Prediction of Web Service Anti-patterns Using Machine Learning Framework

Paper category: Preliminary

Sahithi Tummalapalli[1], Lov Kumar[2],Lalita Bhanu Murthy[3]
BITS Pilani Hyderabad, India[1,2,3]
(p20170433[1], lovkumar[2], bhanu[3])@hyderabad.bitspilani.ac.in

ABSTRACT

Service-Oriented Architecture(SOA) can be implemented via web services which are a product framework that reinforces machine to machine communication over a system. Web services suffer from some setbacks due to bad programming practices design and implementation. These are called "Anti-patterns". Anti-Patterns are counter-productive and poor design practice choices that are conjectured to make software systems harder to maintain. Web service anti-patterns lead to evolution and maintenance problems in software systems. The prediction of anti-patterns in the early stages helps the developers and testers in fixing the design issues. This stimulates the researchers to focus on the detection of web service anti-patterns, to help the unskilled software developers and designers to improve the quality of services being provided.

1. INTRODUCTION

1.1 Origin of the Proposal:

- *Anti-Patterns are counter-productive and poor design practice choices that are conjectured to make software systems harder to maintain.*
- *Classes having Anti-Patterns have approximately three times higher probability of change proneness with respect to classes having no Anti-Patterns [14].*
- *Classes having Anti-Patterns have approximately nine times higher probability of fault proneness with respect to classes having no Anti-Patterns [14].*
- *Fixing bugs is time-consuming: Median repair time of a bug for ArgoUML project = 190 days, PostgreSQL = 200 days, 1/2 of all fixed bugs in Mozilla = took more than 29 days [5].*
- *Anti-patterns have an effect on the development and maintenance of software systems and a study has shown that maintenance of software involves 80% to 90% of the total budget in its life cycle [6]*
- *It is observed from experimental analysis that early detection of code smell affected classes can help to reduce 17.64−20% efforts [6].*

1.2 Research Background:

Service Oriented Architecture(SOA) describes a standard method for requesting services from distributed components and provides the ability to look at business systems as an entire set of services. The primary focus of SOA is on the characteristics of service interface and predictable service behavior[9]. SOA provides methods for system development and integration where systems group functionality around business processes and package these as inter-operable services[4]. It has a transmuting way of dealing with circulated programming and employs mechanism that allows IT systems to work together cohesively within one platform. This circulated programming encourages the inexactly coupled mix and flexibility to change. SOA's can be implemented via web services which is a product framework that reinforces machine to machine communication over a system. It has an interface characterized in a machine processible organization specially WSDL(Web Service Description Language). Web services suffers from some setbacks due to bad programming practises, design and implementation [10]. These are called "Anti-patterns". Anti-Patterns are counter-productive and poor design practice choices that are conjectured to make software systems harder to maintain. Web service anti-patterns leads to evolution and maintenance problems in software systems. The prediction of anti-patterns in early stages helps the developers and testers in fixing the design issues. This may help them to use the available resources effectively. It has been reported by various studies that the timely detection and correction of anti-patterns enhances the performance and quality of the software systems. This stimulates the researchers to focus on the detection of web service anti-patterns, to help the unskilled software developers and designers to improve the quality of services being provided. There are several web-services anti-patterns [10][12]. The five web-services anti-patterns which we consider for our research work are: GOWS : God object Web service, FGWS : Fine grained Web service, DWS : Data Web service, CWS : Chatty Web service, AWS: Ambiguous Web service. The definition of the anti-patterns can be found in the literature. Previous research shows that anti-patterns in web-services is a serious and important problem. Most of the research work till now is focused on the detection

of anti-patterns from object oriented software applications. The research work in the detection of anti-patterns in Service Oriented Architecture(SOA) is still in early stages and there are lack of approaches for automatically detecting web-service anti-patterns from their source code and Web Service Definition Language (WSDL) documents [10][12].

