Prediction of Web Service Antipatterns Using Machine Learning

by

JOHN KELLY VILLOTA PISMAG

A thesis submitted in partial fulfillment
of the requirements for the degree of
Master of Science
(Software Engineering)
in the University of Michigan - Dearborn
2017

Master's Thesis Committee:

Assistant Professor Marouane Kessentini, Chair
Professor Kiumi Akingbehin
Associate Professor Zhiwei Xu

DEDICATION

To my Dad,

my Mom,

my Bro,

because they made this possible

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

ABSTRACT

Web service interfaces are considered as one of the critical components of a Service-Oriented Architecture (SOA) and they represent contracts between web service providers and clients (subscribers). These interfaces are frequently modified to meet new requirements. However, these changes in a web service interface typically affect the systems of its subscribers. Thus, it is important for subscribers to estimate the risk of using a specific service and to compare its evolution to other services offering the same features in order to reduce the effort of adapting their applications in the next releases. In addition, the prediction of interface changes may help web service providers to better manage available resources (e.g. programmers' availability, hard deadlines, etc.) and efficiently schedule required maintenance activities to improve the quality. In this research, we propose to use machine learning, based on times series, for the prediction of web service antipatterns. To this end, we collected training data from quality metrics of previous releases from 8 web services. The validation of our prediction techniques shows that the predicted metrics value, such as number of operations, which are used to feed the antipattern detection rules on the different releases of the 8 web services were similar to the expected ones with a very low deviation rate. In addition, most of the quality issues of the studied Web service interfaces were accurately predicted, for the next releases. The survey conducted with active developers also shows the relevance of prediction technique for both service providers and subscribers.

Keywords: Web services evolution, prediction, quality of services.

CHAPTER 1. INTRODUCTION

Service-based systems heavily depend on the interface of selected services used to implement specific features. However, service providers do not know, in general, the impact of their changes, during the evolution Web services, on the applications of subscribers. The subscribers are reluctant, in general, to use Web services that are risky and not stable [1]. Thus, analyzing and predicting Web service changes is critical but also challenging because of the distributed and dynamic nature of services. As a consequence, recent studies were proposed to understand the evolution of Web services especially at the interface level [1] [2] [3].

The few existing work studying the evolution of Web services are limited to the detection of changes between different releases [2] or the analysis of the types of change introduced to the service interfaces. Romano et al. [1] proposed a tool called WSDLDiff to detect changes between different versions of a Web service interface based on structural and textual similarities measure. Fokaefs et al. [2] suggested another tool, called VTracker, which uses XML differencing techniques, to detect changes in WSDL documents. However, both tools are just limited to the detection of changes between different Web service releases and did not target the problem of predicting future changes or providing recommendations to the service providers or subscribers about the quality of services interface based on the collected data.

We use, in this paper, the changes collected from previous Web service releases to address the following problems. Most of the changes in a web service interface typically affect the systems of its subscribers. Thus, it is important for subscribers to estimate the risk of using a specific service and compare its evolution to other services offering the same features in order to reduce the effort of adapting their applications in the next releases. Subscribers prefer to use, in general, Web services that are stable with a low risk to include bugs and introduce major revisions in the future. In addition, the prediction of interface changes may help web service providers to better manage available resources (e.g. programmers' availability) and efficiently schedule required maintenance activities to improve the quality of developed services. In fact, the prediction of Web service

changes can be used to identify potential quality issues that may occur in the future releases. Thus, it is easier to fix these quality issues as early as possible before that they become more complex.

In this work, we propose a machine learning approach based on time series to predict the evolution of Web services interface from the history of previous releases' metrics. The predicted interface metrics value are used to predict and estimate the risk and the quality of the studied Web services. We evaluated our approach on a set of 8 popular Web services including more than 90 releases. We report the results on the efficiency and effectiveness of our approach to predict the evolution of Web services interfaces and provide useful recommendations for both service providers and subscribers. The results indicate that the prediction results of several Web service metrics, on the different releases of the 8 Web services, were similar to the expected ones with very low deviation rate. Furthermore, most of the quality issues of Web service interfaces were accurately predicted, for the next releases. The survey conducted with a set of developers also shows the relevance of prediction technique for both service providers and subscribers.

The remainder of this report is as follows: Section 2 presents the related work; Section 3 gives an overview about the proposed predictive modelling technique; Section 4 discusses the obtained evaluation results and possible threats of validity of our experiments. Finally, Section 5 concludes and proposes future research directions.

CHAPTER 2. RELATED WORK

We summarize, in this section, the existing work that focus on studying the evolution of Web services.

Fokaefs et al. [2] used the VTracker tool to calculate the minimum edit distance between two trees representing two WSDL files. The outcome of the tool is the percentage of interface changes such as added, changed, and removed elements among the XML models of two WSDL interfaces. Romano et al. [1] proposed a similar tool called WSDLDiff that can identify fewer types of change than VTracker that may help to analyze the evolution of a WSDL interface without manually inspecting the XML changes. Aversano et al. [4] analyzed the relationships between sets of services change during the service evolution based on formal concept analysis. The main focus of the study is to extract relationships among services.

Several studies have been proposed to measure the similarity between different Web services to search for relevant ones or classify them but not to analyze their evolution. Xing et al. [5] suggested a tool, called UMLDiff to detect differences between different UML diagram versions to understand their evolution. Zarras et al. [6] detected evolution patterns and regularities by adapting Lehman's laws of software evolution. The study was focused only on Amazon Web Services (AWS).

Based on this overview of existing work in the area of Web services evolution, the problem of predicting the evolution of Web services was not addressed before. In addition, the use of machine learning algorithms in Web services was limited to the classification of Web Services and their messages into ontologies [7]. These existing machine learning-based studies are not concerned with the analysis of the releases within the same Web service but more about mining different Web services (one release per service) to classify them in order to help the composition of services process for the subscribers based on their requirements.

