International Journal of
INTELLIGENT SYSTEMS AND APPLICATIONS IN ENGINEERING
ISSN:2147-6799 www.ijisae.org Original Research Paper

Novel Technique to Predict the Evolution in SOA Based Services

Zeenat Parween[1], R.B.S. Yadav[1]

Submitted: 29/01/2023 **Accepted:** 04/04/2023

Abstract: An essential part of a Service-Oriented Architecture (SOA) is interface of the service, which functions as an agreement between the one who provides the service and the clients. In order to accommodate changing requirements, these interfaces are often updated. However, a web service's subscribers' systems are frequently impacted by modifications to the interface. As a result, it's critical for users to assess danger of utilizing particular service and equivalence its development to rest of services that offer the similar functionality to minimize effort required to modify their applications in subsequent releases. Furthermore, foreseeing interface changes may assist online service providers in more effectively managing their resources (such as programmers' accessibility, strict timelines etc.) and scheduling necessary maintenance tasks to raise standard of their services. In this article, we suggest using artificial neural networks-based machine learning to forecast how the architecture of Web services interfaces will change over time. In order to achieve this, we gathered training data from six Web services' quality indicators of earlier releases. The validation of our prediction approaches reveals that, with a very less deviation rate, predicted metrics values for the various releases of six SOA based services, such as number of operations, were extremely similar to the ones that were expected. Additionally, with an average precision and recall more than 85%, the majority of quality concerns of examined Web service interfaces were correctly anticipated for subsequent releases. The study done through working developers demonstrates the value of prediction methods for both service consumers and providers.

Keyword: *demonstrates, prediction, essential, effectively*

1 Introduction

The interface of carefully chosen services that are utilised to implement particular functionality strongly influences service-based systems. However, service providers are generally unaware of how the modifications while SOA based services evolves, may affect subscribers' softwares. In general, clients are hesitant to use unstable and dangerous Web services [10]. Hence, evaluation and because Web services are distributed and dynamic, it is important but difficult to foresee when they will change. Recent research have therefore been suggested to comprehend the growth of SOA based services, particularly at Web Service Description Language (WSDL) file level [9, 10, 19].

The recent research that has been done on Web service evolution is confined to identifying changes between releases [9] or analysing the kinds of modifications made to interfaces of SOA based services. Utilizing structural as well as textual similarity measures, Romano et al. [10] introduced a tool known as WSDLDiff to identify evolution made to a Web service interface between versions. Another tool, named VTracker, was recommended by Fokaefs et al. [9], which employs XML differencing method to find evolution in interface files. Though, neither solution addresses the issue of forecasting upcoming evolution nor offering suggestions to the one who provides the service and the clients about the QoS of interface based on discovery of evolution between successive SOA based service releases.

In this article, we handle the following using the changes gathered from prior Web service releases the following issues. The systems of a web service's subscribers are often affected by the majority of changes to the interface. As a result, it's critical for customers to assess the danger of utilising a particular service and to minimize time required to modify their applications in upcoming releases, equivalence its changes to rest of services presenting similar functionalities. In general, subscribers favour using Web services that are reliable and have little chance of developing faults or introducing significant alterations in the future.

Additionally, foreseeing interface modifications may assist online service providers in more effectively managing available resources (such as programmers' availability) and scheduling necessary maintenance tasks to raise the caliber of generated services. In fact, predicting changes to Web services can be utilised to spot potential quality problems that can arise in upcoming releases. Therefore, it is simpler to address these quality problems as soon as possible, before they worsen and become more complicated.

1 Department of Computer Science and Mathematics, Magadh University,
Bodh Gaya, Bihar 824234, India

To forecast the changes of SOA based service interface from the history of earlier releases' metrics, we present a machine learning approach in this study utilizing Artificial Neural Networks (ANN) [5]. Estimated value of the projected interface metrics is utilized to forecast and gauge the danger and effectiveness of research.

The testing of the proposed strategy has been done using six well-known Web services with more than 100 versions. We provide the findings about the efficacy and efficiency of our methodology for forecasting changes of Web services interfaces and offer practical advice for the one who provides the service and the clients. Findings shows very little departure from the expected outcomes when numerous SOA based service metrics were predicted for various releases of six SOA based services. Furthermore, with an average precision and recall more than 85%, majority of Web service interface quality issues were correctly forecasted for the upcoming releases. The poll done among a group of creators also demonstrates value of prediction techniques for both the one who provides the service and the customers.

