Verifying Predictive Services' Quality with Mercury

S. Martínez-Fernández[a,*], X. Franch[a], J. Bisbal[b]

[a]*GESSI Research Group, Universitat Politècnica de Catalunya - Barcelona Tech (UPC), C/Jordi Girona 1-3, 08034, Barcelona, Spain*
[b]*Dept. ICT, Universitat Pompeu Fabra (UPF), C/Tánger 122-140, 08018, Barcelona, Spain*

Abstract

Due to the success of service technology, there are lots of services nowadays that make predictions about the future in domains such as weather forecast, stock market and bookmakers. The value delivered by these predictive services relies on the quality of their predictions. This paper presents Mercury, a tool that measures predictive service quality in the domain of weather forecast, and automates the context-dependent selection of the most accurate predictive service to satisfy a customer query. To do so, candidate predictive services are monitored so that their predictions can be eventually compared with real observations obtained from some trusted source. Mercury is a proof-of-concept to show that the selection of predictive services can be driven by the quality of their predictions. Its service-oriented architecture (SOA) aims to support the easy adaptation to other prediction domains and makes feasible its integration in self-adaptive SOA systems, as well as its direct use by end-users as a classical web application. Thoughout the paper, we show how Mercury was built.

Keywords: tool development, predictive services, service-oriented architecture, service selection, service monitoring, forecast verification

1. Introduction

The success of service technologies has boosted a big offer of services covering many domains. Service customers do not need to worry about the development, maintenance, infrastructure, or any other issue related to the service operation. Instead, they just have to find and choose the most appropriate service offered by some service provider [1].

Therefore, it becomes necessary to assess which service is the most appropriate for fulfilling the customer's needs. These needs may refer to quality of service (QoS), reputation, cost, security, personalisation, locality and so on.

Among all kinds of services, we concentrate on predictive (or forecasting) services. We define *predictive services* as those services whose main functionality is to show in advance a condition or occurrence about the future. Predictive services emerge in many domains: stock

*Corresponding author
Email addresses: smartinez@essi.upc.edu (S. Martínez-Fernández), franch@essi.upc.edu (X. Franch), jesus.bisbal@upf.edu (J. Bisbal)
URL: http://www.essi.upc.edu/~smartinez/ (S. Martínez-Fernández)

market prices, bookmaker results, election polls, sales forecasting and so on. We are quite used to see this functionality offered by websites that provide predictions over specific data [2]. The quality of their predictions is of obvious importance to: citizens because trustable predictions may significantly improve their decision-making; providers because it affects their reputation, and potentially their revenues; and developers, if the prediction algorithm is embodied as a software component ready to be integrated into other systems (e.g., offered as a Web Service (WS)).

An example that is really familiar to all of us is weather forecast. Weather conditions affect our decisions in daily routines such as deciding what to wear first thing in the morning or when planning a trip. To make these decisions, different services like the weather forecast section on TV news or specialized web sites (such as forecastadvisor.com) are consulted. However, sometimes their predictions do not match or they change over time as the date of interest approaches and therefore a software engineering challenge arises: given a portfolio of candidate predictive services, which one is expected to be the most accurate to satisfy the customer needs?

This paper presents *Mercury*, a tool that assesses weather predictive services based on the quality of their predictions. Mercury is a particular instance of *QuPreSS*, a service-oriented reference model for predictive service quality assessment [3]. Mercury is a tangible result from QuPreSS and was built as a proof-of-concept for the challenge of assessing weather predictive services. Thoughout the paper, we show how Mercury was built, focusing on its requirements and the main design decisions.

The preliminary validation that we offer in the paper is a first prototype that is focused on the weather forecast domain. It is the first step of the creation of a general validation framework for the problem of prediction quality in many domains.

The paper is structured as follows. In Section 2, we motivate the problem we want to solve. Section 3 describes the requirements as well as the main design and implementation decisions made on Mercury. The demonstration scenarios are illustrated in Section 4. Section 5 shows lessons learned from the experience of Mercury building. The final section summarizes the paper and identifies a number of future directions.