Our motivation is to investigate the application of machine learning based techniques for building predictive models by considering source code metrics, WSDL metrics and text metrics individually as features for the task of detecting web-services anti-patterns.

2. LITERATURE SURVEY

There is an extensive research done, in the detection of object oriented anti-patterns and code smells. However, techniques for detecting anti-patterns in object oriented systems and techniques for detecting code smells are not applicable in association with web services, as they have different level of granularity i.e service versus class levels, and different technologies and metrics when compared to object oriented services. Unlike object oriented software systems, web service source code is not open source i.e, clients can access only web service interfaces which are publicly available, from which only the skeleton of the code can be automatically generated. This makes the detection of SOA anti-patterns more challenging. The problems of SOA anti-patterns is addressed only by a few works. Dudney et al.[3] wrote the first book on anti-patterns which provides the informal definitions about the set of web service anti-patterns. A range of SOA anti-patterns features were described by Rotem-Gal-Oz[16]. Furthermore, a variety of research work has been done, which addressed the issue of web service anti-pattern detection which is tabulated in Table 1.

3. GAPS IN EXISTING RESEARCH

Most of the research work till now is focused on the detection of anti-patterns from object oriented software applications. The research work in the detection of anti-patterns in Service Oriented Architecture(SOA) is still in early stages. Only a few approaches are proposed for the SOA anti-pattern detection and specification[10][11]. To the best of our knowledge, different sets of metrics along with some static and dynamic analysis methods are used for web services anti-pattern detection in various researches. Furthermore, there are is a limitation on the availability of the conventional prototypes for estimating and rating results of SOA anti-patterns detection techniques. So, our goal is to build an efficient predictive model for the automatic detection of anti-patterns in web services considering source code metrics, WSDL metrics and text/word metrics individually as features and to oversee an exhaustive exploration on the real-world dataset and research the performance of machine learning algorithms, feature selection techniques and data sampling techniques for predicting different types of web service anti-patterns.

- The anti-pattern detection strategies in object oriented software applications cannot be used for anti-pattern detection in SOA, as they have different levels of granularity i.e services versus class levels.
- From an extensive literature study, it is perceived that, several approaches have been used for distinguishing various anti-pattern using different set of source code metrics.
- Most of the procedures proposed did not refer to the reliability of anti-pattern detection using the associated technique which is crucial to know the fruitfulness of the strategy proposed [10].
- Several approaches for the detection of web service anti-patterns have used source-code parsing techniques, which includes the statistical clustering of data like measuring switch statements, counting lines of code etc [15].

4. OBJECTIVES

The project aims to investigate source code, WSDL file, and text mining based approaches to analyze historical information databases to discover intriguing patterns and knowledge which can be used to assist developers in the process of identifying components affected by Anti-Patterns. The proposed research problem can be formulated in the form of a question "How Should I Detect and Fix Anti-Patterns?" which needs to be answered every-time when a Anti-Pattern report is submitted. The question can be further decomposed into following sub-problems and questions.

- Whether the source code, WSDL, word metrics predict code Anti-Patterns ?
- What is the variation in performance parameters of developed models using different set of metrics?
- What are the ramifications of Anti-Patterns on fault-proneness and change-proneness?
- What is the proficiency of dissimilar feature selection techniques to predict web service anti-pattern?
- What is the proficiency of disparate Classification techniques to predict web service anti-pattern?
- What is the proficiency of assorted data balancing strategies over original data to predict web service anti-pattern?

Research Aim:

- To build an efficient predictive model for the detection of anti-patterns in web services.
- To investigate the relationship between the occurrence of anti-patterns and source code metrics, word metrics and WSDL metrics.
- To analyze the application of different aggregation techniques to aggregate class level metrics into system level metrics for the task of web-services anti-pattern detection.
- To explore the correlation between the diversified set of source code metrics, WSDL metrics, aggregation techniques with presence of common web-services anti-patterns.
- To organize an exhaustive factual analysis on real-world dataset and investigate the performance of the machine learning algorithms, feature selection techniques, and data sampling techniques for predicting distinguished web-services anti-patterns.