Following is continuation of this research: The relevant study is presented in Section two, developed predictive modelling technique is defined in Section three, and the evaluation results and potential challenges to the validity of our findings are discussed in Section 4. Sect. 5 ends and suggests directions for additional investigation.

2 Related Work

The current part presents a summary of previous research on development of SOA based services.

The VTracker programme was used by Fokaefs et al. [9] to determine the smallest edit distance among 2 trees that represented 2 interface files. Tool's output is proportion of XML models for two WSDL interfaces that have had items added, altered, or removed. In order to analyse the development of an interface when not manually reviewing evolution in , Romano et al. [10] introduced the WSDLDiff tool, which is comparable to VTracker but can discover fewer types of change. On the basis of formal concept analysis, Aversano et al. [11] examined how links between various groups of services evolve over course of changes in service. The primary goal of numerous studies have been put out to identify or search for relevant Web services by determining how similar they are, but not to evaluate their evolution. A tool named UMLDiff was recommended by Xing et al. [12] to identify discrepancies between several UML diagram versions in order to comprehend their historical development. Lehman's laws of software evolution were modified by Zarras et al. [13] to identify evolution patterns and regularities. The investigation was limited to Amazon Web Services (AWS).

Issue of anticipating changes in SOA based services has not been researched earlier, according to this survey of previous research in the field. Additionally, only the categorization of SOA based services and their messages into ontologies was possible with the application of machine learning techniques in SOA based services [22].Present machine learning-based studies are more focused on mining distinct SOA based services to categorize these services and aid in combination of SOA based services for customers depending on their needs rather than analysing releases within identical SOA based service.

Another area of related research focuses on identifying and defining antipatterns in SOA and Web services, a more recent area. A variety of SOA antipatterns' symptoms were described by Rotem- Gal-Oz [15]. Seven "common" SOA antipatterns that go against established SOA guidelines were listed by Kral et al. [18]. The identification of such antipatterns has been the subject of numerous research studies.

Recently, Moha et al. [20] suggested a rule-based strategy for SCA systems termed SODA (Service Component Architecture). Later, Palma et al. [19] expanded on this research to include SOA based service antipatterns in SODA-W using declarative rule specification based on a domain-specific language (DSL) to detect important indications which describe an antipattern by means of a group of interface metrics. Using 8 bad practises in development of interface for SOA based services, Rodriguez et al. [14, 15] and Mateos et al. [16] presented a group of principles for the one who provides the service to follow when producing WSDLs. A search-based approach utilizing conventional GP was recently developed by Ouni et al. [7] to uncover regularities from samples of SOA based service antipatterns that could then be converted into detection rules.

Use of ANN algorithm to forecasting of changes in SOA based services is covered in the next section.

3 SOA Based Services: Evolution Prediction

As shown in Fig. 1, our method uses historical Web service interface releases to predict how they will change over time, a comprehensive set of metrics to forecast, and a set of detection rules to find probable

upcoming quality problems known as Web service antipatterns. Output of the proposed method is a set of anticipated evolution metric values and potential future quality problems for following release.

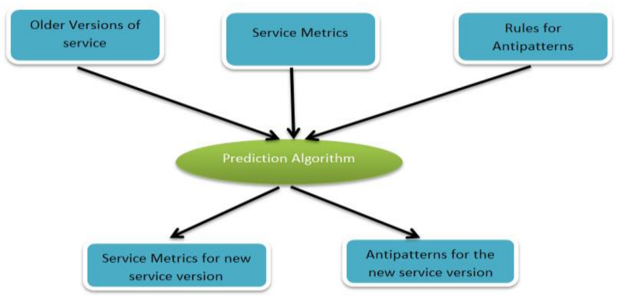
Older Versions of service
Service Metrics
Rules for Antipatterns
Prediction Algorithm
Service Metrics for new service version
Antipatterns for the new service version

Fig. 1 Proposed technique overview

Our prediction methodology is based on artificial neural networks (ANNs), a machine learning technology. ANN adaption to our challenge of forecasting the changes in SOA based services is described below.