Another category of related work focus on detecting and specifying antipatterns in SOA and Web services which is a relatively new area. Rotem-Gal-Oz described the symptoms of a range of SOA antipatterns [8]. Kral et al. [9] listed seven “popular” SOA antipatterns that violate accepted SOA principles. A number of research works have addressed the detection of such antipatterns. Recently, Moha et al. [10] have proposed a rule-based approach called SODA for SCA systems (Service Component Architecture). Later, Palma et al. [3] extended this work for Web service antipatterns in SODA-W using declarative rule specification based a domain-specific language (DSL) to specify/identify the key symptoms that characterize an antipattern using a set of WSDL metrics. Rodriguez et al. [11] [8] and Mateos et al. [12] provided a set of guidelines for service providers to avoid bad practices while writing WSDLs based on eight bad practices in the writing of WSDL for Web services. Recently, Ouni et al. [13] proposed a search-based approach based on standard GP to find regularities, from examples of Web service antipatterns, to be translated into detection rules.

CHAPTER 3. PREDICTION OF WEB SERVICE ANTIPATTERNS USING MACHINE LEARNING

In this chapter, we present an overview of our approach and then we provide the details of our problem formulation and the solution approach.

3.1 Approach Overview

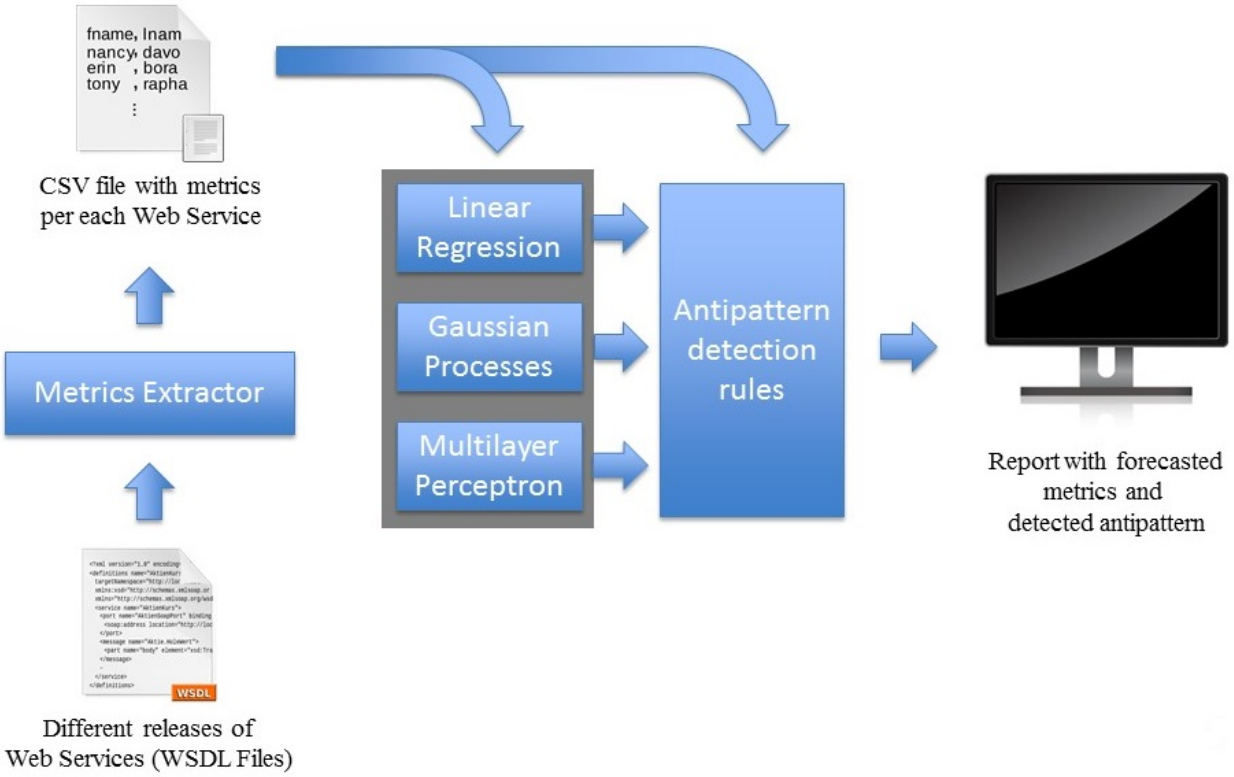

Figure 1. Approach Overview Diagram

As described in Figure 1, our technique takes as input the previous releases of the Web service interfaces to predict its evolution in terms of metrics and the antipatterns detected through them. Those WSDL (Web Services Description Language) files are analyzed in order to calculate metrics in each of the different releases. Table 1 shows the web services and the number of releases used in this research, Table 2 shows which metrics are calculated from each web service release.

Table 1. Web services names and number of releases

Web Service Name	# Releases
Amazon EC2	44
FedEx Rate Service	18
FedEx Ship Service	17
Amazon Mechanical Turk	15
FedEx Track Service	10
Amazon Simple Queue	6
Microsoft Bing Search	5
PayPal Services	7

Table 2. Metrics calculated to each web service release

Metric Name	Definition
NPT	Number of port types
NOD	Number of operations declared
NAOD	Number of accessor operations declared
NOPT	Average number of operations in port types
ANIPO	Average number of input parameters in operations
ANOPO	Average number of output parameters in operations
NOM	Number of messages
NBE	number of elements of the schemas
NCT	Number of complex types
NST	Number of primitive types
NBB	Number of bindings
NBS	Number of services
NPM	Number of parts per message
NIPT	Number of identical port types
NIOP	Number of identical operations
COH	Cohesion
COU	Coupling

Metric Name	Definition
AMTO	Average meaningful terms in operation names
AMTM	Average meaningful terms in message names
AMTMP	Average meaningful terms in message parts
AMTP	Average meaningful terms in port-type names
ALOS	Average length of operations signature
ALPS	Average length of port-types signature
ALMS	Average length of message signature
RCOD	Ratio of CRUDY operations
RAOD	Ratio of accessor operations declared

The metrics calculated are store in CSV files, a file is generated per each web services and each file have metrics for all releases available; those files are used to generate predictions to each one of the metrics selected.