Table 1: Published work about SOA anti-pattern detection

Authors	Method Used	Technique	Anti-Patterns Detected	Metrics Used	Year
Fatima Sabir et al. [17]	Scalable approach for the detection of anti-patterns using anti-pattern detection engine	Rules generated by Enterprise Architect data model	GOWS, DWS, FGWS, AN, Dup-WS, LCO, RPT, CWS, CRUD, LGWS	Data model of Enterprise architect	2017
Wang, H. et al. [19]	Web Service Refactoring opportunities as Multi-objective problems Identification	Multi-Objective Genetic Programming(MOGP)	MRPC, CRUDYI, DS, AN, FG, GOWS.	Web service interface metrics and Web service code metrics	2016
YUGOV, A [20]	Detection of anti-patterns by using graph models	Metric based approach	GOWS	Incoming call rate, Outcoming call rate, response time, number of service connections, cohesion with other services.	2016
Wang, H.et al. [18]	Web Service Evolution prediction	Artificial Neural Network(ANN) to predict anti-patterns in future releases	DS, MS, NS, CS		2016
Ouni, A. et al. [10]	Cooperative parallel evolutionary algorithm(P-EA) for detecting anti-patterns in web-services	Parallel Evolutionary Algorithms	MRPC, CRUDYI, DS, AN, FGWS, GOWS	Interface-level (WSDL) metrics, Dynamic metrics, Code-level metrics.	2015
Coscia et al. [2]	Java to WSDL Mapper	Meta Programming and Text Mining	EDM, RPT, WET, AN, UFI, IC, ISM, LCOP	Object-oriented metrics from the code-implementing services	2014
Palma, F. et al. [12]	Service Oriented Detection for Anti-patterns in Web services (SODA-W) , an approach supported by a framework for specifying and detecting anti-patterns in WSs	Static and dynamic analysis for source code metrics	GOWS, FGWS, AN, CS, RPT, LCO	Static and Dynamic metrics	2014
Palma, F. et al. [13]	Specification and Detection of anti-patterns using a set of metrics by using a rule-based approach	SOFA	MS, DS, NS	Static and Dynamic Metrics	2013
Nayrolles et al. [8]	Service Oriented Mining for Anti-pattern Detection(SOMAD)-detection by mining execution traces.	Association Rule Mining	MS, DS, CS, Kt, BNS, SC	NMA, NDP, NM, COH, CID, IC, OC, TC	2013
Mateos et al. [7]	Detecting WSDL based services using EasySOC tool.	Text Mining, Component based software engineering and Machine learning	WSDL based Services	Object-oriented metrics from the code-implementing services	2013
Coscia et al. [1]	correlation analysis between source code metrics and WSDL implementation code	Detection of anti-patterns using statistical analysis	LCOP, ISM, UFI, IC, AN,RPT, WET, EDM	Object-Oriented metrics	2011

Applying data mining concepts on software repositories containing bugs information, change information, source code, mailing list, requirement documents etc. is a transpiring field that has acquired remarkable research interest in recent times. Different semi-automated tools have been proposed to assist a practitioner in decision making and automating software engineering tasks. The objective of our research is to identify different types of anti-patterns by investigating source code, WSDL file and text based metrics. In particular, our inspiration is to explore the use of AI and machine learning based systems for building prescient models from source code metrics as features for the undertaking of web service anti-pattern detection.

5. METHODOLOGY

Figure 1 shows the research methodology used for developing prediction models to predict web-service Anti-Patterns. In this project, we will use publicly available open source projects from SourceForge for this experiments. For each project, three kinds of metrics sets are used to measure internal structure of the project i.e., source code metrics, WSDL metrics and word metrics.