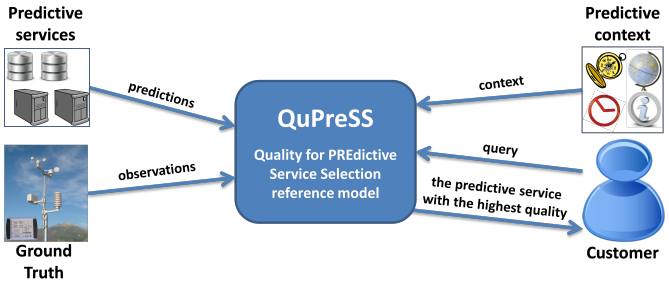
2. The Prediction Problem

Forecast verification is the process of assessing the quality of a prediction by comparing it with its corresponding observation [4]. Forecast verification for predictive services has not received sufficient attention; just few previous works have used prediction quality to guide the selection of services for the weather forecast domain. For instance, Domenico [5] propose geosciences WS to integrate sources of data in order to perform later forecast verification with observations. Therefore, more effort is necessary to integrate methods for forecast verification in SOA monitoring frameworks [3].

In response to this scenario, we have presented elsewhere the QuPreSS reference model [3]. QuPreSS addresses the selection of the predictive service that is expected to be the most accurate to satisfy some given customer needs. To this end, QuPreSS requires the following four inputs (see Fig. 1):

- *Predictive services*. The portfolio of services that offer predictions to the customer. These services need to be localized in a service directory.
- *Ground truth*. Trusted information that is the object of prediction. It is collected from one single source (e.g., service providing real observations once they happen), which is trusted and reliable, hence only one is needed.

Figure 1: The QuPreSS reference model.



- *Predictive context*. Context conditions that may influence the predictions (e.g., date and location).
- *Customer query*. The concrete customer's need that is required to be satisfied by a prediction given for some predictive service.

To get the two former inputs, it is necessary to use a monitoring infrastructure able to capture data from services. The two latter ones come from the customer, indirectly (information about the context gathered, e.g., from her GPS location) or directly (text of the query).

This comprehensive reference model may be implemented in all the prediction domains that we have found.

3. Mercury: A Proof-of-Concept for the Prediction Problem solved by QuPreSS

Mercury is an implementation of the QuPreSS reference model for the weather forecast domain. Therefore, its main functional requirement is to satisfy a customer's weather forecast query by selecting the most appropriate service given the context of the customer and the text of the query. Extensively, Table 1 shows Mercury requirements. Non functional requirements of Table 1 are already addressed by the QuPreSS reference model, which is depicted in Fig. 2. In this section, we present how the tool was built to accomplish the functional requirements of Table 1.

3.1. External sources (functional requirements (FR) 2, 2.1, 3, 3.1 of Table 1)

Mercury needs to have access to a ground truth service and a portfolio of predictive services. They are external and potentially heterogeneous. They can be implemented by WS or other technologies. If an external source is not exposed as a WS, it can be wrapped into a WS proxy. These proxies also take care of parsing the different formats into a unified form. To integrate all these different technological styles, Mercury is organized around an SOA.

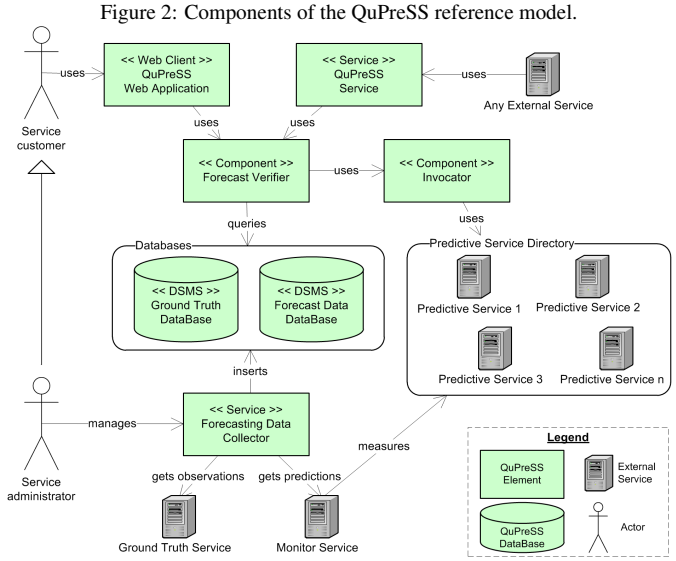
Mercury deals with a simplified scenario in terms of service portfolio and context variables. On the one hand, the ground truth is obtained from the Spanish Meteorological Agency (AEMET[1]) . The agency provides timely observations with the help of more than 700 stations with sensors measuring weather conditions. On the other hand, for demonstration purposes,

[1] ftp://ftpdatos.aemet.es/datos_observacion/resumenes_diarios/

Table 1: Summary of Mercury requirements. Detailed requirements and fit criteria can be found in [6].

