a successful message with the proper forecasts or an error message (if the service is unable to connect with the external source). If we insert an incorrect input, we get an error message. In any event, the services continue working after an error. We must note that the forecasts rarely do not contain all the data, because the source is not offering them yet. In these cases, forecasts services return null values. Table 38 shows the test case for the AEMET weather proxy. For the rest of forecasting services, we have the same test case.

Table 38.- Test case for AEMET Service.

Test ID	Description	Expected Results	Actual Results
1	Call the operation GetAemetForecastByCityCode of the web service AemetWeatherProxy with an existing city code.	If AEMET external source is available: return a message with the forecast. This message may have null values if they are not available in the external source. Otherwise, when it cannot reach AEMET external source, it returns an error message. The web service must continue working after this error.	Pass

2	Call the operation GetAemetForecastByCityCode of the web service AemetWeatherProxy with a non-existing or wrong city code.	It returns an error message. The web service must continue working after this error.	Pass

For our web application, we also did unit testing. Table 39 shows the test case when we ask for the last observations of one city.

Table 39.- Example of Test Case for the Forecast Verifier web application.

Test ID	Description	Expected Results	Actual Results
3	Submit a form with the amount of observations that we want to see for a specified city.	If the database is reachable, a table and a graph are showed with last observations. Otherwise, an error message is displayed.	Pass

We continue the testing process at the integration level. With the integration level we tested the working of the forecasting data collector service. This web service has three main functionalities: configuring SALMon to monitor forecasting services, retrieving forecasting service's output data and obtaining directly data of observations. We tested all this functionality with grey box techniques. Table 40 shows how we tested the work of the operation "start monitoring" of the forecasting data collector service.

Table 40.- Example of Test Case for the forecasting data collector service.

Test ID	Description	Expected Results	Actual Results
4	Start monitoring through SALMon all forecasting services that have already been set up. This is done from the forecasting data collector service.	If SALMon is available, the operation SetServiceProperty of its monitor service is called as many times as necessary (number of cities multiplied by number of forecasting services). Between every call the system waits 20 seconds not to overload SALMon. Then, SALMon calls services in a systematic manner depending in the monitoring interval. Two scenarios can happen: getting a response or not reaching a service. The former saves QoS into the database. The latter generates an error but the system continues working properly. If SALMon is unavailable, the operation returns an error message.	Pass

We did not perform regression testing, since this is the first version of the software.

Finally, non-functional properties (see section 5.2) were tested. As we discussed in previous section, efficiency and security are not important in our proof-of-concept. However, there are external sources (namely XML files from AEMET, Meteocat and Yahoo!; web services as proxies between SALMon and XML files with forecasts; SALMon; and even our databases) which are distributed over the network and even are developed and hosted by different organisations. Thus, we need to know what happen when an external source is unavailable (for reasons out of our control).

We isolated the forecasting data collector and forecast verifier services from the Internet to turn external sources unavailable. When any of these external sources is unavailable, the system generates an error message. The system continues calling every service as the monitoring interval indicates. Obviously, when error happens, the data is missing is the database, but it cannot be solved if the service which offer the data is unavailable.

The implemented system is working in production since June 28th 2011. From this day on, the forecasts of three sources (AEMET, Meteocat and Yahoo!) have been saved once a day. Also real observations taken by AEMET's sensors have been saved once a day. External sources are called four times each day in order to avoid missing data just in case they were unavailable for a little period of time. If they are unreachable a whole day, no information from this date is stored into the database. If external sources are available but they do not offer all requested data, partial data is stored into the database with null fields in case of absence of data.

8.2 Problems during the implementation

In this subsection, we will explain several problems that we had while implementing the tool. With the help of this list, we document the system in order to avoid them in the future.

- Inside the network of the university, where the server is, the port (500) for reading observations from AEMET ftp server was closed. To solve this problem, we opened this port and made a cron job to download the gzip files with the observations. Then, these files were accessed locally. Moreover, just in case that we could need them in the future, we also download the xml files from AEMET, Yahoo! Weather and Meteocat everyday as a backup.
- In localhost, we were working with the latest version of Axis2 (1.5.4, released on December 2010) and in the server the version of Axis2 was 1.3. The web service

clients generated by version 1.5.4 were not compatible with version 1.3. When using them, we got the exception addAnynomuosOperation. So we needed to install the version 1.3 to solve this problem.

- In the actual database of SALMon, the longitude of the varchar field “soap_action” was 100 characters. We increased the size to 200 characters because some soap actions were longer than 100 characters.
- In order to make debug and testing easier and possible, we asked for permission to access the log of tomcat server.
- In SALMon, the same Service cannot be set more than once. It could be used to monitor the same server with different configurations. Our problem was that we make several operations with the same server. Exactly, we called each service one time for each city. In the database design of SALMon, this is not taken into account. And the primary key for all this calls is the same (they share the end point). As a result, we need to parser the output of the call to search for the city code and identify the city.
- SALMonETe did not create properly the calls to the Weather Bug Web Service. This problem could not be solved.
- Sometimes, the forecasts values or observations are not provided. In this case, we have a null result. This could be solved by means of polynomial interpolation (e.g. Lagrange polynomials). In this sense, 0 is not the same as null.

Part IV. Final remarks

Chapter 9 Conclusions

9.1 Conclusion

This master thesis considered firstly the theoretical part, which included making an overview in the topic of forecast verification with observations based on service-oriented architecture (SOA). Afterwards, a technological part, which main goals were monitoring forecasting services through SALMon and assessing the accuracy of them, was also considered. Hence, the contributions of this work are:

- A systematic literature review (SLR).
- A general service-oriented architecture which monitories forecasting services and assess them comparing forecast values with real data from observations.
    - Moreover, we have implemented this architecture for the weather forecast domain as a proof-of-concept.

The systematic literature review provided scientific value by reviewing thoroughly and fairly the literature. As a main result, we can conclude that forecast verification based on SOA does not have an enough amount of research and deserves more attention. Only one study has combined SOA with forecast verification. More effort is necessary to integrate methods for forecast verification in SOA monitoring frameworks. On the other hand, we determined the main quality criterions used to evaluate predicting services. These are consistency, quality and value. We also studied different forecasting domains in which previous research has been carried out: weather forecast, macroeconomic analysis, drug design, results in betting shops and so on.

The second main contribution of this master thesis is the presentation and development of a generic SOA architecture which performs forecast verification with observations. This architecture is scalable and easy to integrate with other system and services. The function of the architecture is to determine which forecasting service provides better forecasts. In order to do that, it monitors forecasting services with the help of SALMon and compare forecasting values with real observations. We have implemented this architecture for the weather forecast domain. The architecture enables the user to consult observations and predictions and to calculate the errors of the predictions.

Personally, I have started to learn how to research during this period. Prior the beginning of the work, I had to expand my knowledge about systematic literature reviews, services-oriented architectures, usage of electronic databases, and conferences, journals and workshops related to my field of interest. Furthermore, I have learnt how to communicate with other developers, since I had to integrate my tool with SALMon. During the

development process, I faced and coped with the problems that came up about a new development paradigm for me: service-oriented architectures. Besides, this work was showed in the poster presentation of the First European Business Intelligence Summer School (eBISS 2011) [58].

9.2 Future work

At present, more important future work relates to:

- Increasing the amount of monitored services and recollect more data about the predictions. Currently, we monitor three forecasting services of weather forecast and we save daily maximum and minimum temperatures. By saving more information, such as rainfall probability and wind speed, assessments can be more complete and interesting.
- Redirecting to the most proper service. The final user is only interested in the most adequate service for his needs. So he does not care about the rest. Moreover, we think that each service has its own strengths and weaknesses. For instance, a forecasting service may provide good forecasts for Spain and bad ones for USA. Or it could predict better rainfall probabilities than daily temperatures. The idea is to redirect the user to the service that is more precise for his city and the atmospheric phenomenon in which he is interested (or other parameters).
- We have implemented the architecture for the weather forecast domain. In the future, we plan to implement it for more domains like stock market prices and results in betting shops.

Bibliography

References

[1] Succeeding through service innovation: A service perspective for education, research, business and government. IBM IfM - University of Cambridge Institute for Manufacturing, 2008. ISBN: 978-1-902546-65-0.

[2] **M. Oriol, et al.** Monitoring Adaptable SOA-Systems using SALMon. Madrid: Workshop on Service Monitoring, Adaptation and Beyond, 2008. Vols. ICB-Research Report, 34, pages 19 - 28.