Artificial Neural Network (ANN): ANN models are mathematical representations of nervous system's operation [2–5]. It is made up of numerous interconnected objects known as artificial neurons. ANNs are built on learning. It is a feature of those systems which adapt themselves that can get better at solving problem because of prior practice [1]. A mapping among a collection of I/Ps and their corresponding O/Ps is created by an ANN. This model is capable of handling noisy signals and incomplete data in non-linear regression analysis. This model is capable of managing noisy signals as well as incomplete data in non-linear regression analysis. Multi-Layer Perception ANN (MLP-ANN) was employed in this study [2]. MLP-ANNs are intriguing for modelling black-box functions because they are universal approximators, for which very less is known related to their form. Each neuron's O/P is stated as below:

$$y = \phi\left(\sum_{i=1}^{n} w_i a_i + b\right)$$

Here, w stands for weight vector, a for input vector, b for bias, for activation function, and n for the hidden layer's number of neurons. It takes an exponential number of hidden neurons to cover whole I/P space since a hidden neuron only affects the outputs of the network for I/Ps that are close to its centre. Due to this, it is recommended that MLP-ANN be used for problems requiring a limited number of I/Ps, such as our prediction of the evolution of SOA based services challenge.

ANNs ae used because they are among the most accurate predictive models, particularly when dealing with noisy and lacking data. All of the neurons in its multilayered architecture are fully connected, and the weights of the connections were established at random at the start of the training. The sigmoid function is utilised [5] as the activation function since it is suitable for continuous data. The initial layer of the network, which consists of three layers, is made up of p input neurons. The number x_{kt} is given to each neuron. A group of hidden neurons make up the hidden layer. The network can be trained using the learning algorithm, which is an iterative method. Its performance is determined by two factors. The momentum factor, which attempts to prevent local minima by stabilizing weights, is initial parameter. Learning rate, which determines how quickly weights are adjusted, is the second factor.

education procedure. Data for training set should be standardized before the learning process. Since the min-max strategy is one of the most accurate techniques, we decided to use it in this instance [8]. As shown in Table 1, we adapted the below list of indicators from the work [7] to forecast release dates of the upcoming Web services.

Table 1: Metrics for interface of SOA based service

Name of Metric	Description
NBE	# of Elements of Schemas
NCT	# of Complex types
NST	# of Primitive types
NOM	# of Messages
NOD	# of Operations

ANIPO	# of input parameters in operations
ANOPO	# of output parameters in operations

Proposed ANN solutions are expressed as below during the learning process:

let's indicate by O matrix that contains mathematical values associated to collection of metrics to forecast. O is made up of p columns and n lines, where p denotes count of steps and n denotes count of metrics to forecast (releases).

$$O = \begin{bmatrix} x_{11} & x_{12} & \cdots & x_{1p} \\ x_{21} & x_{22} & \cdots & x_{2p} \\ \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot \\ x_{n1} & x_{n2} & \cdots & x_{np} \end{bmatrix}$$

Learning technique. Depending upon whether the ANN model is linear or non-linear, there are many learning algorithms. Back-propagation (BP), a method of supervised learning, is used by our MLP model to train network. Modified linear perceptron, or MLP, might differentiate among data that are not linearly separable. One of the most well-known and often utilised training methods is BP, which is extensively discussed in the literature [5]. The learning rate () and momentum () parameters used in our BP neural network's training have been set to moderate values. Each time a training vector is shown to network, the weights are updated. Sum square error is used as network's exit strategy or end condition until it hits a certain threshold determined before the network was started. Based on the default configuration of the Weka1 framework, our implementation.

4 Experiments

We carried out a series of experiments using six frequently used Web services in order to assess how well our prediction system was able to forecast change trends of SOA based services.

The setup of the tests, our research questions, and the description and discussion of the findings are all included in this section. Finally, we talk about various risks that pertain to our research.

Research Questions and Evaluation Metrics

Our Web services prediction approach's applicability, performance, and utility are all addressed in the following three study problems that we have developed. The following are 3 research questions:

RQ1: How far our method accurately forecast how Web services will change?

RQ2: To what extent can we foresee problems with Web service quality using our approach?

RQ3: Can developers apply the findings of our predictions?