Number	Requirement
FR 1	The tool shall compare the predictions from predictive services with real observations.
FR 1.1	The tool shall make a ranking of a set of weather predictive services based on the accuracy of their predictions.
FR 2	The tool shall read prediction data from several weather predictive services.
FR 2.1	The weather predictive services currently considered are: RSS Yahoo! Weather, Meteocat, and AEMET.
FR 3	The tool shall read real observations coming from a trusted source.
FR 3.1	The trusted source (i.e., ground truth) currently considered is: AEMET.
FR 4	The tool shall monitor and parser data from both types of external sources: predictive services and ground truth service.
FR 5	The tool shall save data from both types of external sources: predictive services and ground truth service.
FR 6	The tool shall be able to offer data to external systems.
FR 7	The tool shall give current weather forecast from the most accurate predictive service for a specified city.
NFR 1	The tool shall be extensible.
NFR 2	The tool shall be developed as a service.
NFR 3	The tool should be adhered to standards.
NFR 4	The tool shall be able to work with continuous data flows.
NFR 5	The tool shall use existing components when possible.
NFR 5.1	The tool shall be able to connect to SALMon to monitor web services.
NFR 6	The tool shall obey legal statements from external web services.
NFR 7	The tool shall work when external sources are unavailable.

Figure 2: Components of the QuPreSS reference model.



the portfolio of predictive services is composed of: AEMET itself[2], Meteocat[3], and Yahoo! Weather[4]. These services are continuously monitored by Mercury in order to be able to infer in which contexts they are more adequate by comparing their predictions with the ground truth over time (see Section 3.2).

The three predictive services are wrapped into a WS with a pre-defined format that consists of an array of elements of type *ApiForecastData*. This type includes a superset of elements (which can be null) with information about a weather forecast for a date: *ConditionID*, *Description*, *Icon*, *Image*, *IsNight*, *Prediction*, *ShortPrediction*, *ShortTitle*, *TempHigh*, *TempLow*, *TempUnit*, *Title* and *WebUrl*. For more information about these elements, the reader is referred to [6].

3.2. The Monitor and the Forecasting Data Collector services (FR 4)

In the heart of the architecture lies a monitor. It saves in a systematic manner the QoS of each predictive service and the response given to every periodical request launched to these services. We use the SALMon monitor infrastructure to implement this service [7]. This infrastructure was chosen given its adaptability and performance exhibited in previous uses [8][9][10]. The data collection process works as follows.

1. *Setup:* the parameters of the system are initialized. For instance, which are the cities put under the control of the monitor and which is the portfolio of predictive services considered. Also, the data collection process is initiated.
2. *Ground truth collection:* the Forecasting Data Collector service daily saves a summary of the real observations (e.g., high and low temperatures of the day) once they happen for each city. Observations are provided by AEMET.
3. *Prediction collection:* First, SALMon gets the response (with predictions) that every predictive service gives for each city. This operation is repeated several times per day to cope with temporal unavailability (every 6 hours by default). At the end of the day, the Forecasting Data Collector service gets the last response that SALMon obtained for each predictive service. The data provided by the monitor is treated for several reasons, e.g., predictive services may differ in the number of days in advance that the provide forecasts. Once this treatment finishes, the prediction of the day is stored in the databases.

3.3. Databases (FR 5)

Mercury includes two database management systems (DBMS): one in charge of saving observations (Ground Truth Database) and another one to save predictions (Forecast Data Database). In those prediction domains in which predictions change very frequently, it would be better to use data stream management systems instead of DBMS to process continuous data flows. Nevertheless, this is not the case for weather forecasting, where oscillations are seldom dramatic.

To determine the data model of our tool, we consolidated the information given by the considered sources (i.e., all three predictive services and the ground truth service). These sources give different information about weather conditions (e.g., wind speed, humidity...). Still, a subset of information is common to all of them: maximum temperature, minimum temperature, unit of temperature, date of forecast/observation and location of forecast/observation.

[2] http://www.aemet.es/es/eltiempo/prediccion/municipios
[3] http://dadesobertes.gencat.cat/ca/dades-obertes/prediccions.html
[4] http://developer.yahoo.com/weather

Figure 3: Data conceptual model for weather forecast.

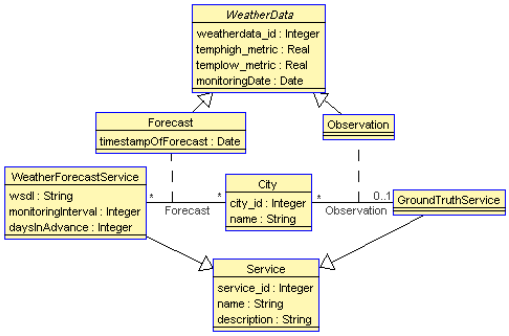


Fig. 3 shows the conceptual model describing the data collected. The two main entities are *Forecast* and *Observation*. Since both of them share most of the attributes, we define these common attributes in an abstract entity *WeatherData*.