[3] **M. Oriol.** Quality of Service (QoS) in SOA Systems. A Systematic Review. Master Thesis UPC, 2009.

[4] **M. Oriol, X. Franch and J. Marco.** SALMon: A SOA System for Monitoring Service Level Agreements. Universitat Politècnica de Catalunya Technical Report, 2010. LSI-10-18-R.

[5] **B. Kitchenham.** Procedures for Performing Systematic Reviews. Keele University Technical Report, 2004. TR/SE-0401.

[6] **P. Breretona, et al.** Lessons from applying the systematic literature review process within the software engineering domain. Journal of Systems and Software, 2007. Vol. 80, 4, pages 571 - 583.

[7] Google Scholar. [Online] Last access: July 2011. http://scholar.google.es/.

[8] ISI Web of Knowledge. [Online] Last access: July 2011. http://www.accesowok.fecyt.es/wos/.

[9] Inspec (Engineering Village). [Online] Last access: July 2011. www.engineeringvillage2.org.

[10] AMC Digital Library. [Online] Last access: July 2011. http://portal.acm.org/.

[11] IEEE Xplore. [Online] Last access: July 2011. http://ieeexplore.ieee.org.

[12] Science Direct. [Online] Last access: July 2011. http://www.sciencedirect.com/.

[13] Springer. [Online] Last access: July 2011. http://www.springerlink.com/.

[14] Scirus. [Online] Last access: July 2011. http://www.scirus.com/.

[15] DBLP. [Online] Last access: July 2011. http://www.informatik.uni-trier.de/~ley/db/.

[16] **A. Michlmayr, et al.** End-to-End Support for QoS-Aware Service Selection, Binding, and Mediation in VRESCo. IEEE Transactions on Services Computing, 2010. Vol. 3, 3, pages 193 - 205.

[17] Forecast Verification - Issues, Methods and FAQ. [Online] Last access: July 2011. http://www.cawcr.gov.au/projects/verification/verif_web_page.html.

[18] **A.H. Murphy.** What is a good forecast? An essay on the nature of goodness in weather forecasting. Weather Forecasting, 1993. Vol. 8, pages 281 - 293.

[19] **B. Domenico.** Forecast Verification with Observations: Use Case for Geosciences Web Services. [Online] 2007. http://www.unidata.ucar.edu/projects/THREDDS/GALEON/Phase2Connections/VerificationUseCase.html.

[20] **L. Bueno Ruas de Oliveira, et al.** Reference Models and Reference Architectures Based on Service-Oriented Architecture: A Systematic Review. Copenhagen, Denmark (ECSA'10). Lecture Notes in Computer Science, 2010. Vol. 6285, pages 360 - 367.

[21] **M.H. Valipour, et al.** A brief survey of software architecture concepts and service oriented architecture. Beijing, China: 2nd IEEE International Conference on Computer Science and Information Technology (ICCSIT' 09), 2009. pages 34 - 38.

[22] **M.P. Papazoglou, et al.** Service-Oriented Computing: State of the Art and Research Challenges. Computer, 2007. Vol. 40, 11, pages 38 – 45.

[23] **U. Zdun, C. Hentrich and W.M.P. van der Aalst.** A survey of patterns for service-oriented architectures. International Journal of Internet Protocol Technology, 2006. Vol. 1, 3, pages 132 – 143.

[24] **G. Canfora and M. Di Penta.** Service-Oriented Architectures Testing: A Survey. Budapest, Hungary: Proceedings of the 31st International Spring Seminar on Electronics Technology (ISSSE 2008), 2008. pages 78 - 105.

[25] **A.T. Endo and A.S. Simao.** Formal Testing Approaches for Service-Oriented Architectures and Web Services: a Systematic Review. Technical Report Instituto de Ciências Matemáticas e de Computaçao, 2010. Vol. 348. ISSN - 0103-2569.

[26] **A. Bertolino.** Software testing research: Achievements, challenges, dreams. Los Alamitos, California: Future of Software Engineering (FOSE), IEEE-CS Press, 2007.

[27] **N. Delgado, A.Q. Gates and S. Roach.** A taxonomy and catalog of runtime software-fault monitoring tools. IEEE Transactions on software Engineering, 2004. pages. 859 - 872.

[28] **R. Guha.** Flexible Web Service Infrastructure for the Development and Deployment of Predictive Models. J. Chem. Inf. Model, 2008. Vol. 48, pages 456 – 464.

[29] **S. Punitha and C. Babu.** Performance Prediction Model for Service Oriented Applications. Dalian (China). The 10th IEEE International Conference on High Performance Computing and Communications (HPCC 2008), 2008. Vol. HPCC 2008, pages 995 - 1000.

[30] **M. Marzolla and R. Mirandola.** Performance Prediction of Web Service Workflows. QoSA 2007, 2007. Vol. LNCS 4880, pages 127 – 144.

[31] **M.H. Ning, et al.** Forecasting software aging of service-oriented application server based on wavelet network with adaptive genetic algorithm. Jacksonville (Florida, USA). Fourth International Conference on Autonomic Computing (ICAC'07), 2007, page 10.

[32] **D. Han, et al.** A Service-Oriented Architecture Based Macroeconomic Analysis & Forecasting System. Frontiers of WWW Research and Development - APWeb 2006: Lecture Notes in Computer Science, 2006. Vol. 3841, pages 1107 - 1117.

[33] **C.B.C. Latha, et al.** A Service Oriented Architecture for Weather Forecasting Using Data Mining. Int. J. of Advanced Networking and Applications, 2010. Vol. 02, 02, pages 608 - 613.

[34] **N. Xiao, et al.** A service-oriented monitoring system with a forecasting algorithm of a time sequence-based hybrid model. International Journal of Parallel, Emergent and Distributed Systems, 2008. Vol. 23, 2, pages 137 - 151.

[35] **T. Dyba and T. Dingsøyr.** Empirical studies of agile software development: A systematic review. Information and Software Technology, 2008. Vol. 50, 9-10, pages 833 - 859.

[36] **M. Palacios, J. García-Fanjul and J. Tuya.** Testing in Service Oriented Architectures with dynamic binding: A mapping study. Information and Software Technology, 2010. Vol. 53, 3, pages 171 - 189.

[37] OASIS. [Online] Last access: July 2011. http://www.oasis-open.org/.

[38] Web Services Activity. [Online] Last access: July 2011. http://www.w3.org/2002/ws/.

[39] Jini. [Online] Last access: July 2011. http://www.jini.org.

[40] **J. Bisbal, et al.** Legacy Information Systems: Issues and Directions. IEEE Software, 1999. Vol. 16, 5, pages 103 - 111.

[41] **A. Arsanjani, et al.** S3: A Service-Oriented Reference Architecture. IT Professional, 2007. Vol. 9, 3, pages 10 - 17.

[42] Reference Model for Service Oriented Architecture 1.0. [Online] Last access: July 2011. http://docs.oasis-open.org/soa-rm/v1.0/soa-rm.html.

[43] **D. Peters.** Automated testing of real-time systems. Technical Report, Memorial University of Newfoundland, 1999.

[44] The R Project for Statistical Computing. [Online] Last access: July 2011. http://www.r-project.org/.

[45] R - Wikipedia. [Online] Last access: July 2011. http://en.wikipedia.org/wiki/R_%28programming_language%29.

[46] JRI - Java/R Interface. [Online] Last access: July 2011. http://www.rforge.net/JRI/.

[47] Weather Bug API. [Online] Last access: March 2011. http://weather.weatherbug.com/desktop-weather/api.html.

[48] Betfair Sports API. [Online] Last access: July 2011. http://bdp.betfair.com/index.php?option=com_content&task=view&id=33&Itemid=62.

[49] Weather Bug Web Service WSDL. [Online] Last access: June 2011. http://api.wxbug.net/weatherservice.asmx?wsdl.

[50] Yahoo! weather RSS feed. [Online] Last access: June 2011. http://developer.yahoo.com/weather/.

[51] Open data Meteocat. [Online] Last access: June 2011. http://dadesobertes.gencat.cat/ca/dades-obertes/prediccions.html.

[52] AEMET forecasts. [Online] Last access: June 2011. http://www.aemet.es/es/eltiempo/prediccion/municipios.

[53] AEMET observations. [Online] Last access: June 2011. ftp://ftpdatos.aemet.es/datos_observacion/resumenes_diarios/.