Utilizing ANNs algorithm on several SOA based service releases, we determined difference between real expected metrics value and forecasted ones to respond to RQ1. In order to achieve this, we took into account the list of indicators mentioned in the preceding section. The following is the definition of error rate:

$$e\_rate(M_i, S) = |PM_i - EM_i|,$$

where EM is the expected value and PM is measure that was predicted by means of ANNs. For each of the Web services taken into consideration, we determined the error rate for single and multiple releases over time.

In order to compare the predicted and expected Web services antipatterns, we produced precision and recall scores to address RQ2:

$$\mathrm{RC}\mathit{recall} = \frac{\text{predicted antipatterns} \cap \text{expected antipatterns}}{\text{expected antipatterns}} \in [0,1]$$

$$\mathrm{PR}_{\text{precision}} = \frac{\text{predicted antipatterns} \cap \text{expected antipatterns}}{\text{predicted antipatterns}} \in [0,1]$$

From the literature, we took into account five categories of antipatterns [20]: Information service (IS: SOA based service containing only information connect activities), Fine Grained service (FGS: SOA based service containing many fine-grained operations), Multi service (MS: SOA based service executing multiple operations), Heavily Grained service (HGS: heavily grained SOA based service), and Vague service are examples of services (VS: SOA based service containing vague titles of operations). You can get additional information on current SOA based service containing antipatterns in cited references [19, 20]. To find anticipated and real Web service antipatterns, we applied the manually developed rules in [7].

We employed a post-study questionnaire to get developers' thoughts on the outcomes of our predictions in order to respond to RQ3. We also wanted to determine how these findings may benefit programmers creating SOA based systems. In order to achieve the same, twenty four programmers were polled, comprising of eleven who were part of a SOA based work as well as offered certain services to clients in locomotive area. Thirteen undergraduates (8 MTech and 5 PhD) who were studying software engineering in University make up the remaining participants. Nine of the 13 students are employed as programmers in the software business, either full- or part-time. All of the members are helpers with at least two years of development experience. First, pre-study questionnaire with five items was given to the participants. The questionnaire assisted in gathering background data on the respondents, such as their position in organization, skill with programming, and acquaintance with SOA based services and SOA based systems. Additionally, each participant took part in a presentation on SOA based service antipatterns and cleared 5 exams that assessed how well they understood how to use quality metrics to assess the design of Web services.

4.1 Considered SOA based services

These six Web services were chosen for authentication due to various versions of interface are accessible to the public as well as fall under various groups. The following six Web services are described statistically in Table 2:

Name of the service	Number of versions considered	Average # of antipatterns
Amazon Elastic Book Store	35	121
Eucalyptus	28	82
AWS Migration Hub	25	11
Elastic Load Balancing	31	91
AWS CodePipeline	11	18
AWS CodeStar	8	15

- Amazon Elastic Book Store: Amazon Elastic Block Store (Amazon EBS) is an easy-to-use, scalable, high performance block-storage service. It is developed for Amazon Elastic Compute Cloud (Amazon EC2).
- Eucalyptus: It was originated by the Eucalyptus organization. The objective of the service was to develop cloud computing components which are helpful in developing Amazon Web Services.
- AWS Migration Hub: is used to give a common place to gather the data required for the valuation, preparation, and monitoring of migration of other applications to AWS.
- Elastic Load Balancing: This service allows you to routinely dispense inbound traffic of various applications among various destinations and virtual devices. These devices could be in multiple Availability Zones (AZs).
- AWS CodePipeline service aims to provide you a smooth release of various versions of your application and to give regular updates on the setup of your application.
- AWS CodeStar allows one to rapidly design, develop, and use applications on AWS. It gives an integrated user interface which allows one to effortlessly manage the s/w development tasks at a single location.

4.2 Results

Outcomes for RQ1. Figures 2, 3, and 4 shows outcomes for 1st RQ. According to Fig. 3, the majority of SOA based service metrics are correctly forecast on various SOA based services with an average error rate less than 3. The average mistake rate is higher for the AWS CodeStar and AWS Code Pipeline services. This can be because the training set for this service is less than those for other services.

Due to the substantial training data available for Elastic Load Balancing, metrics were forecast with at least deviation score of 2.6 for this service. Lowest deviation score, 1.9, belongs to Eucalyptus. This demonstrates the outcomes which are autonomous of training data and the size of SOA based services to be evaluated.