Observations are data points given by the *GroundTruthService*, whilst Forecasts are data points that a *WeatherForecastService* makes in advance. In both cases, they refer to a particular location (*City*).

The temporal dimension plays a fundamental role in the model. We have three temporal variables to reconcile: 1) the number of days predicted by a predictive service (attribute *daysInAdvance* in *WeatherForecastService*), 2) the date in which a prediction was made (*timeStampOfForecast* in *Forecast*), and 3) the date of the prediction (*monitoringDate* in *WeatherData*).

3.4. Forecast Verifier (FR 1, 1.1, 7)

The Forecast Verifier component handles customer queries. It has two functionalities. First, it supports the decision-making process of choosing the most accurate predictive service from the available portfolio. Second, it handles the query over this chosen service. For this second functionality, it relies on the Invocator component, which invokes the chosen service to return its predictions to the customer (see Fig. 2).

The quality parameter defined in this proof-of-concept was the accuracy of high and low temperature forecasts. More precisely, we have implemented two measures, the mean-squared error and the approximation error. Both of them quantify the difference between values implied by an estimator (i.e., predictive services) and the truth values.

The mean-squared error (MSE) of a predictive service PS is defined as:

$$MSE_{PS} = \frac{\sqrt{\sum_{i=1}^{m}(\mu_{Tgt} - T_{PSi})^2}}{n} \quad (1)$$

where μ_{Tgt} is the average of temperatures (high or low) of the corresponding ground truth observations ("corresponding" means the observations that have the same value for the *monitoringDate* attribute as the *monitoringDate* attribute of the predictions of T_{PSi}); T_{PSi} is a temperature (high or low) as predicted by the PS; i is an index (which refers to a prediction made for a date interval); n is the total amount of observations from the gt (ground truth) and m the total amount

of predictions made by the *PS* for a date interval. When the *PS* gives predictions more than one day in advance, m is greater than n because there are several predictions for the same date.

The approximation error (AE) is calculated as follows:

$$AE_{PS} = \frac{\sum_{i=1}^{n} |T_{GTi} - T_{PSi}|}{\sum_{i=1}^{n} |T_{GTi}|} \quad (2)$$

where T indicates a high or low temperature that is either a prediction from the *PS* or its corresponding observation from the ground truth (GT); i is an index (which refers to a date); and n is the total amount of observations/predictions compared.

A quality-based ranking of predictive services can be done by using any of these two measures. The default ranking is made with the mean-squared error of predictions for the last fifteen days, in which predictive services with lower mean-squared error are more accurate. This allows to select the predictive service with the highest likelihood to be right. This "best" service is the one with the highest quality in the past under the variables (e.g., a city) of a customer query.

3.5. Front-ends (FR 6)

Mercury provides two different front-ends. First, a web client application that can be used by customers to directly enter their queries. The parameters of the query are the date and the location in which the customer is interested, and the kind of response (report or redirection). Second, a WS that allows external services interoperating with Mercury via a WSDL definition. Section 4 provides more details on this part.

3.6. Technologies

Mercury has been implemented in Java. Mercury services run under the WS engine Axis2 and the Tomcat application server. MySQL has been chosen as DBMS. The web client uses jQuery and Highcharts javascript libraries for graphics.

4. Demonstration Scenarios

We have designed two demonstration scenarios to show the functionalities and usage of the web client interface and WS. They are explained below and in the demonstration video accesible at the following URL: `http://youtu.be/XE58jSwcIic` .

4.1. Analysis Scenario

To analyze under which circumstances prediction models perform better or worse, Mercury can be requested to generate a report with the ranking of predictive services ordered by their mean-squared error or approximation error. This ranking is generated given a specified city and a period of time. There are two kinds of reports:

1. *Report with predictions made in a specific date:* It analyses all weather forecasts made in the chosen date for a specific city. For instance, if the customer asks which service made better forecasts on August 4th 2011 in Tarragona, she gets the mean-squared errors of predictions made by all predictive services when forecasting next days' weather (5th, 6th...). As we can see in Fig. A.13 (see Appendix A), the service that made better forecasts was Yahoo.

2. *Report with predictions made `for` a specific date:* It analyzes all weather forecasts made for the given date and a specific city. For example, if the customer asks which service gave better forecasts for July 31st 2011 in Lleida, she gets mean-squared errors of predictions previously made (on 30th, 29th...) by all services for July 31st. The best service was AEMET (Fig. A.20, see Appendix A).