[54] Mean squared error - Wikipedia. [Online] Last access: June 2011. http://en.wikipedia.org/wiki/Mean_squared_error.

[55] Approximation error- Wikipedia. [Online] Last access: June 2011. http://en.wikipedia.org/wiki/Approximation_error.

[56] Web Services Interoperability Organization (WS-I). [Online] Last access: August 2011. http://www.ws-i.org/.

[57] Testing Overview and Black-Box Testing Techniques. [Online] Last access: August 2011. http://agile.csc.ncsu.edu/SEMaterials/BlackBox.pdf.

[58] **S. Martínez-Fernández, X. Franch and J. Bisbal.** Accuracy Assessment of Forecasting Services. *Poster presentation in First European Business Intelligence Summer School (eBISS).* Paris, France. Available on: http://cs.ulb.ac.be/conferences/ebiss2011/files/martinez.pdf, Last access: July 2011.

[59] **N. M. Josuttis.** SOA in Practice. The Art of Distributed System Design, 2007. ISBN: 0-596-52955-4.

Appendix

Appendix A: Glossary

Table 41 shows an alphabetical list of terms in the particular domain of forecast verification based on SOA with the definitions for those terms.

Table 41.- Glossary

Term	Definition
Backend	A system that maintains the data and/or business rules of a specific domain. In SOA, a backend is usually wrapped by some basic services. [59]
BPEL	Business Process Execution Language. An XML-based language used to orchestrate services to composed services or process services.
Business process	A structured description of the activities or tasks that have to be done to fulfil a certain business need.
Contract	The complete description of a service interface between one user and one provider.
Forecast	A prediction of the future state (of the weather, stock market prices, or whatever).
Forecast verification	The process of assessing the quality of a forecast.
Forecasting services	Services that provide predictions.
Ground truth	It refers to information that is collected on location. It is considered as trusted source.
Interoperability	The ability of systems using different platforms and languages to communicate with each other. Services' interoperability is achieved by supporting the protocol and data format of the service and clients. It should be define in the service contract.
Loose coupling	Coupling refers to the number of dependencies between modules. Loosely coupled modules have a few well know dependencies. SOA promote loose coupling between service users and service providers.
Monitoring	Monitoring generally means to be aware of the state of a system. A monitor is a tool that observes the behaviour of a system and determines if it is consistent with a given specification.
QN	Queuing network theory is a mathematical model used in computer systems performance analysis to predict attributes of a system and attributes of its subparts.
QoS	Quality of service.
Request	A message that is sent by a consumer as an initial message in most message exchange patterns.
Response	A message that is sent by a provider as an answer to a service request.
SALMon	SALMon is an on-going research framework. The goal of SALMon is to provide a tool for (1) monitoring the QoS of a Service, (2) report the obtained Quality information and (3) report SLA violations.
Service	Services are autonomous, platform-independent, self-contained and well-defined modules which perform software functionalities. They can be described, published, discovered and loosely coupled in novel ways. Since

	they are network-available, any service can be reused. They can be basic services or the basis to compose more complex service-oriented systems.
Service provider	The person that offer the service.
Service user	The consumer or client of the service.
SLA	A service level agreement is a part of a service contract where the level of service is formally defined.
SLR	A systematic literature review is a way to identify, evaluate and interpret all available research to a particular research question, or topic area, or phenomenon of interest.
SOA	A service-oriented architecture is essentially a collection of services that are able to communicate with each other. It is a paradigm for organizing and utilizing distributed capabilities that may be under the control of different ownership domains. It provides a uniform means to offer, discover, and interact with the use capabilities to produce desire effects consistent with measurable preconditions and expectations.
SOAP	The simple object access protocol is a protocol specification for exchanging structured information in the implementation of web services in computer networks.
Software architecture	Software architecture is a set of plans which guide the selection of architecture elements, their interactions, and the constraints of these interactions. Software architecture works as a bridge between requirements and implementation. The architecture of a system describes its gross structure.
SOC	Service-oriented computing.
UDDI	Universal description, discovery and integration is a mechanism to register and locate web service applications.
XML	The extensible markup language for documents that are supposed to contain structured information and that are supposed to be exchanged among different systems and platforms.
Web service	Web services are specialized SOA implementations that embody the core aspects of a service-oriented approach to architecture. Therefore, Web services just indicate a collection of technologies, such as SOAP and XML.
WSDL	The web services description language is an XML-based language that is used for describing the functionality offered by a Web service.

Appendix B: User manual and demo

The tool, which is called *forecast verifier* in the architecture, is available on http://gessi.lsi.upc.edu/accuracyassessment/. It is a web application that enables users to analyse the best service for their needs. The user can consult: observations, predictions and its errors.

The index page is shown in Figure 27. Under the title of the page, there is a menu with four options:

- Home, which is the start page.
- Observations, to see last x observations of a city.
- Predictions made in a specified day, to get all weather forecasts made in the chosen date of invocation (the day that forecasts were taken by the web services) for a specific city.
- Predictions made for a specified day, which serves to get all weather forecasts made for a specific day (the predictions for this day) and for a specific city.

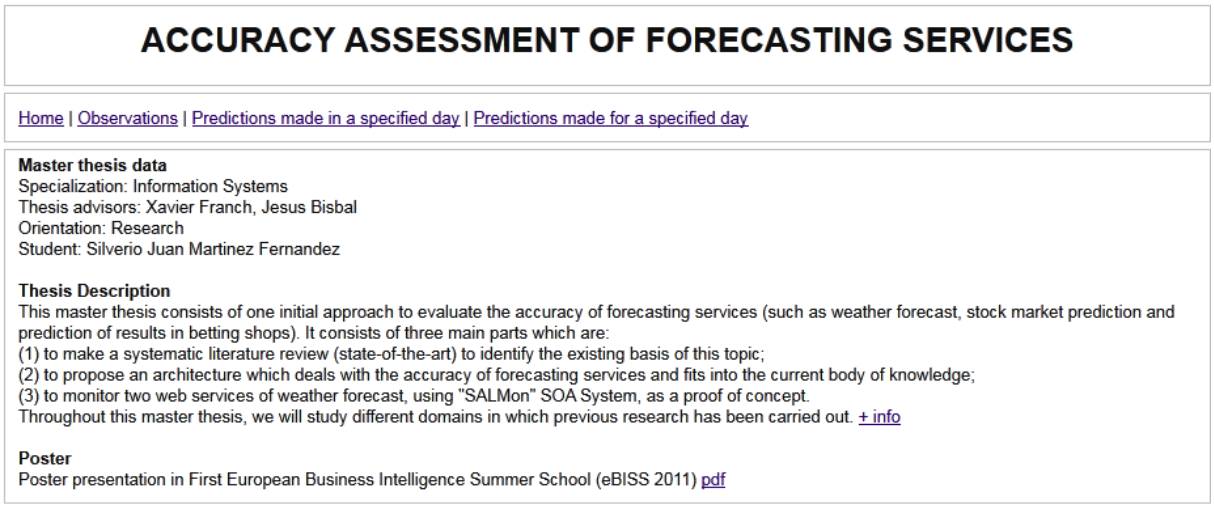

Figure 27.- Main screen of the forecast verifier application.

The following subsections illustrate how the different pages work.

1. Page of observations

In the observations page, real observations taken by AEMET's sensors in one particular city can be requested. The form that is shown in Figure 28 has as input parameters the number of days and the city to be consulted.

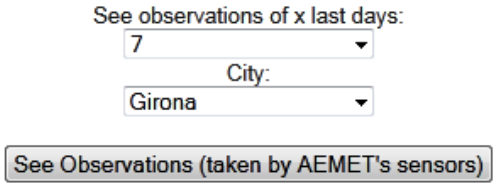

Figure 28.- Form of observations menu.

After clicking the button “see observations”, maximum and minimum temperatures of the city are shown. They are shown in two ways: with a table (Figure 29) and with a graph (Figure 30). The table has three columns: date, which refers to the date of the observation; max, which is the maximum temperature of the day in Celsius; and min, which is the minimum temperature of the day in Celsius. The graph is a line chart which contains two lines with the maximum and minimum temperatures of last days. The X-axis contains the date of the observations and the Y-axis the temperature in Celsius. If we put the mouse over the observation of one day, numerical values are shown.