- Source code metrics: It is intuitive that the more complex a file is, the more Anti-Pattern will happen in it. Thus, the source code metrics are used as features to find files start to Anti-Pattern.
- WSDL Metrics: These metrics are used to compute the web service interfaces complexity and complexity of WSDL file. As we know the WSDL file having high interface complexity are more likely to have Anti-Pattern as compared to other WSDL files.
- Word Metrics: It is intuitive that the frequencies of words in the source code has more correlation with fault and change-proness of classes due to Anti-Pattern.

The subsequent step consists of applying the feature selection techniques i.e., feature ranking and subset selection techniques on the dataset produced, to recover the metric set which is reasonable for building the prescient model. At that point, the dataset is standardized using the Min-Max Normalization technique and the data sampling techniques are utilized to manage the class imbalance problem. Further, some advanced machine learning and ensemble techniques are applied with the cross-validation approach for model development and the outcomes obtained are used for performance analysis.

6. WORK FLOW

In this section, we are specifying the methodology to be followed for the **"Automatic detection of anti-patterns"**:

- **Phase 1**:
  - Literature Survey on the detection of object-oriented anti-patterns.
  - Literature Survey on the detection of anti-patterns in Service Oriented Architecture(SOA).
  - Literature Survey on Machine Learning Algorithms.
- **Phase 2**:
  - Finding the association between the occurrence of anti-patterns and source code metrics, WSDL metrics and text based metrics.
  - Investigating the application of different aggregation techniques to aggregate class level metrics into system level metrics for undertaking web-services anti-pattern detection.

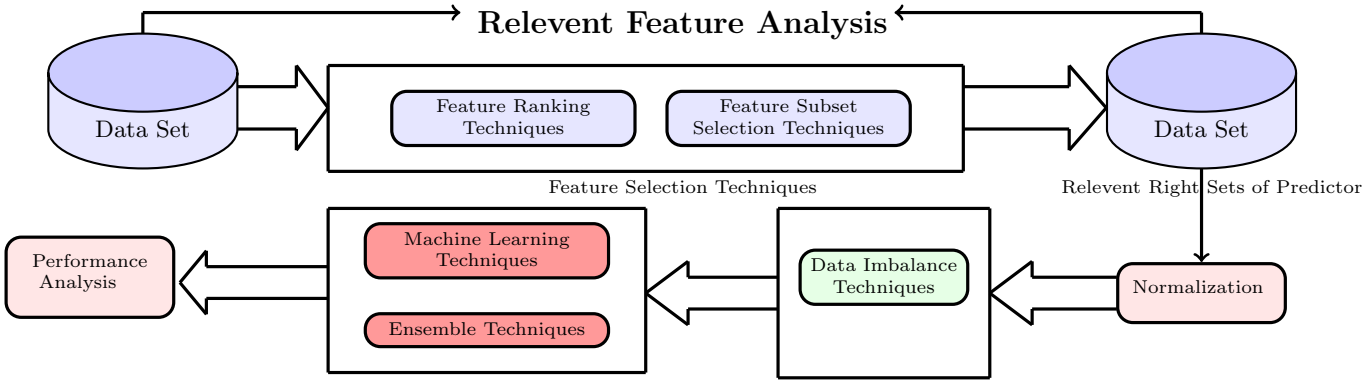

Figure 1: Framework of proposed work

- **Phase 3**: Performing relevant feature selection techniques and investigating the impact of different data balancing techniques to manage the class imbalanced problem.
- **Phase 4**: Building a predictive model for the detection of anti-patterns using Machine Learning Framework.
- **Phase 5**: Building a predictive model for the detection of anti-patterns using ensemble techniques.

7. REFERENCES

[1] J. L. O. Coscia, C. Mateos, M. Crasso, and A. Zunino. Avoiding wsdl bad practices in code-first web services. In *Proceedings of the 12th Argentine Symposium on Software Engineering (ASSE2011)-40th JAIIO*, pages 1–12, 2011.