The forecasting algorithms used to generate the predicted values are: Linear regression, Gaussian processes, and Multilayer Perceptron; those three algorithms forecast values of each metric, considering them as an individual element and as a group.

The last part of the process uses the metrics generated and the original metrics to evaluate the error rate of the predictions generated by each algorithm and compare between them. Additionally, those metrics are used to detect Web Antipatterns and the results are analyzed to state the accuracy of the antipatterns prediction.

Table 3. Antipattern detection rules

Antipattern	Detection Rule
Multi service	(NOD(s)≥17 AND COH(s)≤0.43 AND (NOPT(s)≥7.8 OR LCOM≤12) OR (NOD(s)≥24 AND COH(s)≤0.39 AND NPT(s)≥2 AND NST(s)≥41 OR NCT(s)≥32) AND (LCOM3<0.71 OR CBO>4)
Nano service	(NCT(s)≤5 OR NST(s)≤8 AND NPT(s)≤2 AND NOD(s)≤5 AND COH(s)≥0.42) OR (NOPT(s)≤4.2 AND COUP(s)≥0.36 AND COH(s)≥0.39 AND NOD(s)≤6 OR NPT(s)≤2)

Antipattern	Detection Rule
Chatty Service	((NPT(s)≤3 OR DAM≥1) AND NOD(s)≥10 AND RAOD(s)≥0.38 AND (NCT(s)≥15 OR ANOPO(s)≥8.1) AND (NOM(s)>=38 OR NPM(s)>=2.2) AND COH(s)≤0.42)
Data service	((ANIPO(s)≥4 OR ANOPO(s)≥4) AND (NCT(s)≥31 OR NOM(s)>=79) AND (COH(s)≥0.31 OR LCOM3≥0.68 AND DAM≥1) AND NAOD(s)>=13)
Ambiguous Service	(ALOS(s)≤1.6 OR ALOS(s)≥4.9 AND AMTO(s)≤0.6 AND NIOP(s)≥4 OR AMTM(s)≤0.52)

3.2 Forecasting tool

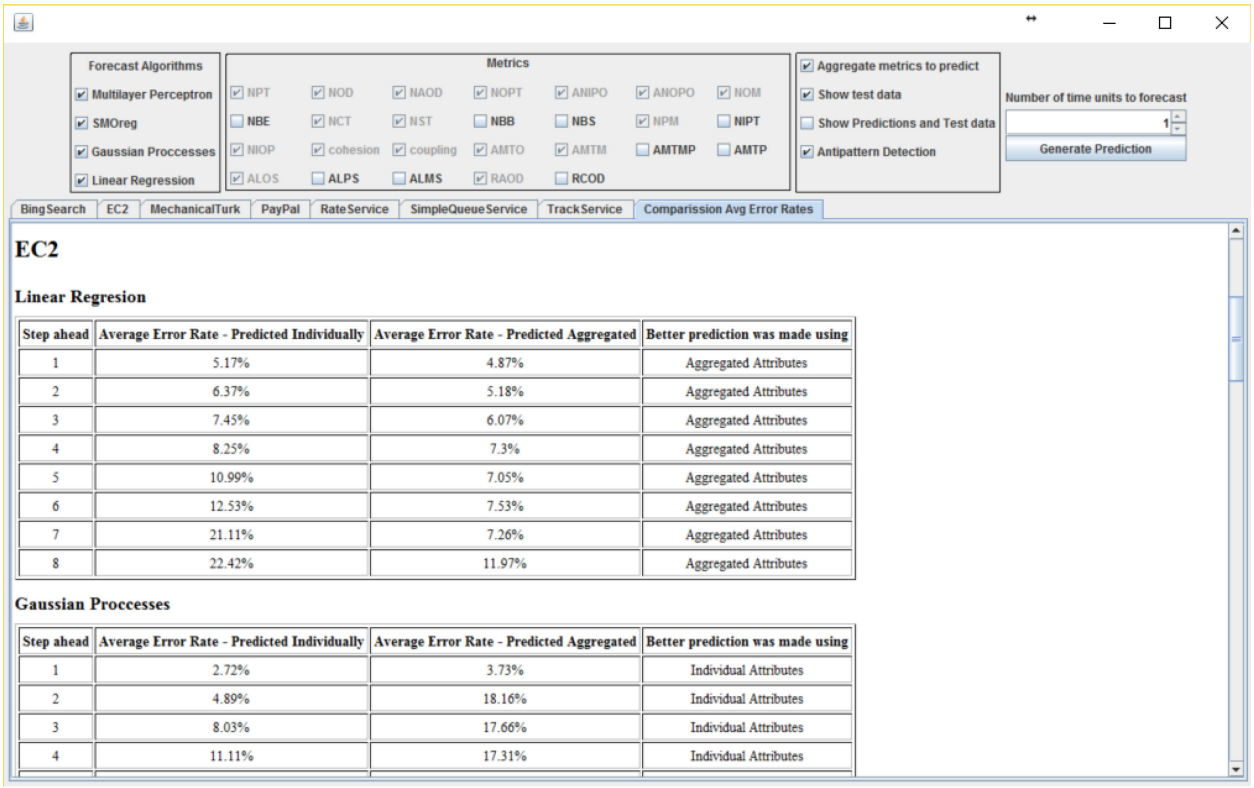

Figure 2. Forecasting tool screenshot

To create customized predictions a tool was created using Java programming language, this tool allows users to choose which algorithms want to use in the prediction and the comparison, which

metrics should to be predicted, what data should be displayed and how many time units or steps the predictions should have.

This tool is feed with WSDL files, those are organized in subdirectories representing each web services to be analyzed and those subdirectories need to be located a folder called WSDL which must be in the same place as the application (JAR file in this case)

CHAPTER 4. VALIDATION

In order to evaluate the ability of our prediction framework to efficiently predict antipatterns of Web services, we conducted a set of experiments based on eight widely used Web services. In this section, we first present our research questions, the experiments setup and then describe and discuss the obtained results. Finally, we discuss some threats related to our experiments.