Date	Max	Min
Tue, 9 Aug 2011	28.1	17.7
Wed, 10 Aug 2011	30.2	16.2
Thu, 11 Aug 2011	31.6	15.0
Fri, 12 Aug 2011	33.2	17.7
Sat, 13 Aug 2011	33.4	19.3
Sun, 14 Aug 2011	33.9	19.8
Mon, 15 Aug 2011	33.3	20.5

Figure 29.- Example of a table with observations.

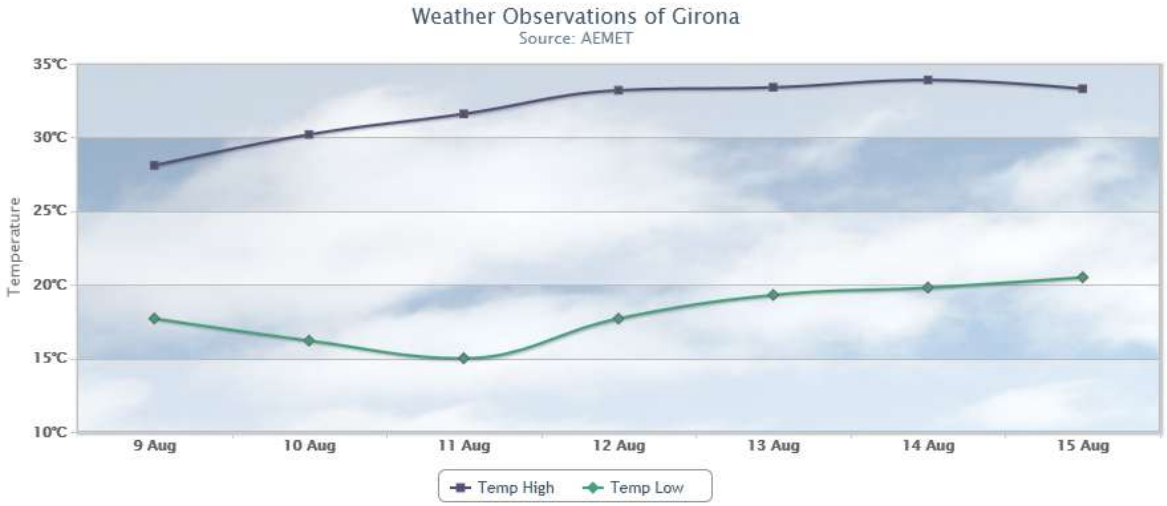

Figure 30.- Example of a graph with observations.

2. Page of predictions made in a specified day

The “predictions made in a specified day” page includes a form to get all weather forecasts made in the chosen date of invocation (the day that forecasts were taken by the web services) for a specific city. The input parameters are the date of invocation and the city.

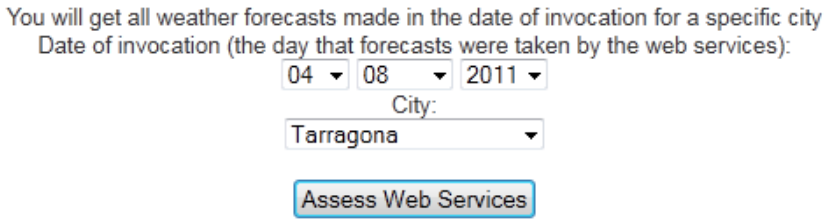

Figure 31.- Form of the predictions made in a specified day page.

This query returns all weather forecasts made in a specified day. Furthermore, forecast verification with observations is performed by means of mean squared error and approximation error.

First of all, a table with all weather forecasts made by all web services is shown (see Figure 32). This table contains 6 columns. The first one indicates the day of the forecast. In other words, all information displayed in a row refers to the same day. The second and third columns represent real maximum and minimum temperature respectively. The fourth and fifth columns contain the predicted maximum and minimum temperatures respectively. For instance, the highlighted row of Figure 32 contains predictions that were taken on August 4th but predict the weather for August 6th. Finally, the last column indicates the forecasting web service that performed the forecast.

Date Forecast	Real TempHigh	Real TempLow	Predicted TempHigh	Predicted TempLow	Web Service
Thu, 4 Aug 2011	28.4	20.6	31.0	21.0	AemetWeatherProxy
"	"	"	29.0	21.0	YahooWeatherProxy
Fri, 5 Aug 2011	28.3	22.9	29.0	22.0	AemetWeatherProxy
"	"	"	29.0	21.0	YahooWeatherProxy
"	"	"	26.0	20.0	MeteocatWeatherProxy
Sat, 6 Aug 2011	28.0	24.1	29.0	22.0	AemetWeatherProxy
"	"	"	28.0	20.0	MeteocatWeatherProxy
Sun, 7 Aug 2011	27.0	22.8	31.0	22.0	AemetWeatherProxy
Mon, 8 Aug 2011	27.6	21.5	30.0	22.0	AemetWeatherProxy
Tue, 9 Aug 2011	30.0	16.7	28.0	19.0	AemetWeatherProxy
Wed, 10 Aug 2011	26.7	16.8	28.0	19.0	AemetWeatherProxy

Figure 32.- Example of a table with predictions made a specified day.

The same information appears at the end of the page in graphs. There are two line charts with high (see Figure 33) and low temperatures (see Figure 34) respectively. The X-axis contains the date of the observations and forecasts and the Y-axis the temperature in Celsius. Observations are plotted with rectangles and then joined with a thick continuous blue line. Forecasts are plotted with different shapes and then joined with a thin dashed line. Each of these dashed lines includes forecasts made by the same web service.

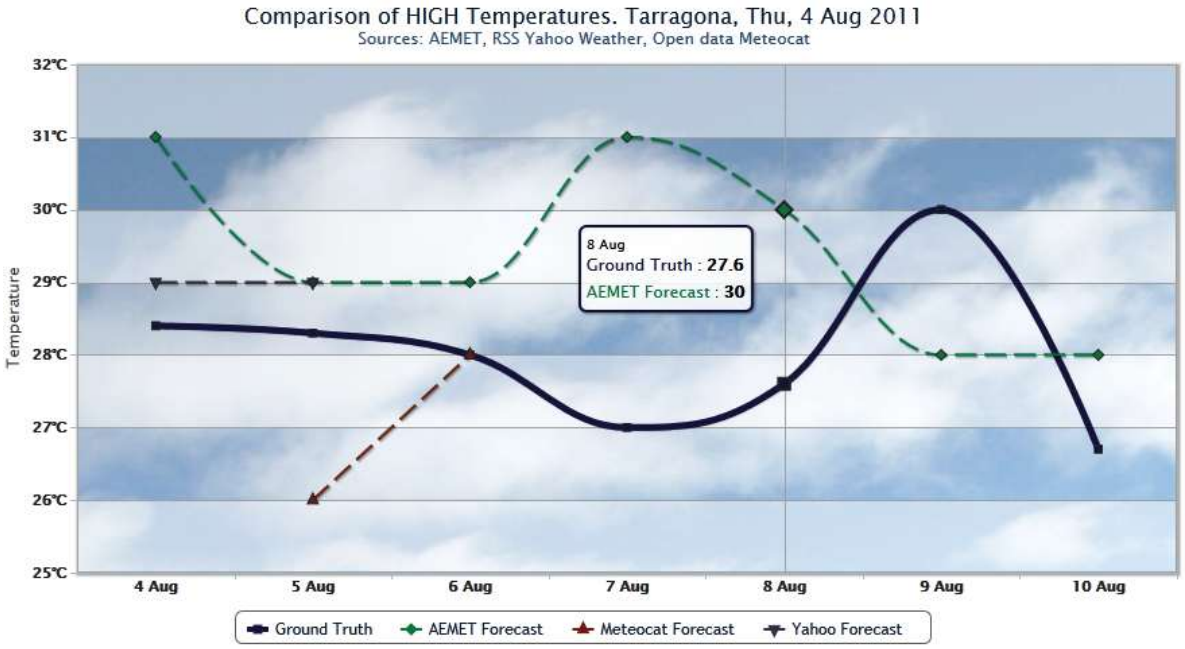

Figure 33.- Graph with high temperatures in the predictions made a specified day page.