The demonstration video shows the graphical representation of the results as displayed by Mercury.

4.2. Prediction Scenario

In this scenario, the customer specifies a city and she gets the forecasts (with the highest likelihood to be right) for the next days. To do so, Mercury compares the errors that predictive services made in the predictions for that city in the last fifteen days, and returns the current forecasts from the predictive service that has been more accurate for that city in the past.

This functionality is not only available in the web client interface, but also as a WS. Fig. 4 shows an excerpt of the response given by the WS when asking for forecasts in Barcelona at November 2nd in 2012.

5. Discussion and Lessons Learned

Next, we show the difficulties or problems that we had to overcome, and then the strengths and weaknesses of the project.

The requirements basically required to integrate and monitor various web services, and to progressively add new predictive services and other ways to measure their quality. As a result, we focused on developing a tool with a scalable architecture based on SOA whereas we simplified the predictive service portfolio and context variables. We found two mayor problems during the development of the tool. First, we needed to perform adaptations to the current version of

Figure 4: Request (above) and response (below) of the ForecastVerifierWS.

```
- <soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
    xmlns:q0="http://gessi.lsi.upc.edu/accuracyassessment/ForecastVerifierWS"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  - <soapenv:Body>
    - <q0:GetTheBestForecastService>
        <city>Barcelona</city>
        <date1>2012-11-02</date1>
        <date2>2012-11-08</date2>
        <unitType>c</unitType>
      </q0:GetTheBestForecastService>
    </soapenv:Body>
  </soapenv:Envelope>
```

```
SOAP Response Envelope:
- <soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/">
  - <soapenv:Body>
    - <ns8:GetTheBestForecastServiceResponse
        xmlns:ns8="http://gessi.lsi.upc.edu/accuracyassessment/ForecastVerifierWS">
      - <anyType>
          <IsNight>false</IsNight>
          <ShortTitle>FRI</ShortTitle>
          <TempHigh>19</TempHigh>
          <TempLow>12</TempLow>
        </anyType>
      - <anyType>
          <IsNight>false</IsNight>
          <ShortTitle>SAT</ShortTitle>
          <TempHigh>20</TempHigh>
```

SALMon, namely: adding the functionality of saving the whole response of the service which is monitored; and increasing the longitude of a varchar field to support longer soap actions. Second, in the beginning of the project a fourth weather predictive service had been monitored, but after two months it became a pay-per-use service, so we had to stop to monitor it. Also, the ground truth service has been evolved by the provider, what has implied new changes. We could have avoided this by only using web services with long term support.

Regarding testing, we started the process at the lowest level: unit testing of predictive services that we developed and our web application. Later, we continued the testing process at the integration level. With the integration level we tested the working of the predictive data collector service. Finally, non-functional properties were tested. The most crucial non-functional requirement was reliability. External sources were distributed over the network and even developed and hosted by different organizations. Thus, we needed to test what happen when an external source is unavailable (for reasons out of our control). For details about how testing was performed, the reader is referred to [6].

By far, we had the biggest problem while uploading the services to the production environment. First, the network of the university had the port 500 closed, which we were using to read observations from AEMET ftp server. To solve this problem, we opened this port and made a cron job to download the observations. Second, in the development environment, 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. Therefore, we needed to generate them again with the version 1.3 to solve this problem. Third, in order to make debug and testing easier, we asked for permission to access the log of Tomcat server.

The rest of the problems were related to the need of having an expert in the predictive domain for the development of these tools, since it is needed to perfectly know the complex predictive context to assess the quality of predictions. For instance, non-expert stakeholders may have difficulties to properly cope with trivial problems (e.g., if an observation says that it rained just a few drops, can we consider as correct a forecast that predicted the rain?) or to design more complete validation plans.

5.1. Strengths and weaknesses

Mercury has allowed to assess the feasibility of the QuPreSS reference model and to understand the complexity of the prediction problem. We experienced the successful selection of predictive services by instantiating QuPreSS for the weather forecast domain. Moreover, QuPreSS may be applied for all the prediction domains that we have found.

Another strength of this project is that in our research group there were already people with experience in SOA. As a result, Mercury development has benefited from the already defined SOA infrastructure implanted for SALMon, and knowledge about standards and de-facto technologies that had been gathered in previous experiences. It contributed to efficiently build Mercury and reduce the learning curve in the beginning.

On the other hand, among the weaknesses we should remark the following ones: the tool was developed as a part of a master thesis with a limited duration; we are experiencing difficulties to find more students to follow-up this work; and finally, as an academic tool, there are limited resources (e.g., we implemented this first proof-of-concept for the weather forecast domain because it was the predictive domain for which more free services were available).