4.1 Research Questions and Evaluation Metrics

We defined the following three research questions that address the applicability, performance, and the usefulness of our Web services prediction approach. The three research questions are as follows:

RQ1: To what extent can our approach predict correctly the evolution of web services?

RQ2: To what extent can our approach predict antipatterns?

RQ3: Can our prediction results be useful for developers?

To answer RQ1, we calculated the average error rate from the differences between the metrics values predicted and the actual values in calculated in each release, after the previous process the average error rate is group by algorithm and the results generates a new value defined as algorithm error rate. To this end, we considered the list of metrics described in section 3.1. The average error rate and the algorithm error rate are defined as follows:

$$Average\ Error\ Rate = \frac{\sum^{\frac{predicted\ metric\ value - real\ metric\ value}{real\ metric\ value}}}{\#\ metrics}$$

$$Algorithm\ Error\ Rate = \frac{\sum Average\ Error\ Rate}{\#\ Web\ Services\ Forecasted}$$

We calculated the error rate for one and many steps (releases) over time for every of the considered web services.

To answer RQ2, we calculated the antipattern detection accuracy where the average of the correctly predicted antipatterns is totalized and use to calculate the algorithm antipattern detection accuracy. The antipattern detection accuracy and the algorithm antipattern detection accuracy are defined as follows:

$$Antipattern\ detection\ accuracy = \sum \frac{Correct\ antipaterns\ predition}{Total\ Posible} \quad (5)$$

$$Algorithm\ antipattern\ detection\ accuracy = \frac{\sum Antipattern\ detection\ accuracy}{\#\ Web\ Services\ Forecasted}$$

We considered five types of antipatterns from the literature [10]: Multi-service (MS: a service implementing many operations), Nano-service (NS: too-fine grained service), Chatty-service (CS: a service including many fine-grained operations), Data-service (DS: a service including only data access operations) and Ambiguous service (AS: a service including ambiguous names of operations). More details about existing Web service antipatterns can be found in the following references [10] [3]. We used the manually defined rules in [13] to detect the predicted and actual Web service antipatterns.

To answer RQ3, we used a post-study questionnaire that collects the opinions of developers on our prediction results and antipatterns detection. We also wished to assess how these results may help developers working on services-based applications. To this end, we asked 24 software developers, including 11 developers working in a Web development startup and providing some Web services for customers from the automotive industry sector. The remaining participants are 13 graduate students (8 MSc and 5 PhD students) in Software Engineering at the University of Michigan-

Dearborn. 9 out the 13 students are working either full-time or part-time programmers in Software industry. All the participants are volunteers and have a minimum of 2 years of experience as a developer. The participants were first asked to fill out a pre-study questionnaire containing five questions. The questionnaire helped to collect background information such as their role within the company, their programming experience, their familiarity with Web services and service-based applications. In addition, all the participants attended one lecture about Web service antipatterns and passed five tests to evaluate their performance to evaluate the design of Web services using quality metrics.

4.2 Studied Web Services

We selected these 8 Web services for our validation because different releases of their WSDL interface are publicly available and belong to different categories. Table 2 provides some descriptive statistics about these eight Web services:

- Amazon EC2: Amazon Elastic Compute Cloud is a web service that offers resizable compute capacity in the cloud. In this study, we have considered a total of 44 releases from 2006 until 2014.
- Amazon Simple Queue Service (Amazon SQS) offers reliable hosted queues for storing messages exchanged between computers. We considered in our study a total of 6 releases.
- FedEx Track service offers accurate update of the status of shipments. We used 10 releases from this Web service.
- FedEx Ship Service: The Ship Service provides functionalities for managing package shipments and their options. A total of 17 releases are considered in our experiments from this Web service.
- FedEx Rate Service: The Rate Service provides the shipping rate quote for a specific service combination depending on the origin and destination information supplied in the request. We used 18 releases for our prediction algorithm.
- Amazon Mechanical Turk Requester: it is a web service that provides an on-demand, scalable, human workforce to complete jobs that humans can do better than computers

such as recognizing objects in photos. We used 15 releases developed between 2005 until 2012.

- Microsoft Bing: is a web service which allow users submit queries to and retrieve results from the Bing Engine. We used 5 releases from version 2.0 to version 2.4.
- PayPal: is a web service which allow user to incorporate the functionalities offered by PayPal in applications oriented to web and mobile environments. We used 4 releases.

4.3 Results

4.3.1 Results for RQ1.

Figure 3 shows the results obtained from calculating the average error rate in each web services selected using individual metrics prediction.