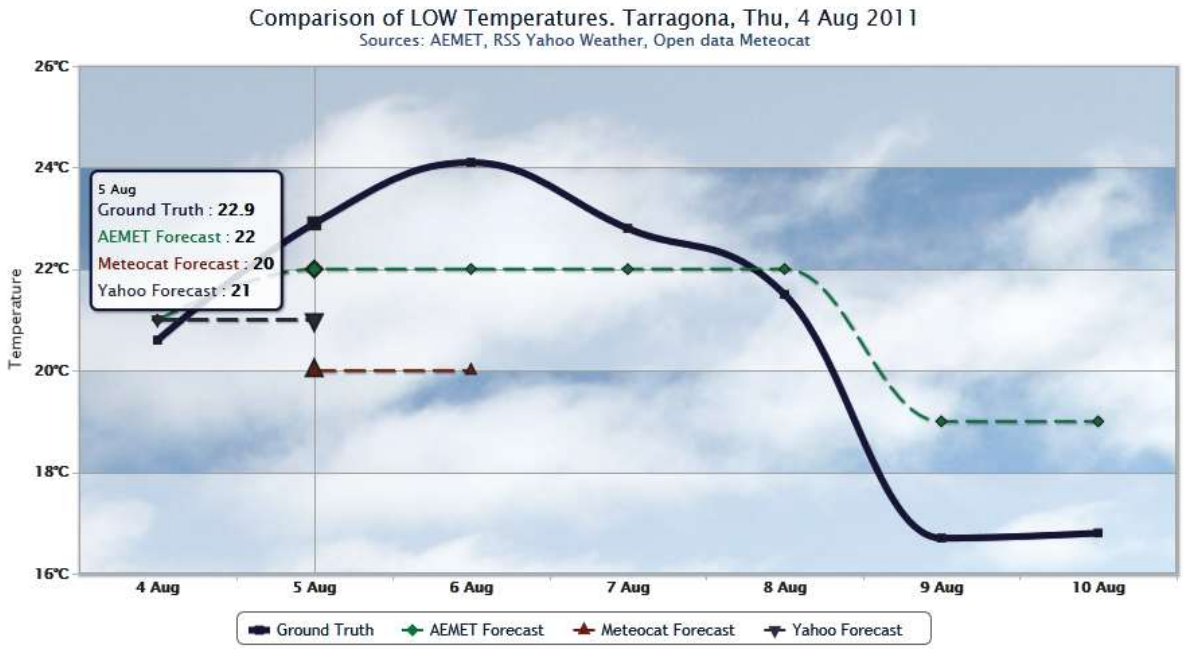

Figure 34.- Graph with low temperatures in the predictions made a specified day page.

Moreover, the user gets an assessment of each service for a specified city and day. It consists of the comparison (by means of the mean squared error and the approximation error) of the ground truth with the forecasting data. As usual, this information is shown in both ways: tables and graphs. Figure 35 shows two tables with the mean squared error and the approximation error respectively. In the tables, the first column has the name of the web service. The second and third columns demonstrate the errors of the high and low temperatures respectively. The last column shows the average error, which is an arithmetic mean between the two previous errors.

MeanSquaredError	TempHigh	TempLow	Average Error
Aemet	0,7	0,5	0,6
Yahoo	0,46	0,53	0,49
Meteocat	1,08	2,47	1,78

Approximation Error	TempHigh	TempLow	Average Error
Aemet	7,14%	6,33%	6,74%
Yahoo	2,29%	5,29%	3,79%
Meteocat	4,09%	14,89%	9,49%

Figure 35.- Errors in the predictions made in a specified day page.

If we click in the name of the error, we get the errors of the web services for the last 5 days. With the help of this information, the user discovers the forecasting service which is currently offering more accurate predictions. This information is showed in a table and a graph. The table (see Figure 36) and the graph (see Figure 37) show the average errors that web services made each day. The Y-axis of the graph represents the average mean squared error. For instance, in this example, the most reliable web service is AEMET (because it is the one with the lowest error) and the worst one is Yahoo.

Mean Squared Errors	31 Jul	1 Aug	2 Aug	3 Aug	4 Aug
Aemet	0,42	0,55	0,56	0,55	0,6
Yahoo	0,82	1,21	1,37	1,01	0,49
Meteocat	0,58	0,44	0,62	0,44	1,78

Figure 36.- Table with mean squared errors of previous 5 days.

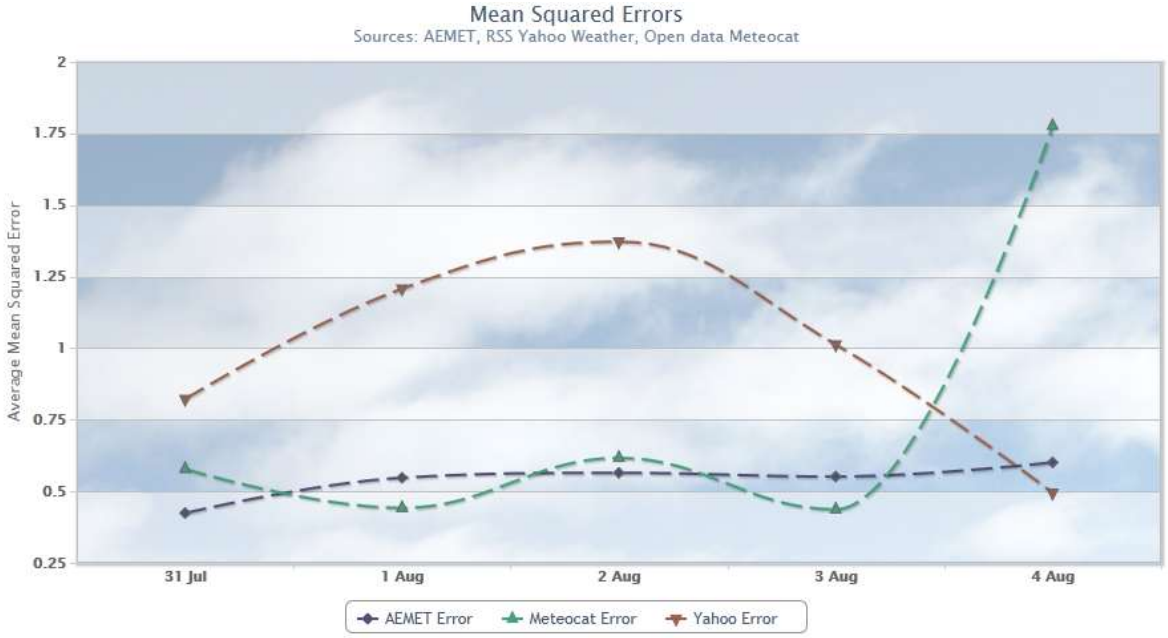

Figure 37.- Line chart with mean squared errors of previous 5 days.

3. Page of predictions made for a specified day

In the "predictions made for a specified day" page, users can consult all weather forecasts made for a specific day (the predictions for this day) and for a specific city. Firstly, the initial form needs two parameters: the date of forecast (the day we are interested to know the predictions) and the city. Figure 38 shows the form.

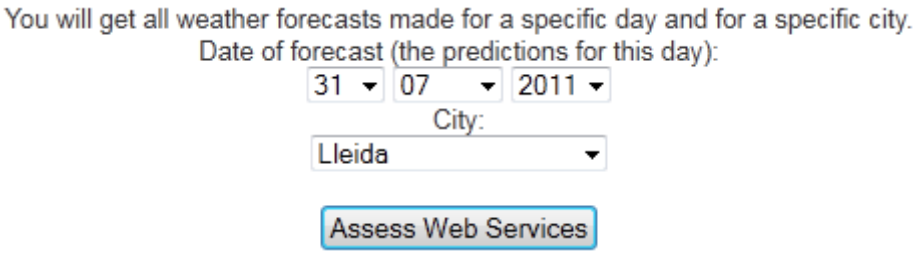

Figure 38.- Form of the page to consult predictions made for a specified day.

After filling the form, we get the information as follows. A table with all available forecasts for this day are shown in Figure 39. It has 6 columns. The first one indicates the day when the forecast was made. The second and third columns represent the real observations of the high temperature and low temperature respectively. Obviously, since all forecasts of this table have predictions for the same day, these two columns have the same value in all rows. The fourth and fifth columns contain the predicted maximum and minimum temperature respectively. For instance, the highlighted row of Figure 39 contains predictions that were taken on July 30th but predict the weather for July 31th. Finally, the last column indicates the forecasting web service that performed the forecast.