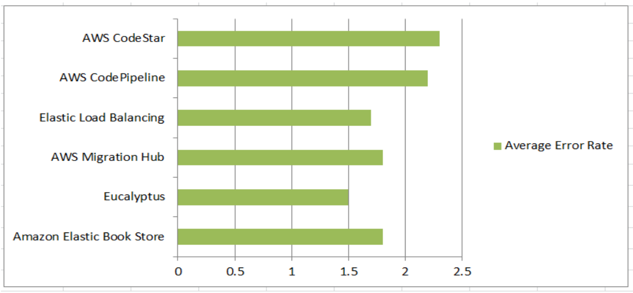

Fig. 2 Average error rate on various SOA based services

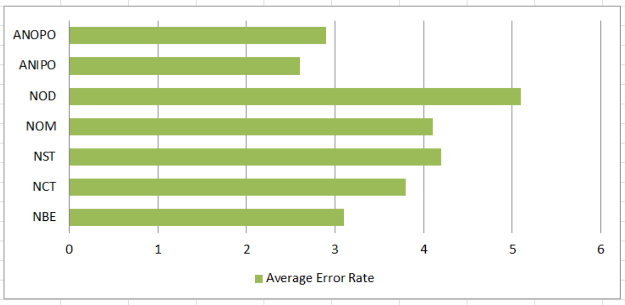

Fig. 3 Average error rate per metric on various SOA based services

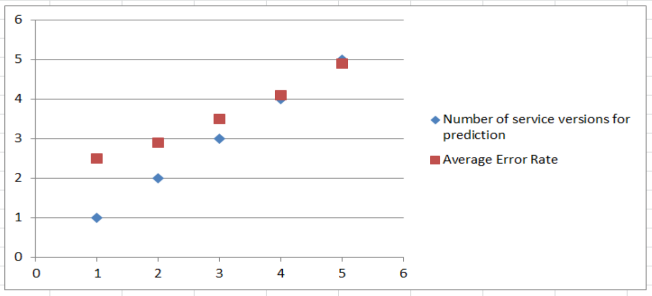

Fig. 4 Average error rate of various metrics on various SOA services based in each forecast step

Outcomes of the average error rate by measure are displayed in more detail in Figure 3. The findings unequivocally prove that our findings are unrelated to the kind of statistic to be predicted. The error rate, however, varies depending on the range of each statistic. The number of operations per service, for instance, is predicted to have the highest error rate because of its high variability and wider range than the other metrics. Figure 4 shows how well our algorithm can forecast the metrics value not just for the upcoming issue also of following five issues. With the exception of AWS CodeStar, which was not taken into account owing to the small number of releases, the findings obtained on various Web services actually demonstrate error rate for fifth forthcoming issue is extremely low with a score of lower than 5.

In response to 1st study RQ, the proposed method can accurately forecast how Web service metrics will change over time.

Outcomes for RQ2. Figures 5, 6, and 7 shows the outcomes. Fig. 5 shows majority of anticipated quality problems (Web service antipatterns) for upcoming release. With an average precision and recall that were more than eighty two and eighty five percentage respectively, the proposed forecast method correctly identified Web service antipatterns on the various services. The accuracy is higher than for the other systems with more than 88% for Elastic Load Balancing service as well as Amazon Elastic Book Store.

These systems' smaller size and lower concentration of predictable antipatterns can be used to explain why. The precision for Elastic Load Balancing Service is equally strong, meaning that major projected antipatterns are true. It demonstrates that the results of precision analysis are unaffected by magnitude of the SOA based services under evaluation. Although the precision for AWS Code Pipeline is lowest (eighty one percentage), it is still respectable. There are many unclear services in AWS Code Pipeline that are challenging to identify with metrics.

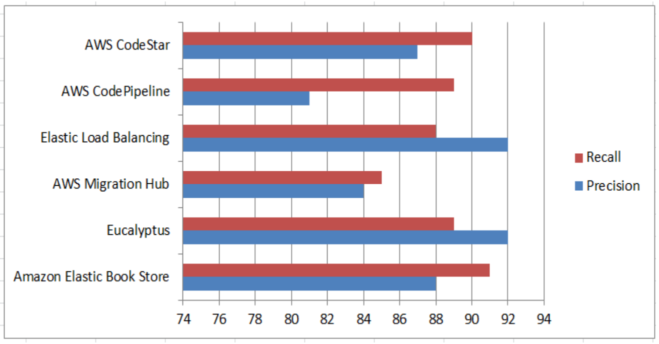

Fig. 5 Average precision and recall of forecast antipatterns on various SOA based services