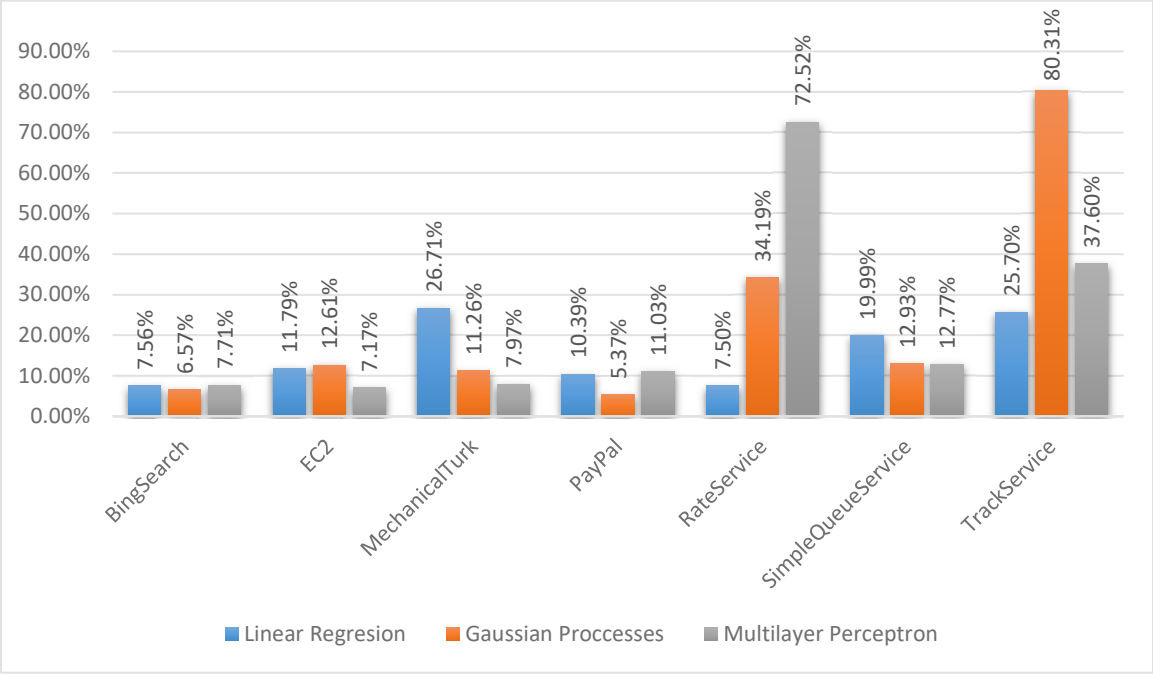

Figure 3. Algorithm average error rate using individual metrics comparison per web service

Figure 4 shows the results obtained from calculating the average error rate in each web services selected using aggregated metrics prediction.

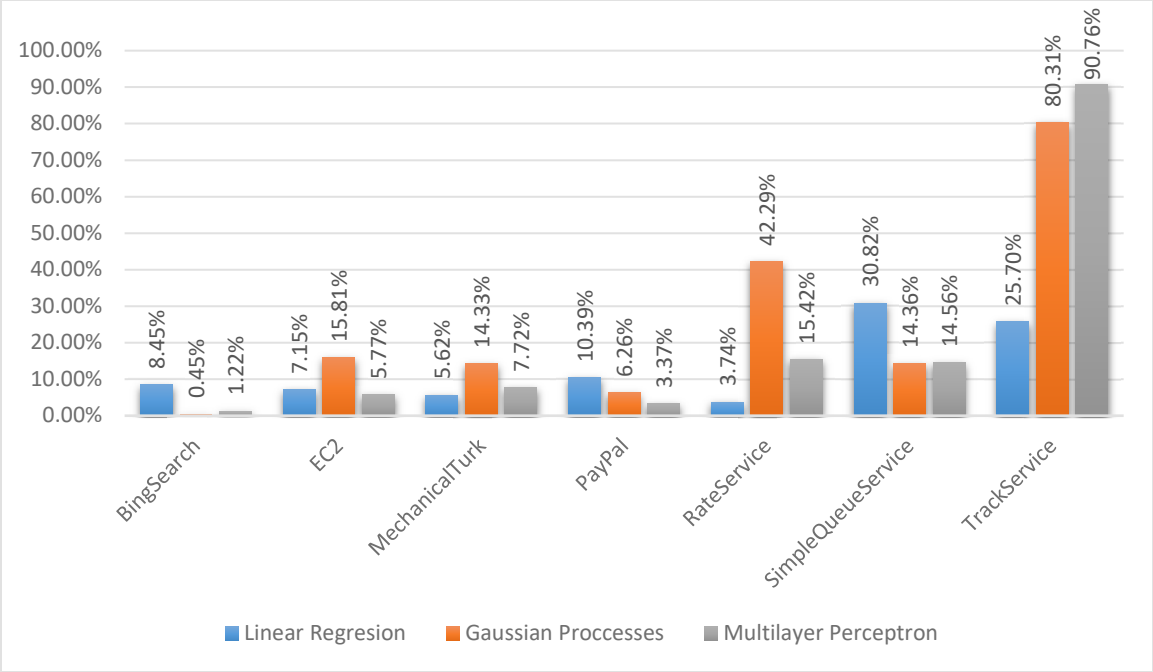

Figure 4. Algorithm average error rate using aggregated metrics comparison per web service

The first thing to notice is the no correlation between the number or releases and the accuracy of the prediction calculated; Amazon EC2 web service presents accurate results been the web services with more releases to be analyzed, but Microsoft Bing Search web service which has 5 releases got in average the better predictions.

The second, and most important aspect analyzed, is how the error rate values decrease using aggregated metrics to calculate the predictions. Figure 5 shows the differences between both types of predictions, and the cases where there is an actual increasing in the prediction accuracy.

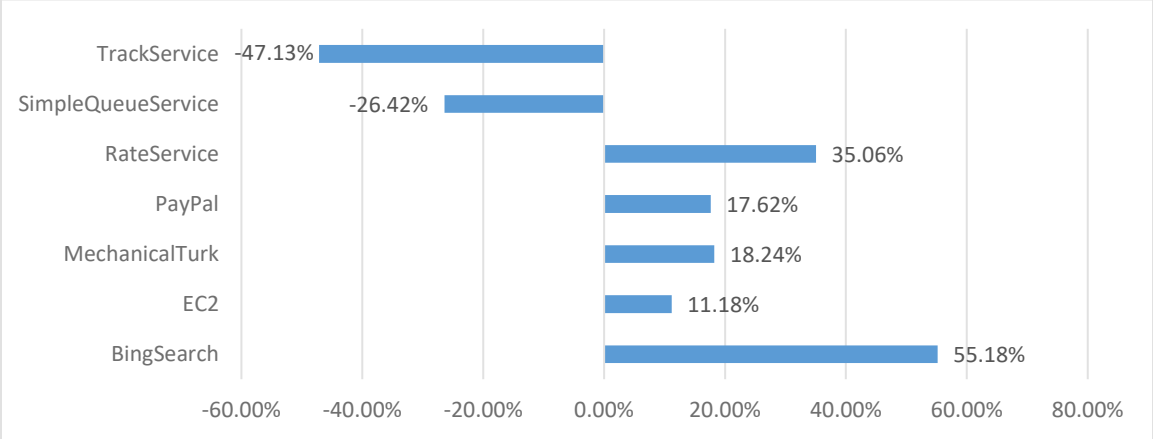

Figure 5. Percentage difference between individual and aggregated predictions per web service

As the figure 5 shows, Linear regression algorithm offers the best predictions with 15.66% of error predicting metrics individually and 13.13% of error predicting metrics as group.