Lleida's Weather Forecasts for the day Sun, 31 Jul 2011

Date Invocation	Real TempHigh	Real TempLow	Predicted TempHigh	Predicted TempLow	Web Service
Mon, 25 Jul 2011	30.4	17.6	29.0	15.0	AemetWeatherProxy
Tue, 26 Jul 2011	30.4	17.6	30.0	17.0	AemetWeatherProxy
Wed, 27 Jul 2011	30.4	17.6	32.0	17.0	AemetWeatherProxy
Thu, 28 Jul 2011	30.4	17.6	31.0	18.0	AemetWeatherProxy
Fri, 29 Jul 2011	30.4	17.6	31.0	17.0	AemetWeatherProxy
Sat, 30 Jul 2011	30.4	17.6	31.0	17.0	AemetWeatherProxy
Sun, 31 Jul 2011	30.4	17.6	31.0	18.0	AemetWeatherProxy
Sat, 30 Jul 2011	30.4	17.6	33.0	18.0	YahooWeatherProxy
Sun, 31 Jul 2011	30.4	17.6	32.0	18.0	YahooWeatherProxy
Fri, 29 Jul 2011	30.4	17.6	33.0	18.0	MeteocatWeatherProxy
Sat, 30 Jul 2011	30.4	17.6	31.0	17.0	MeteocatWeatherProxy

Figure 39.- Table with all the forecasts that have been made for a specified day.

The same information which is contained in Figure 39, it is shown at the end of the page in graphs. There are two line charts with high and low temperatures respectively. They are shown in Figure 40 and Figure 41. The X-axis represents the date when forecasts were made and the Y-axis the temperature in Celsius. All forecasts predict the temperature of the same day. The difference is that the forecasts were made in different days. Observations of this day are plotted with rectangles and then joined with a thick continuous blue line. They are shown to be visually compared with forecasts. Forecasts are plotted with different shapes and then joined with a thin dashed line. Each of these dashed lines includes forecasts made by the same web service.

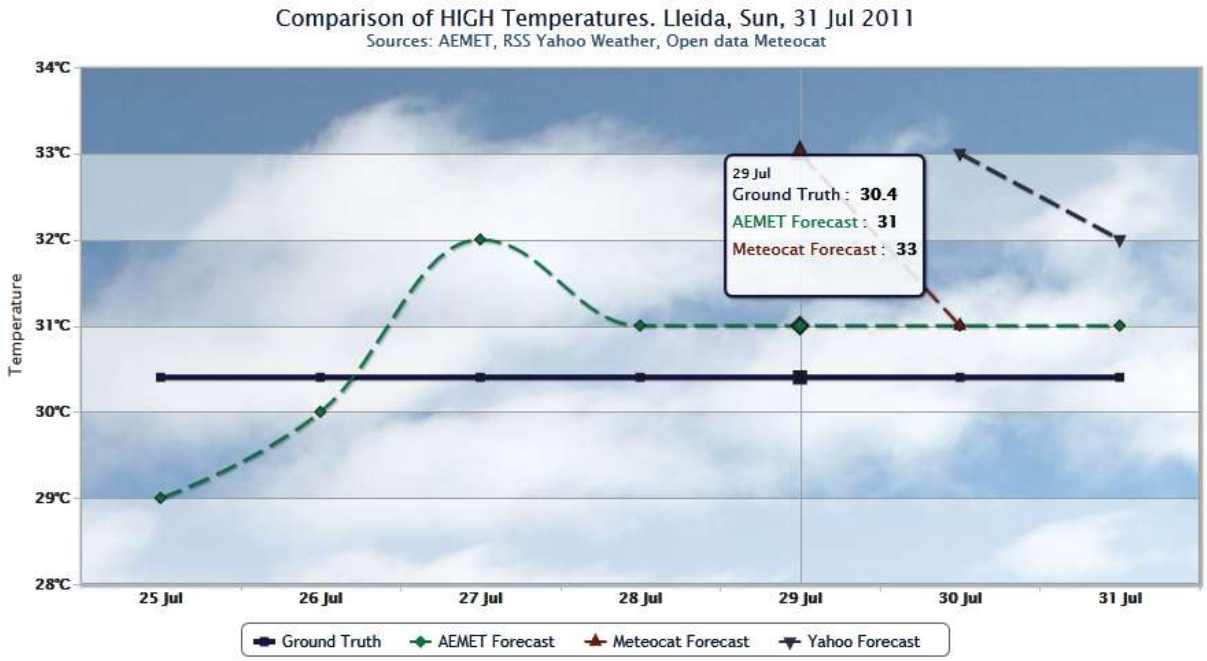

Figure 40.- Chart with high temperatures in the predictions made for specified day page.

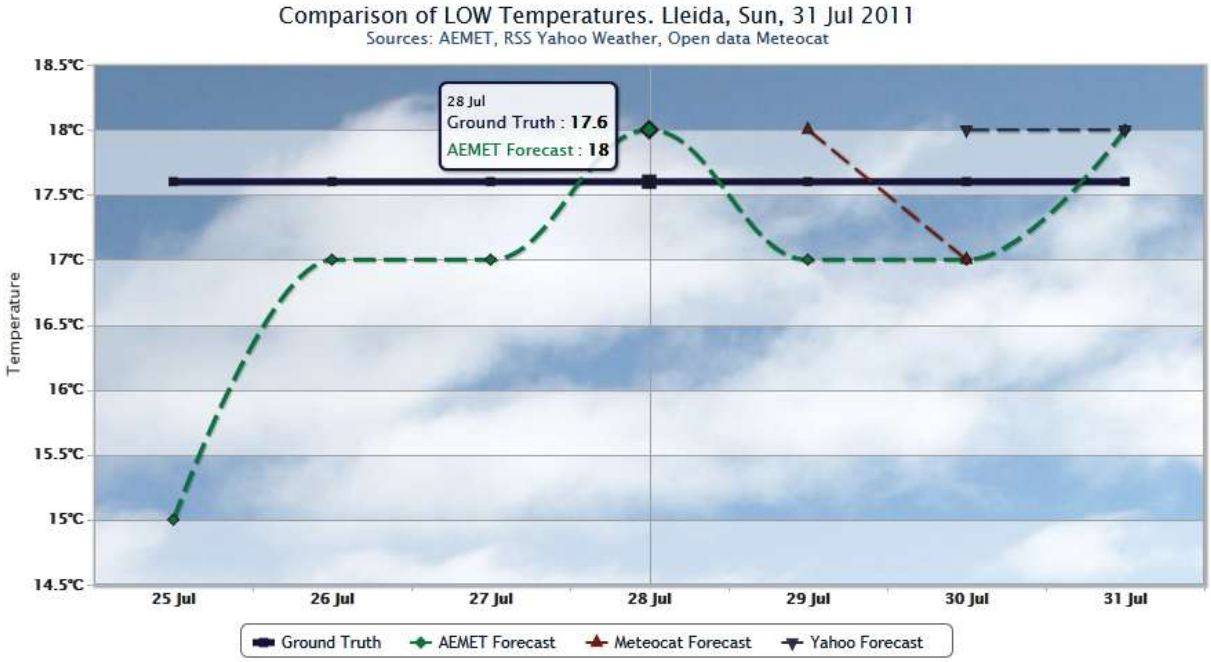

Figure 41.- Chart with low temperatures in the predictions made for specified day page.

The mean squared error (MSE) and approximation error of the forecasting web services (see Figure 42), the table (see Figure 43) and chart (Figure 44) with mean squared errors of previous 5 days are shown as it is explained in the previous subsection.

MeanSquaredError	TempHigh	TempLow	Average Error
Aemet	0,35	0,42	0,39
Yahoo	1,53	0,28	0,9
Meteocat	1,33	0,36	0,85

Approximation Error	TempHigh	TempLow	Average Error
Aemet	2,73%	4,71%	3,72%
Yahoo	6,91%	2,27%	4,59%
Meteocat	5,26%	2,84%	4,05%

Figure 42.- Errors in the predictions made in for specified day page.

Mean Squared Errors	27 Jul	28 Jul	29 Jul	30 Jul	31 Jul
Aemet	0,33	0,27	0,49	0,53	0,39
Yahoo	1,04	1,79	1,11	0,99	0,9
Meteocat	0,89	1,2	1,61	0,25	0,85

Figure 43.- Table with MSEs in the predictions made in for specified day page.