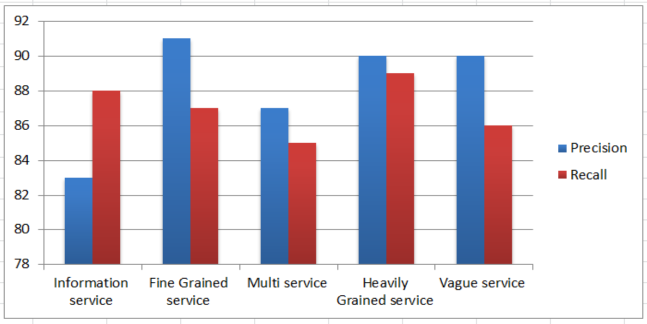

Fig. 6 Average precision and recall per antipattern type on various SOA based services

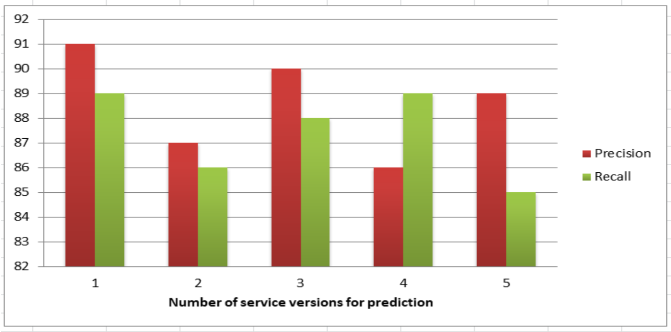

Fig.7 Average precision and recall on SOA based services per prediction step

Regarding the recall, the same observations hold true. For 6 SOA based services, the average recall was more than eighty eight percentage. More than 90% of the time, Eucalyptus and Elastic Load Balancing have superior accuracy to rest of methods. The fact that certain systems use more training data than others can be used to explain this. Since the precision for Elastic Load Balancing Service is also excellent (about ninety two percentage), the quality of prediction results was not significantly impacted by the number of the training data. Output accuracy and recall scores are in disagreement, which is an interesting finding given that the services with maximum accuracy ratings also had lowermost recall. These ratings are suitable for all SOA based services, though.

A significant advantage of the proposed method is its capacity to forecast quality difficulties not just for the upcoming release as sown in Fig. 7 but also for following five issues. In reality, acquired data unmistakably demonstrate that accuracy and recall, with an average higher than 88%, are still more for all SOA based services when forecasting quality issues for fifth imminent issue. Because there weren't many releases available, we didn't take the AWS CodeStar into account while evaluating it.

In conclusion, it is evident from the data that the proposed approach accurately forecasts problems with SOA based service quality.

Results for RQ3. The research employed a post-study questionnaire to get participant feedback on their use of our prediction tool and research outcomes in order to address RQ3. Participants were asked to score their agreement with below statements on a Likert scale with a range of one (total divergence) to five (full agreement):

• The anticipated metrics values can be used to calculate the cost and risk of utilizing a particular SOA based service, and they may also assist programmer in choosing good service for the needs.

• Anticipated quality problems could assist managers and developers in better planning maintenance tasks and lowering the cost of resolving these problems.

For the first and second statements, the participants' agreement was 4.6 and 4.8, correspondingly.

It demonstrates value of the proposed work forecast findings for experiment's programmers.

The post-study questionnaire's remaining questions dealt with the advantages and disadvantages (and potential improvements) of our prediction strategy. Following is a summary of the developers' comments. The majority of participants indicate that our findings might aid service provider developers in determining the time for restructure their SOA based service applications. For instance, whenever forecast findings indicate that quality issue, like multi-service antipattern, may become much more serious after a few releases, they may want to consider performing some refactorings. The developers like our tool's capability because it enables them to find refactoring possibilities as soon as possible.

The participants thought that service-based application developers would benefit from using our technology. In reality, while choosing a Web service from a variety of possibilities, the vast number of members remark the value of steadiness and quality of SOA based services. A service's instability could have a detrimental impact on its systems in the future and could be a sign that it has numerous faults, which would explain why there have been several fresh releases. The subject also appreciated the ability to predict antipatterns since it made it simpler for the programmers to assess the quality of SOA based services in upcoming issues by counting antipatterns rather of looking at a variety of metrics.