6. Conclusions

We have presented Mercury, an implementation for the weather forecast domain of the QuPreSS reference model. QuPreSS addresses the challenge of selecting the most accurate predictive service from a given portfolio to satisfy the customer's needs in a certain domain. The idea is that there are predictive services all across the web and tools become neccesary to wire them up. Mercury enables to spawn a set of predictive services. Its main contribution is to make the interconnection of such predictive services possible in order to assess their predictions.

There are three kinds of envisioned users for any instantiation of QuPreSS such as Mercury: individual citizens who want to obtain the best possible prediction for a particular query given their current context; service designers who can integrate the tool in the core of self-adaptive SOA systems to guide its evolution; domain specialists (e.g., meteorologists, brokers, etc.) who want to understand when their prediction models behave better or worse. The current implementation of Mercury has been validated with three real weather predictive services. The main goal of the validation has been to assess the feasibility of the approach and to understand its complexity for designing more complete validation plans.

As future work, a larger-scale validation over this proof-of-concept would require an ontological analysis of the domain of interest to identify the relevant domain concepts and their relationships. Also, we plan to use data-mining to identify the current knowledge about key parameters that determine predictive service quality, and to construct new Mercury-like instantiations of QuPreSS for other domains.

Acknowledgments

This work has been supported by the Spanish project TIN2010-19130-C02-00.

References

[1] M. Papazoglou, Web services: principles and technology, Addison-Wesley, 2008.

[2] K. Nikolopoulos, K. Metaxiotis, V. Assimakopoulos, E. Tavanidou, A first approach to e-forecasting: a survey of forecasting web services, Information management & computer security 11 (2003) 146–152.

[3] S. Martínez-Fernández, J. Bisbal, X. Franch, Qupress: A service-oriented framework for predictive services quality assessment, in: 7th International Conference on Knowledge Management in Organizations: Service and Cloud Computing, Springer, pp. 525–536.

[4] A. H. Murphy, What is a good forecast? an essay on the nature of goodness in weather forecasting, Weather and Forecasting 8 (1993) 281–293.

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

[6] S. Martínez-Fernández, Accuracy assessment of forecasting services (2011).

[7] M. Oriol Hilari, J. Marco Gómez, J. Franch Gutiérrez, D. Ameller, et al., Monitoring adaptable soa systems using salmon (2010).

[8] C. Muller, M. Oriol, M. Rodríguez, X. Franch, J. Marco, M. Resinas, A. Ruiz-Cortes, Salmonada: A platform for monitoring and explaining violations of ws-agreement-compliant documents, in: Principles of Engineering Service Oriented Systems (PESOS), 2012 ICSE Workshop on, IEEE, pp. 43–49.

[9] O. Sammodi, A. Metzger, X. Franch, M. Oriol, J. Marco, K. Pohl, Usage-based online testing for proactive adaptation of service-based applications, in: Computer Software and Applications Conference (COMPSAC), 2011 IEEE 35th Annual, IEEE, pp. 582–587.

[10] A. Kertész, G. Kecskeméti, A. Marosi, M. Oriol, X. Franch, J. Marco, Integrated monitoring approach for seamless service provisioning in federated clouds, in: Parallel, Distributed and Network-Based Processing (PDP), 2012 20th Euromicro International Conference on, IEEE, pp. 567–574.

Appendix A. User Manual and Demo

The web client of the tool, Mercury, is a web application that enables users to analyse the best service for their needs. The user can consult: observations, predictions and its errors. It is available on `http://gessi.lsi.upc.edu/accuracyassessment/` . Below, we show the features of the analysis scenario (see Section 4.1). The demonstration video, available at `http://youtu.be/XE58jSwcIic`, includes a summary of both scenarios (analysis and prediction).

The index page is shown in Figure A.5. 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.

The following subsections illustrate how the different pages work.

Appendix A.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 A.6 has as input parameters the number of days and the city to be consulted.

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 A.7) and with a graph (Figure A.8). 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.

Figure A.5: Main screen of the forecast verifier application.

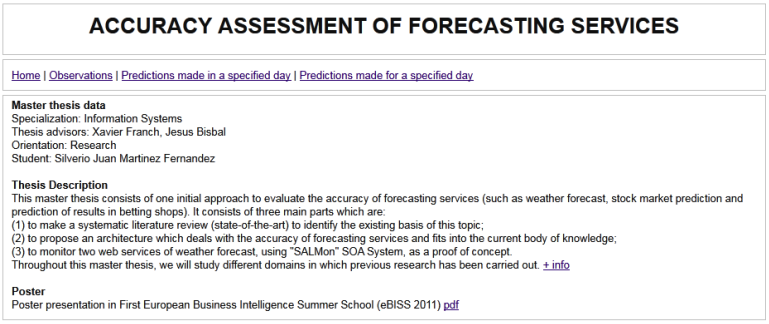


Figure A.6: Form of observations menu.