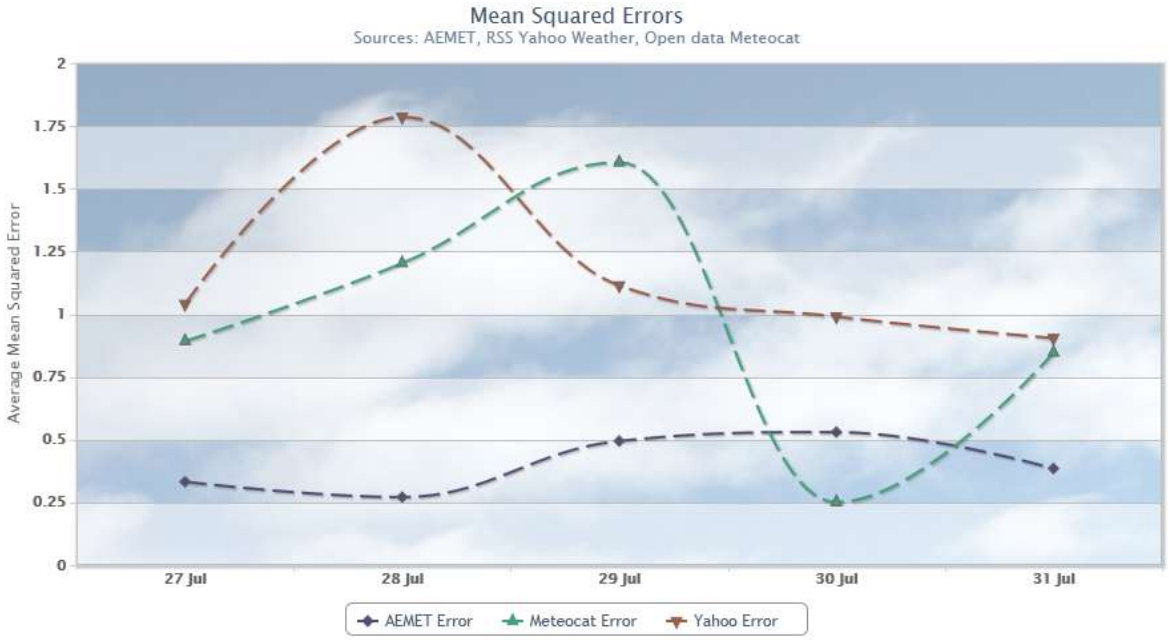

Figure 44.- Graph with MSEs in the predictions made in for specified day page.

Appendix C: Administrator manual

The administrator manages the process of collection of ground truth and forecast data with the *forecasting data collector* web service. He is also in charge of maintaining the forecasting web services that have proxies function. Web services can only be accessed from the university's network for security reasons. All of them are available on: http://gessi.lsi.upc.edu/accuracyassessment/services/listServices

The proxies, that have been made to integrate external sources which provide forecasts, include an operation to get forecasts by city code. The input parameters are explained in section 7.1.

The *forecasting data collector* web service (WSDL in Table 42) has 4 operations:

- Setup Catalan cities. They are Barcelona, Girona, Lleida and Tarragona. They are going to be monitored by 3 web services: *AemetWeatherProxy* (WSDL in Table 43), *MeteocatWeatherProxy* and *YahooWeatherProxy*. Monitoring does not start until the operation "start monitoring" is done.
- Start monitoring. It starts to monitor, via SALMon, setup cities. The default value of monitoring interval is 21600 seconds. The default value of timeout is 60 seconds.
- Start parsing ground truth. It starts to parser ground truth (observations) of the day before from AEMET's ftp. They are saved into the *ground truth database*. This operation is performed every day in a systematic manner.
- Start parsing forecast data. It starts to parser SALMon's invocations of the day before. They are saved into the *forecast data database*. This operation is performed every day in a systematic manner.

Table 42.- Forecasting data collector service's WSDL.

```
<wsdl:definitions name="ForecastingDataCollector"
      targetNamespace="http://gessi.lsi.upc.edu/accuracyassessment/Forecast
ingDataCollector/">
      <wsdl:types>
            <xsd:schema

      targetNamespace="http://gessi.lsi.upc.edu/accuracyassessment/Forecast
ingDataCollector/">
                  <xsd:element name="SetupCatalanCities">
                        <xsd:complexType>
                              <xsd:sequence />
                        </xsd:complexType>
                  </xsd:element>
                  <xsd:element name="SetupCatalanCitiesResponse">
                        <xsd:complexType>
                              <xsd:sequence>
                                    <xsd:element name="out"
type="xsd:string" />
                              </xsd:sequence>
                        </xsd:complexType>
                  </xsd:element>
                  <xsd:element name="StartMonitoring">
                        <xsd:complexType>
                              <xsd:sequence />
                        </xsd:complexType>
                  </xsd:element>
                  <xsd:element name="StartMonitoringResponse">
                        <xsd:complexType>
                              <xsd:sequence>
                                    <xsd:element name="out"
type="xsd:string" />
                              </xsd:sequence>
                        </xsd:complexType>
                  </xsd:element>
                  <xsd:element name="StartToParserInvocations">
                        <xsd:complexType>
                              <xsd:sequence />
                        </xsd:complexType>
                  </xsd:element>
                  <xsd:element name="StartToParserInvocationsResponse">
                        <xsd:complexType>
                              <xsd:sequence>
                                    <xsd:element name="out"
type="xsd:string" />
                              </xsd:sequence>
                        </xsd:complexType>
                  </xsd:element>
                  <xsd:element name="StartToParserGroundTruth">
                        <xsd:complexType>
                              <xsd:sequence />
                        </xsd:complexType>
                  </xsd:element>
                  <xsd:element name="StartToParserGroundTruthResponse">
                        <xsd:complexType>
                              <xsd:sequence>
                                    <xsd:element name="out"
```

```
type="xsd:string" />
                                    </xsd:sequence>
                              </xsd:complexType>
                        </xsd:element>
                  </xsd:schema>
            </wsdl:types>
            <wsdl:message name="StartMonitoringResponse">
                  <wsdl:part name="parameters"
element="tns:StartMonitoringResponse">
                  </wsdl:part>
            </wsdl:message>
            <wsdl:message name="SetupCatalanCitiesRequest">
                  <wsdl:part name="parameters" element="tns:SetupCatalanCities">
                  </wsdl:part>
            </wsdl:message>
            <wsdl:message name="SetupCatalanCitiesResponse">
                  <wsdl:part name="parameters"
element="tns:SetupCatalanCitiesResponse">
                  </wsdl:part>
            </wsdl:message>
            <wsdl:message name="StartToParserGroundTruthResponse">
                  <wsdl:part name="parameters"
element="tns:StartToParserGroundTruthResponse">
                  </wsdl:part>
            </wsdl:message>
            <wsdl:message name="StartToParserInvocationsRequest">
                  <wsdl:part name="parameters"
element="tns:StartToParserInvocations">
                  </wsdl:part>
            </wsdl:message>
            <wsdl:message name="StartMonitoringRequest">
                  <wsdl:part name="parameters" element="tns:StartMonitoring">
                  </wsdl:part>
            </wsdl:message>
            <wsdl:message name="StartToParserGroundTruthRequest">
                  <wsdl:part name="parameters"
element="tns:StartToParserGroundTruth">
                  </wsdl:part>
            </wsdl:message>
            <wsdl:message name="StartToParserInvocationsResponse">
                  <wsdl:part name="parameters"
element="tns:StartToParserInvocationsResponse">
                  </wsdl:part>
            </wsdl:message>
            <wsdl:portType name="ForecastingDataCollector">
                  <wsdl:operation name="SetupCatalanCities">
                        <wsdl:documentation>
                              Setup 4 catalan cities (Barcelona, Girona, Lleida
and Tarragona). They are going to be monitored by 4 web services:
WeatherBug, AemetProxy, MeteocatProxy and Yahoo Proxy. Monitoring does not
start until the operation "startMonitoring" is done.
      </wsdl:documentation>
                        <wsdl:input message="tns:SetupCatalanCitiesRequest">
                        </wsdl:input>
                        <wsdl:output message="tns:SetupCatalanCitiesResponse">
                        </wsdl:output>
                  </wsdl:operation>
                  <wsdl:operation name="StartMonitoring">
                        <wsdl:documentation>
```

```
                        Starts to Monitor, via SALMon, the cities that have
been setup. Default value of monitoring interval: 21600 seconds. Default
value of timeout: 60 seconds.
    </wsdl:documentation>
                <wsdl:input message="tns:StartMonitoringRequest">
                </wsdl:input>
                <wsdl:output message="tns:StartMonitoringResponse">
                </wsdl:output>
          </wsdl:operation>
          <wsdl:operation name="StartToParserInvocations">
                <wsdl:documentation>
                        Starts to Parser SALMon's invocations of the day
before. They are saved into the forecast data database.
    </wsdl:documentation>
                <wsdl:input
message="tns:StartToParserInvocationsRequest">
                </wsdl:input>
                <wsdl:output
message="tns:StartToParserInvocationsResponse">
                </wsdl:output>
          </wsdl:operation>
          <wsdl:operation name="StartToParserGroundTruth">
                <wsdl:documentation>
                        Starts to Parser ground truth (observations) of the
day before from AEMET's ftp. They are saved into the forecast data
database.
    </wsdl:documentation>
                <wsdl:input
message="tns:StartToParserGroundTruthRequest">
                </wsdl:input>
                <wsdl:output
message="tns:StartToParserGroundTruthResponse">
                </wsdl:output>
          </wsdl:operation>
      </wsdl:portType>
      <wsdl:binding name="ForecastingDataCollectorSOAP"
type="tns:ForecastingDataCollector">
          <soap:binding style="document"
                transport="http://schemas.xmlsoap.org/soap/http" />
          <wsdl:operation name="SetupCatalanCities">
                <soap:operation

      soapAction="http://gessi.lsi.upc.edu/accuracyassessment/ForecastingDa
taCollector/SetupCatalanCities" />
                <wsdl:input>
                      <soap:body use="literal" />
                </wsdl:input>
                <wsdl:output>
                      <soap:body use="literal" />
                </wsdl:output>
          </wsdl:operation>
          <wsdl:operation name="StartMonitoring">
                <soap:operation

      soapAction="http://gessi.lsi.upc.edu/accuracyassessment/ForecastingDa
taCollector/StartMonitoring" />
                <wsdl:input>
                      <soap:body use="literal" />
                </wsdl:input>
```

```
                  <wsdl:output>
                        <soap:body use="literal" />
                  </wsdl:output>
            </wsdl:operation>
            <wsdl:operation name="StartToParserInvocations">
                  <soap:operation

      soapAction="http://gessi.lsi.upc.edu/accuracyassessment/ForecastingDa
taCollector/StartToParserInvocations" />
                  <wsdl:input>
                        <soap:body use="literal" />
                  </wsdl:input>
                  <wsdl:output>
                        <soap:body use="literal" />
                  </wsdl:output>
            </wsdl:operation>
            <wsdl:operation name="StartToParserGroundTruth">
                  <soap:operation

      soapAction="http://gessi.lsi.upc.edu/accuracyassessment/ForecastingDa
taCollector/StartToParserGroundTruth" />
                  <wsdl:input>
                        <soap:body use="literal" />
                  </wsdl:input>
                  <wsdl:output>
                        <soap:body use="literal" />
                  </wsdl:output>
            </wsdl:operation>
      </wsdl:binding>
      <wsdl:service name="ForecastingDataCollector">
            <wsdl:port name="ForecastingDataCollectorSOAP"
binding="tns:ForecastingDataCollectorSOAP">
                  <soap:address
location="http://gessi.lsi.upc.edu/accuracyassessment/" />
            </wsdl:port>
      </wsdl:service>
</wsdl:definitions>
```