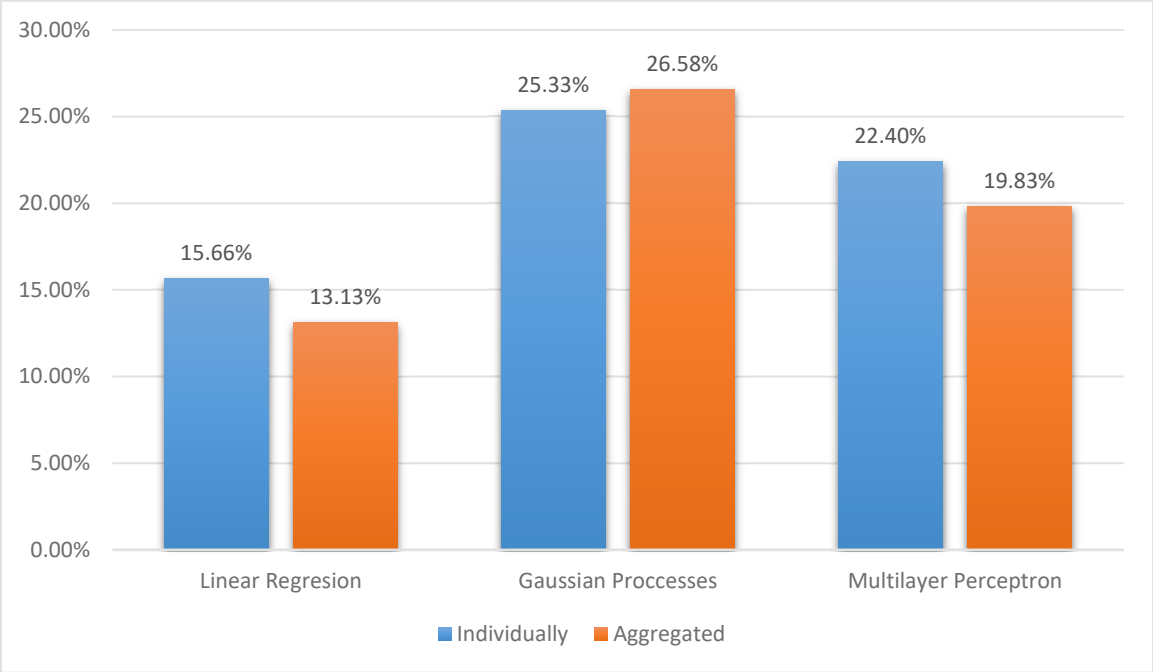

Figure 6. Algorithm average error rate comparison

Reviewing individual values, Gaussian Processes got the lower error rate with 0.45% using aggregated attributes prediction on Bing Search web service followed by Multilayer Perceptron with 1.22% in the same web service and using aggregated attributes. On the other extreme, Multilayer Perceptron got the highest error rate with 90.76% using aggregated attributes prediction on FedEx Track Service web service followed by Gaussian Processes with 80.31% in the same web service and using either individual attributes or aggregated attributes.

Is important to notice the reduction of the error rate if the metrics values are predicted as a group or aggregated, comparing the average error rate for each release metrics predicted in both cases the results shows 67% more precision using this method. (See figure 4.)

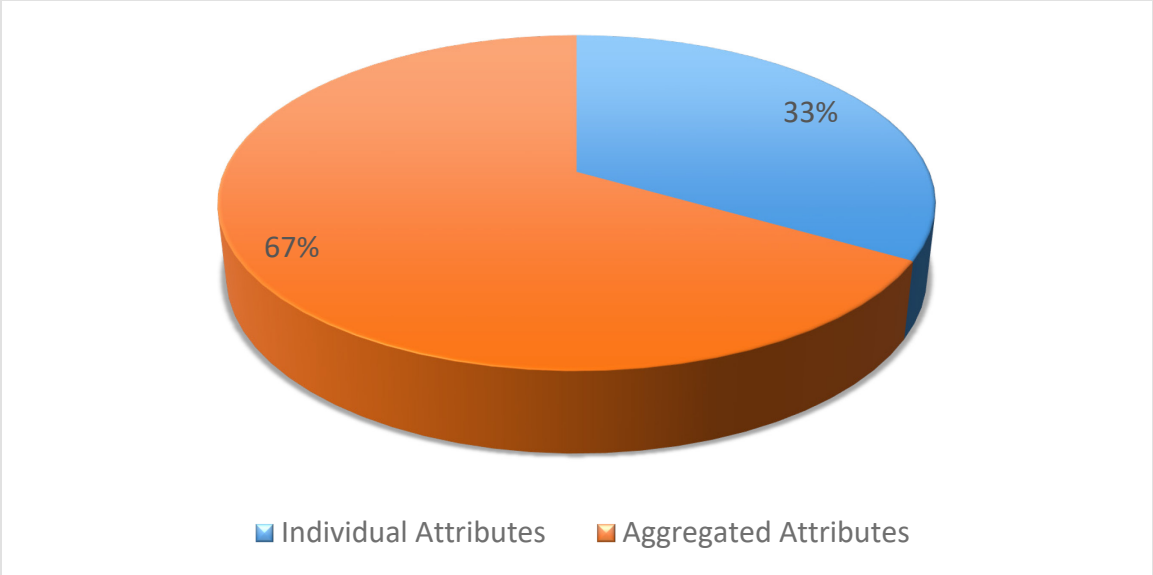

Figure 7. Percentage of better results generated using aggregated and individual attributes

4.3.2 Results for RQ2.

As is expected, the algorithm with best metrics prediction will be the most accurate to help in the prediction of antipatterns. As figure 5 shows, Linear Regression has a 99.05% of concordance between the expected antipatterns and the predicted ones.