The participants also made some recommendations for potential adjustments to our prediction methodology. Some participants think that expanding the technique by including a latest function to automatically analyze the risk, price, and profits of employing other potential SOA based services will be highly beneficial. Utilizing visualisation techniques to analyse the development of the SOA based services in order to quickly approximate the steadiness is another potential improvement.

4.3 Threats to Validity

There are four different dangers that could undermine the reliability of our tests. The sentences that follow discuss each of these.

The statistical correlation between the treatment and the result is what is considered to be important for conclusion validity. The threat that is produced by the parameter tuning of the ANNs employed in our trials can be assessed as a part of the upcoming research. Trial and error is utilised to determine the parameter values that are employed in our experiments. To create an adaptive parameter tuning technique for the proposed method, though, in which parameters are modified during implementation to deliver greatest performance, would be an intriguing angle.

Causal association between conduct and result is important to internal validity. For the purpose of identifying potential future quality problems in the upcoming releases, we applied a set of manually constructed rules [19]. Though, outcomes attained depend on rules applied, and few of quality problems that were predicted may not be significant antipatterns for the service provider's engineers to address.

Connection between theory and what is observed is construct validity. Interviews were conducted with a number of developers to assess the applicability of our prediction results. The participant variability in terms of experience could have an impact on our study's findings in terms of the selection threat.

By ensuring that each participant has roughly the same amount of web development knowledge and familiarity with Web services, we were able to address the threat of selection. We did not place a time limit on the questionnaire's completion for the fear of weariness, and we also emailed the questionnaires to the participants, giving them the necessary amount of time to finish each task.

The generalizability of our findings is referred to as external validity. Six commonly utilized Web services from various domains with varying sizes were the subjects of our research in this study. We cannot, however, claim that our findings apply to rest of SOA based services and to rest of practitioners. We need to replicate this study in the future to verify our results. Additionally, we could only use certain metrics in our analysis.

5 Conclusion and Future Work

In this study, we suggested a method for forecasting the development of Web services. In fact, determining the risk of using a certain service and contrasting its development with other potential services that might offer the same features may be crucial for users. Furthermore, anticipating changes could assist online service providers in better allocating resources and planning necessary maintenance tasks to raise quality. In this article, we suggest using artificial neural networks-based machine learning to forecast how the architecture of Web services interfaces will change over time. We gathered training data from quality metrics of prior releases from 6 Web services to validate the suggested methodology. The validation of our prediction approaches reveals that, with a very low deviation rate, the predicted metrics values for the various releases of the 6 Web services, such as number of operations, were extremely similar to the ones that were expected. Additionally, with an average precision and recall higher than 85%, the majority of the quality concerns of the examined Web service interfaces were correctly anticipated for the next releases. The poll of programmers demonstrates the value of the prediction technique for both service providers and subscribers.

Future study will involve testing our prediction methods using various indicators, Web services, and developers in order to draw conclusions about its general applicability. Additionally, we exclusively predicted the evolution of Web services in this research. By creating additional risk measures based on the anticipated metrics value, we intend to expand the strategy. We will also research the long-term effects of expected quality problems on the usability and appeal of Web services.

References

[1] Simon, H.A.: Why should machines learn? (Chap. 2). In: Michalski, R.S., Carbonell, J.G., Mitchell, T.M. (eds.) Machine Learning. Tioga, Palo Alto (1983)

[2] Gardner, M.W., Dorling, S.R.: Artificial neural networks (the multilayer perceptron). Atmos. Environ. **32**, 2627–2636 (1998)

[3] Cobourn, W., Dolcine, L., French, M., Hubbard,M.: A comparison of nonlinear regression and neural network models for ground-level ozone forecasting. J. Air Waste Manag. Assoc. **4**, 19–68(2001)

[4] Agirre-Basurko, E., Ibarra-Berastegi, G., Madariaga, I.: Regression and multilayer perceptron-based models to forecast hourly O_3 and NO_2 levels in the Bilbao area. Environ. Model Softw. **21**, 430–446 (2006)

[5] Haykin, S.: Neural Networks: A Comprehensive Foundation. Macmillan College Publishing Company, New York (1994)