Table 43.- AEMET weather proxy service's WSDL.

```
<wsdl:definitions name="AemetWeatherProxy"
      targetNamespace="http://gessi.lsi.upc.edu/accuracyassessment/AemetWea
therProxy/">
      <wsdl:types>
            <xsd:schema

      targetNamespace="http://gessi.lsi.upc.edu/accuracyassessment/AemetWea
therProxy/">
                  <xsd:complexType name="ArrayOfApiForecastData">
                        <xsd:sequence>
                              <xsd:element maxOccurs="unbounded"
minOccurs="0" name="anyType"
                                    nillable="true"
type="tns:ApiForecastData" />
```

```
                        </xsd:sequence>
                  </xsd:complexType>
                  <xsd:complexType name="ApiForecastData">
                        <xsd:sequence>
                              <xsd:element maxOccurs="1" minOccurs="0"
name="ConditionID"
                                    type="xsd:string" />
                              <xsd:element maxOccurs="1" minOccurs="0"
name="Description"
                                    type="xsd:string" />
                              <xsd:element maxOccurs="1" minOccurs="0"
name="Icon"
                                    type="xsd:string" />
                              <xsd:element maxOccurs="1" minOccurs="0"
name="Image"
                                    type="xsd:string" />
                              <xsd:element maxOccurs="1" minOccurs="0"
name="IsNight"
                                    type="xsd:boolean" />
                              <xsd:element maxOccurs="1" minOccurs="0"
name="Prediction"
                                    type="xsd:string" />
                              <xsd:element maxOccurs="1" minOccurs="0"
name="ShortPrediction"
                                    type="xsd:string" />
                              <xsd:element maxOccurs="1" minOccurs="0"
name="ShortTitle"
                                    type="xsd:string" />
                              <xsd:element maxOccurs="1" minOccurs="0"
name="TempHigh"
                                    type="xsd:string" />
                              <xsd:element maxOccurs="1" minOccurs="0"
name="TempLow"
                                    type="xsd:string" />
                              <xsd:element maxOccurs="1" minOccurs="0"
name="TempUnit"
                                    type="xsd:string" />
                              <xsd:element maxOccurs="1" minOccurs="0"
name="Title"
                                    type="xsd:string" />
                              <xsd:element maxOccurs="1" minOccurs="0"
name="WebUrl"
                                    type="xsd:string" />
                        </xsd:sequence>
                  </xsd:complexType>
                  <xsd:element name="GetAemetForecastByCityCode">
                        <xsd:complexType>
                              <xsd:sequence>
                                    <xsd:element maxOccurs="1" minOccurs="1"
name="cityCode"
                                          type="xsd:string" />
                              </xsd:sequence>
                        </xsd:complexType>
                  </xsd:element>
                  <xsd:element name="GetAemetForecastByCityCodeResponse">
                        <xsd:complexType>
                              <xsd:sequence>
                                    <xsd:element maxOccurs="1" minOccurs="0"
```

```
      name="GetAemetForecastByCityCodeResult"
type="tns:ArrayOfApiForecastData" />
                              </xsd:sequence>
                        </xsd:complexType>
                  </xsd:element>
            </xsd:schema>
      </wsdl:types>
      <wsdl:message name="GetAemetForecastByCityCodeResponse">
            <wsdl:part name="parameters"
element="tns:GetAemetForecastByCityCodeResponse">
            </wsdl:part>
      </wsdl:message>
      <wsdl:message name="GetAemetForecastByCityCodeRequest">
            <wsdl:part name="parameters"
element="tns:GetAemetForecastByCityCode">
            </wsdl:part>
      </wsdl:message>
      <wsdl:portType name="AemetWeatherProxy">
            <wsdl:operation name="GetAemetForecastByCityCode">
                  <wsdl:documentation>Get forecast based
                        on citycode from Aemet
                        Opendata.</wsdl:documentation>
                  <wsdl:input
message="tns:GetAemetForecastByCityCodeRequest">
                  </wsdl:input>
                  <wsdl:output
message="tns:GetAemetForecastByCityCodeResponse">
                  </wsdl:output>
            </wsdl:operation>
      </wsdl:portType>
      <wsdl:binding name="AemetWeatherProxySOAP"
type="tns:AemetWeatherProxy">
            <soap:binding style="document"
                  transport="http://schemas.xmlsoap.org/soap/http" />
            <wsdl:operation name="GetAemetForecastByCityCode">
                  <soap:operation

      soapAction="http://localhost:8080/AemetWeatherProxy/GetAemetForecastB
yCityCode"
                        style="document" />
                  <wsdl:input>
                        <soap:body use="literal" />
                  </wsdl:input>
                  <wsdl:output>
                        <soap:body use="literal" />
                  </wsdl:output>
            </wsdl:operation>
      </wsdl:binding>
      <wsdl:service name="AemetWeatherProxy">
            <wsdl:port name="AemetWeatherProxySOAP"
binding="tns:AemetWeatherProxySOAP">
                  <soap:address
location="http://gessi.lsi.upc.edu/accuracyassessment/" />
            </wsdl:port>
      </wsdl:service>
</wsdl:definitions>
```