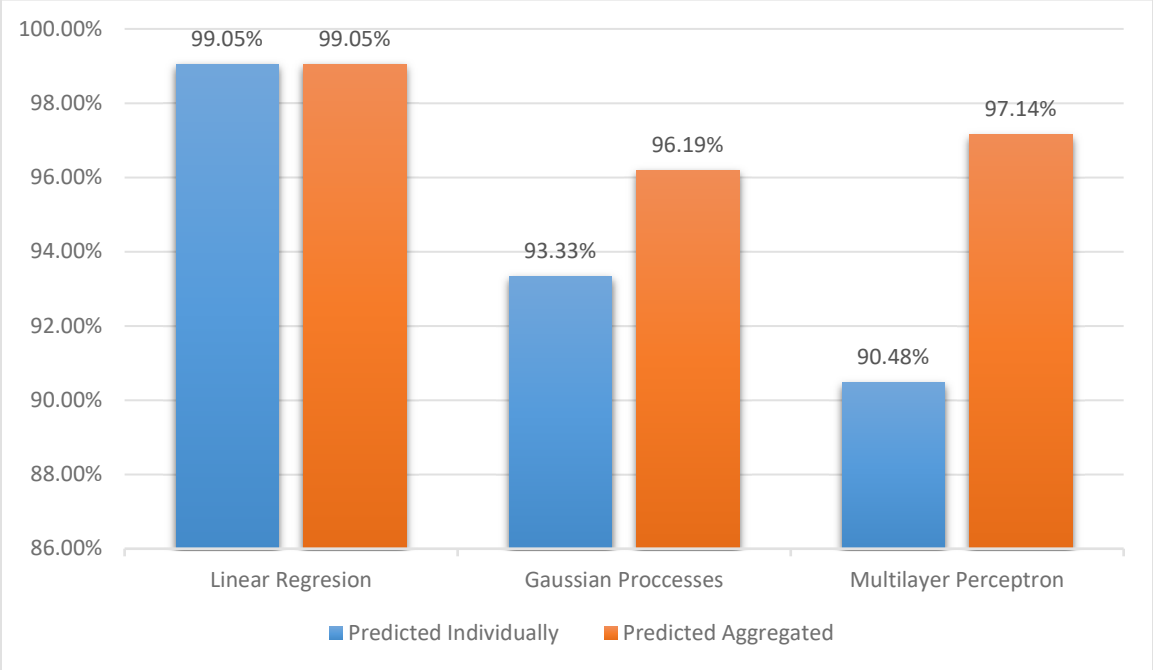

Figure 8. Comparison of the algorithms accuracy of the antipattern prediction

In addition, the metrics predicted individually and aggregated generates antipatterns detection with high values (superior to 94%) but the higher detection prediction is obtained using aggregated attributes.

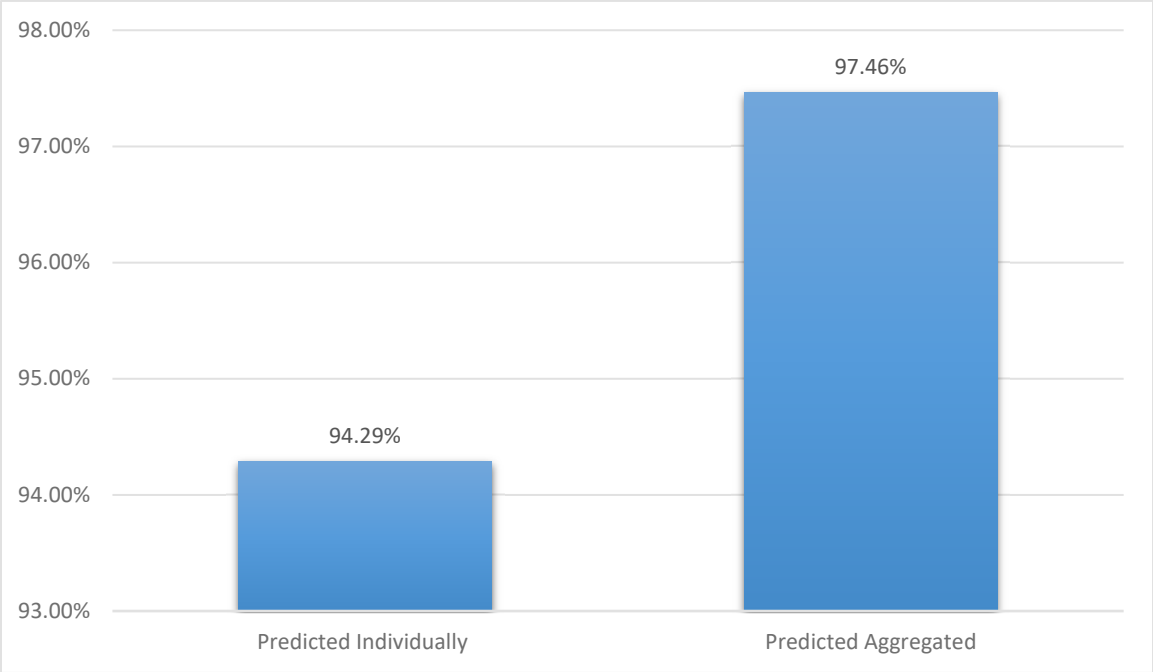

Figure 9. Comparison of the accuracy of the antipattern prediction using individual and aggregated attributes

To summarize, it is clear based on the obtained results that Linear Regression algorithm predicting attributes as a group offers the most accurate prediction of future presence of web antipatterns.

4.3.3 Results for RQ3.

To answer RQ3, we used a post-study questionnaire to the opinions of the participants about their experience in using our prediction tool and results. The questionnaire asked participants to rate their agreement on a Likert scale from 1 (complete disagreement) to 5 (complete agreement) with the following statements:

- The predicted metrics value are useful to estimate the risk and cost of using a specific Web service and may help the developer to select the best service based on his preferences.
- The predicted quality issues may help developers and managers to better schedule maintenance activities and reduce the cost of fixing these issues.