See observations of x last days:

7

City:

Girona

See Observations (taken by AEMET's sensors)

Figure A.7: Example of a table with observations.

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 A.8: Example of a graph with observations.

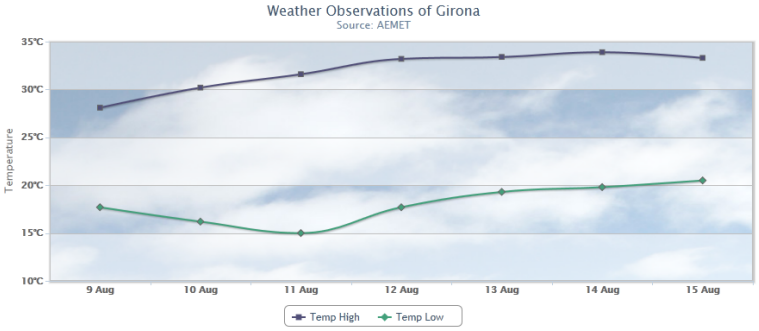


Figure A.9: Form of the predictions made in a specified day page.

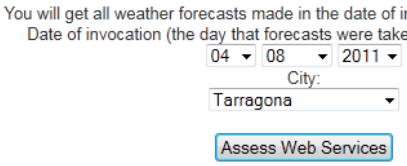


Figure A.10: Example of a table with predictions made a specified day.

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

Appendix A.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.

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 A.10). 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 A.10 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.

The same information appears at the end of the page in graphs. There are two line charts with high (see Figure A.11) and low temperatures (see Figure A.12) 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.

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 A.13 shows two tables with the mean squared error and the approximation

Figure A.11: Graph with high temperatures in the predictions made a specified day page.

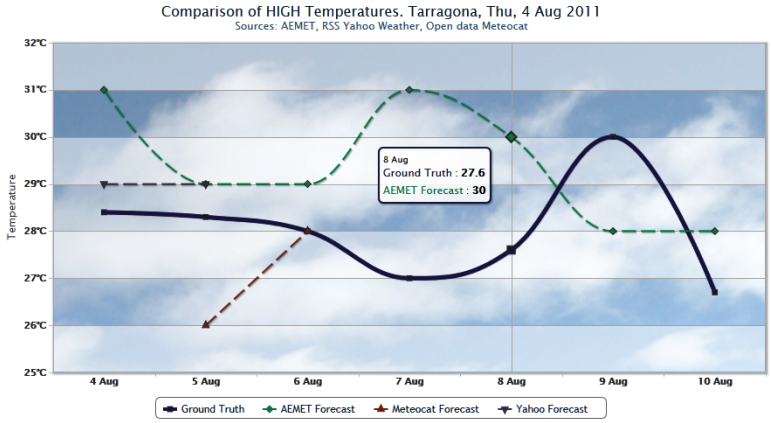


Figure A.12: Graph with low temperatures in the predictions made a specified day page.

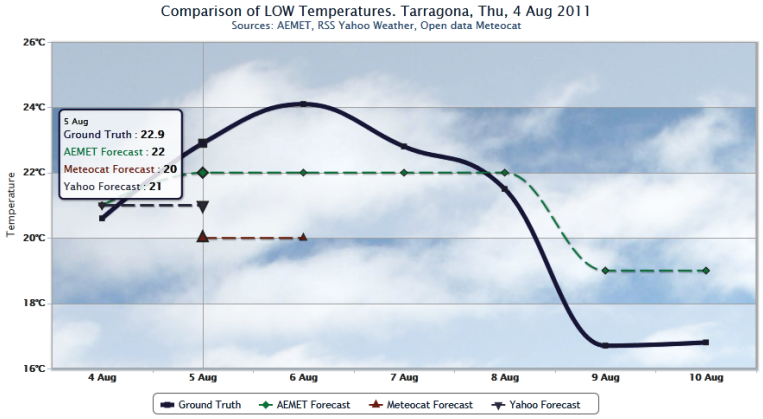


Figure A.13: Errors in the predictions made in a specified day page.

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 A.14: Table with mean squared errors of previous 5 days.

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

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.

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 A.14) and the graph (see Figure A.15) 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.

Appendix A.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 A.16 shows the form.