[6] Ouni, A., Kessentini, M., Inoue, K.: Search-based web service antipatterns detection. In: IEEE Transactions on Services Computing, pp. 1–21. IEEE (2016, to appear)

[7] Ouni, A., Gaikovina, K.R., Kessentini, M., Inoue, K.: Web service antipatterns detection using genetic programming. In: 24th ACM Genetic and Evolutionary Computation Conference (GECCO), pp. 1351–1358 (2015)

[8] Al Shalabi, L., Shaaban, Z., Kasasbeh, B.: Data mining: a preprocessing engine. J. Comput. Sci. **2**(9), 735–739 (2006)

[9] Fokaefs, M., Mikhaiel, R., Tsantalis, N., Stroulia, E., Lau, A.: An empirical study on web service evolution. IEEE International Conference on Web Services (ICWS11), pp. 261–269. IEEE (2011)

[10] Romano, D., Pinzger, M.: Analyzing the evolution of web services using fine-grained changes. In: 19th IEEE International Conference on Web Services, ICWS, Honolulu, pp. 392–399(2012)

[11] Aversano, L., Di Penta, M., Falanga, A., Scognamiglio, R.: Visualizing the evolution of web services using formal concept analysis. In: Eighth International Workshop on Principles of Software Evolution, pp. 57–60 (2005)

[12] Xing, Z., Stroulia, E.: UMLDiff: an algorithm for object-oriented design differencing. In: Proceedings of the 20th IEEE/ACM International Conference on Automated Software Engineering (ASE 2005), pp. 54–65. ACM, New York (2005)

[13] Zarras, A.V., Vassiliadis, P., Dinos, I.: Keep calm and wait for the spike! Insights on the evolution of amazon services. In: Proceedings of the 28th International Conference on Advanced Information Systems Engineering (CAiSE), (2016)

[14] Rodriguez, J.M., Crasso, M., Mateos, C., Zunino, A.: Best practices for describing, consuming, and discovering web services: a comprehensive toolset. Softw. Pract.Experience **43**(6), 613–639 (2013)

[15] Rodriguez, J.M., Crasso, M., Zunino, A., Campo, M.: Automatically detecting opportunities for web service descriptions improvement. In: Cellary, W., Estevez, E. (eds.) Software Services for e-World. IFIP AICT, vol. 341, pp. 139–150. Springer, Heidelberg (2010)

[16] Mateos, C., Rodriguez, J.M., Zunino, A.: A tool to improve code-first web services discoverability through text mining techniques. Softw. Pract. Experience **45**(7), 925–948 (2015)

[17] Rotem-Gal-Oz, A., Bruno, E., Dahan, U.: SOA Patterns, pp. 38–62. Manning Publications,

Greenwich (2012)

[18] Kral, J., Zemlicka, M.: Popular SOA antipatterns.In: Future Computing, Service Computation, Cognitive, Adaptive, Content, Patterns, pp. 271–276. IEEE (2009)

[19] Palma, F., Moha, N., Tremblay, G., Guéhéneuc, Y.-G.: Specification and detection of SOA antipatterns in web services. In: Avgeriou, P., Zdun, U. (eds.) ECSA 2014. LNCS, vol. 8627, pp. 58–73. Springer, Heidelberg (2014)

[20] Moha, N., Palma, F., Nayrolles, M., Conseil, B.J., Guéhéneuc, Y.-G., Baudry, B., Jézéquel, J.-M.: Specification and detection of SOA antipatterns. In: Liu, C., Ludwig, H., Toumani, F., Yu, Q. (eds.) Service Oriented Computing. LNCS, vol. 7636, pp. 1–16. Springer, Heidelberg(2012)

[21] Oldham, N., Thomas, C., Sheth, A.P., Verma, K.: METEOR-S web service annotation framework with machine learning classification. In: Cardoso, J., Sheth, A.P. (eds.) SWSWPC 2004. LNCS, vol. 3387, pp. 137–146. Springer, Heidelberg (2005)

[22] Klusch, M., Kapahnke, P., Zinnikus, I.: SAWSDL-MX2: a machine-learning approach for integrating semantic web service matchmaking variants. IEEE International Conference on Web Services, ICWS 2009, pp. 275–288. IEEE (2009)