The agreement of the participants was 4.6 and 4.8 for the first and second statements respectively. This confirms the usefulness of our prediction results for the developers considered in our experiments.

The remaining questions of the post-study questionnaire were about the benefits and also limitations (possible improvements) of our prediction approach. We summarize in the following the feedback of the developers. Most of the participants mention that our results may help developers of the service providers to decide when to refactor their Web service implementations. For example, they can consider to perform some refactorings when the prediction results show that the quality issue may become much more severe after few releases such as a multi-service antipattern. Thus, the developers liked the functionality of our tool that helps them to identify refactoring opportunities as early as possible.

The participants found our tool helpful for also the developers of Service-based applications. In fact, the majority of the participants mention that they consider the stability and quality of services as important criteria to select a Web service when several options are available. The non-stability of a service may negatively impact their systems in the future and it is maybe an indication that the used service includes many bugs explaining several new releases. Furthermore, the subject liked the prediction of antipatterns feature since it is easier for them to evaluate the quality of Web services in next releases based on the number of antipatterns rather than analyzing a set of metrics.

The participants also suggested some possible improvements to our prediction approach. Some participants believe that it will be very helpful to extend the tool by adding a new feature to automatically calculate the risk, cost and benefits of using different possible Web services. Another possibly suggested improvement is to use some visualization techniques to evaluate the evolution of the We services to easily estimate their stability.

CHAPTER 5. CONCLUSION

This research proposed an approach to predict the evolution of web services and use this information to predict . In fact, it is maybe important for subscribers to estimate the risk of using a selected service and compare its evolution to other possible services offering the same features. Furthermore, the prediction of future changes may help web service providers to better manage available resources and efficiently schedule required maintenance activities to improve the quality. In this paper, we propose to use machine learning, based on Artificial Neuronal Networks, for the prediction of the evolution of Web services interface design. To validate the proposed approach, we collected training data from quality metrics of previous releases from 6 Web services. The validation of our prediction techniques shows that the predicted metrics value, such as number of operations, on the different releases of the 6 Web services were similar to the expected ones with a very low deviation rate. In addition, most of the quality issues of the studied Web service interfaces were accurately predicted, for the next releases, with an average precision and recall higher than 82%. The survey conducted with developers also shows the relevance of prediction technique for both service providers and subscribers.

Future work involves validate our prediction technique with additional metrics, Web services and developers to conclude about the general applicability of our methodology. Furthermore, in this paper we only focused on the prediction of Web services evolution. We plan to extend the approach by defining new risk measures based on the predicted metrics value. In addition, we will study of the impact of predicted quality issues on the usability and popularity of Web services over time.

REFERENCES

[1] D. Romano and M. Pinzger, "Analyzing the Evolution of Web Services Using Fine-Grained Changes," in *IEEE 19th International Conference on Web Services (ICWS)*, Honolulu, 2012.

[2] M. Fokaefs, R. Mikhaiel, N. Tsantalis, E. Stroulia and A. Lau, "An Empirical Study on Web Service Evolution," in *IEEE International Conference on Web Services*, 2011.

[3] Y.-G. Guéhéneuc, G. Tremblay, N. Moha and F. Palma, "Specification and Detection of SOA Antipatterns in Web Services," in *Software Architecture*, Vienna, Springer International Publishing, 2014, pp. 58-73.

[4] L. Aversano, M. Bruno, M. Di Penta, A. Falanga and R. Scognamiglio, "Visualizing the evolution of Web services using formal concept analysis," in *IWPSE '05 Proceedings of the Eighth International Workshop on Principles of Software Evolution*, Lisbon, Portugal, 2005.

[5] Z. Xing and E. Stroulia, "UMLDiff: an algorithm for object-oriented design differencing," in *ASE '05 Proceedings of the 20th IEEE/ACM international Conference on Automated software engineering*, Long Beach, California, 2005.

[6] L. Dinos, P. Vassiliadis and A. V. Zarras, "Keep Calm and Wait for the Spike! Insights on the Evolution of Amazon Services," in *Advanced Information Systems Engineering*, vol. 9694, Ljubljana, Springer International Publishing, 2016, pp. 444-458.

[7] "SAWSDL-MX2: A Machine-Learning Approach for Integrating Semantic Web Service Matchmaking Variants," in *ICWS 2009. IEEE International Conference on Web Services*, Los Angeles, California, 2009.

[8] J. M. Rodriguez, M. Crasso, A. Zunino and M. Campo, "Automatically Detecting Opportunities for Web Service Descriptions Improvement," in *Software Services for e-World*, Buenos Aires, Argentina, 2010.

[9] J. Král and M. Žemlicka, "Popular SOA Antipatterns," in *2009 Computation World: Future Computing, Service Computation, Cognitive, Adaptive, Content, Patterns*, Athens, Greece, 2009.

[10] J.-M. Jézéquel, B. Baudry, Y.-G. Guéhéneuc, B. J. Conseil, M. Nayrolles, F. Palma and N. Moha, "Specification and Detection of SOA Antipatterns," in *Service-Oriented Computing*, Shanghai, Springer Berlin Heidelberg, 2012, pp. 1-16.

[11] J. M. Rodriguez, M. Crasso, C. Mateos and A. Zunino, "Best practices for describing, consuming, and discovering web services: a comprehensive toolset," *Software Practice and Experience,* vol. 43, no. 6, p. 613–639, 2012.

[12] C. Mateos, J. M. Rodriguez and A. Zunino, "A tool to improve code-first Web services discoverability through text mining techniques," *Software: Practice and Experience,* vol. 45, no. 7, p. 925–948, 2014.

[13] A. Ouni, R. G. Kula, M. Kessentini and K. Inoue, "Web Service Antipatterns Detection Using Genetic Programming," in *GECCO '15 Proceedings of the 2015 Annual Conference on Genetic and Evolutionary Computation*, Madrid, Spain, 2015.