After filling the form, we get the information as follows. A table with all available forecasts for this day are shown in Figure A.17. 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 A.17 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.

Figure A.15: Line chart with mean squared errors of previous 5 days.

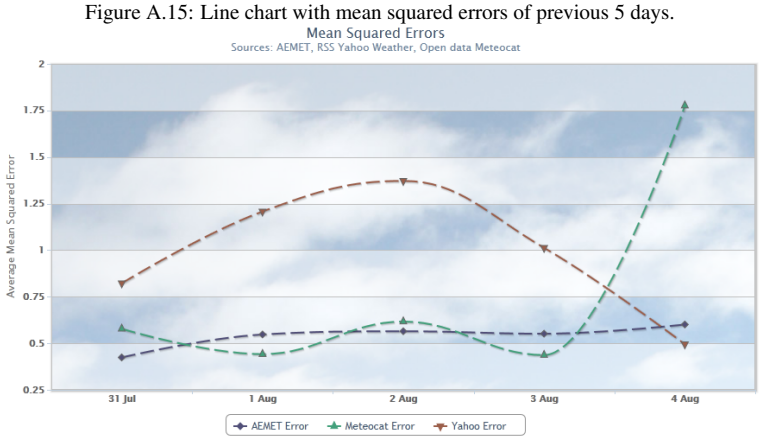


Figure A.16: Form of the page to consult predictions made for a specified day.

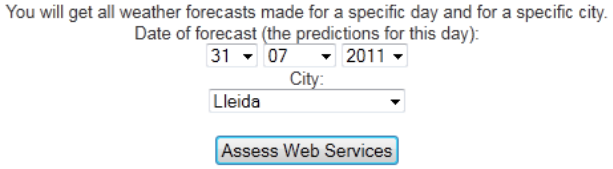


Figure A.17: Table with all the forecasts that have been made for a specified day.

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 A.18: Chart with high temperatures in the predictions made for specified day page.

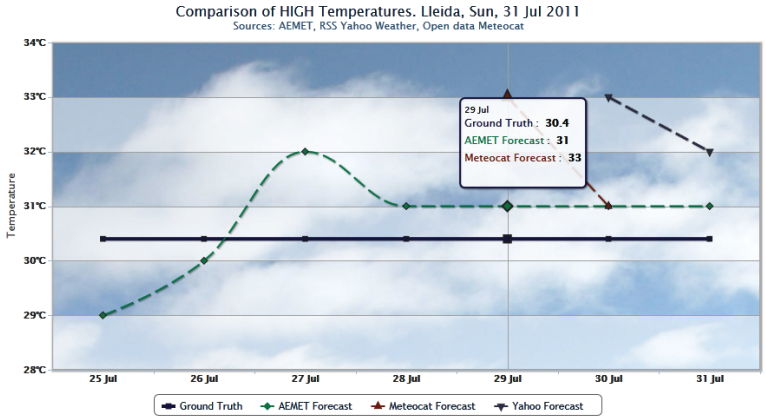
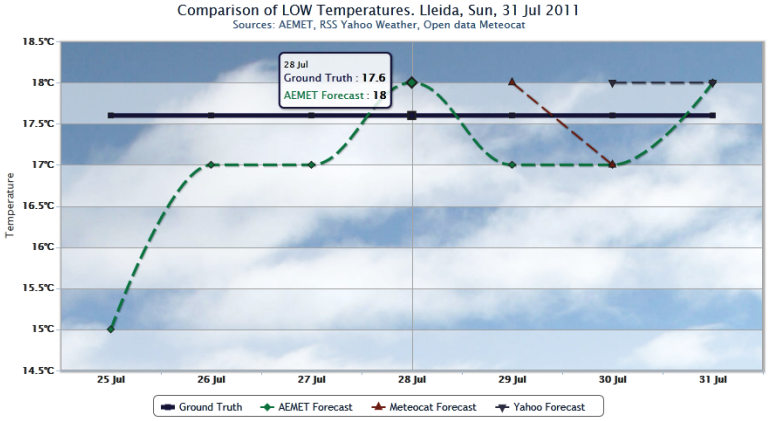


Figure A.19: Chart with low temperatures in the predictions made for specified day page.



The same information which is contained in Figure A.17, 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 A.18 and Figure A.19. 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.

The mean squared error (MSE) and approximation error of the forecasting web services (see Figure A.20), the table (see Figure A.21) and chart (Figure A.22) with mean squared errors of previous 5 days are shown as it is explained in the previous subsection.

Figure A.20: Errors in the predictions made in for specified day page.

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 A.21: Table with MSEs 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 A.22: Graph with MSEs in the predictions made in for specified day page.