[2] J. L. O. Coscia, C. Mateos, M. Crasso, and A. Zunino. Refactoring code-first web services for early avoiding wsdl anti-patterns: Approach and comprehensive assessment. *Science of Computer Programming*, 89:374–407, 2014.

[3] B. Dudney, S. Asbury, J. K. Krozak, and K. Wittkopf. *J2EE antipatterns*. John Wiley & Sons, 2003.

[4] Y. Kun, W. Xiao-Ling, and Z. Ao-Ying. Underlying techniques for web services: A survey. In *Journal of software*. Citeseer, 2004.

[5] C. Le Goues, M. Dewey-Vogt, S. Forrest, and W. Weimer. A systematic study of automated program repair: Fixing 55 out of 105 bugs for 8 each. In *Software Engineering (ICSE), 2012 34th International Conference on*, pages 3–13. IEEE, 2012.

[6] H. Liu, Z. Ma, W. Shao, Z. Niu, et al. Schedule of bad smell detection and resolution: A new way to save effort. *IEEE Transactions on Software Engineering*, 38(1):220–235, 2012.

[7] C. Mateos, M. Crasso, A. Zunino, and J. O. Coscia. Revising wsdl documents: Why and how-part ii. *IEEE Internet Computing*, 17(5):46–53, 2013.

[8] M. Nayrolles, N. Moha, and P. Valtchev. Improving soa antipatterns detection in service based systems by mining execution traces. In *Reverse Engineering (WCRE), 2013 20th Working Conference on*, pages 321–330. IEEE, 2013.

[9] E. Newcomer and G. Lomow. *Understanding SOA with Web services*. Addison-Wesley, 2005.

[10] A. Ouni, R. Gaikovina Kula, M. Kessentini, and K. Inoue. Web service antipatterns detection using genetic programming. In *Proceedings of the 2015 Annual Conference on Genetic and Evolutionary Computation*, pages 1351–1358. ACM, 2015.

[11] A. Ouni, M. Kessentini, K. Inoue, and M. O. Cinnéide. Search-based web service antipatterns detection. *IEEE Transactions on Services Computing*, 2015.

[12] F. Palma, N. Moha, G. Tremblay, and Y.-G. Guéhéneuc. Specification and detection of soa antipatterns in web services. In *European Conference on Software Architecture*, pages 58–73. Springer, 2014.

[13] F. Palma, M. Nayrolles, N. Moha, Y.-G. Guéhéneuc, B. Baudry, and J.-M. Jézéquel. Soa antipatterns: An approach for their specification and detection. *International Journal of Cooperative Information Systems*, 22(04):1341004, 2013.

[14] F. Palomba, G. Bavota, M. Di Penta, F. Fasano, R. Oliveto, and A. De Lucia. On the diffuseness and the impact on maintainability of code smells: a large scale empirical investigation. *Empirical Software Engineering*, 23(3):1188–1221, 2018.

[15] G. Rasool and Z. Arshad. A lightweight approach for detection of code smells. *Arabian Journal for Science and Engineering*, 42(2):483–506, 2017.

[16] A. Rotem-Gal-Oz, E. Bruno, and U. Dahan. *SOA patterns*. Manning, 2012.

[17] F. Sabir, G. Rasool, and M. Yousaf. A lightweight approach for specification and detection of soap anti-patterns. *INTERNATIONAL JOURNAL OF ADVANCED COMPUTER SCIENCE AND APPLICATIONS*, 8(5):455–467, 2017.

[18] H. Wang, M. Kessentini, and A. Ouni. Prediction of web services evolution. In *International Conference on Service-Oriented Computing*, pages 282–297. Springer, 2016.

[19] H. Wang, A. Ouni, M. Kessentini, B. Maxim, and W. I. Grosky. Identification of web service refactoring opportunities as a multi-objective problem. In *Web Services (ICWS), 2016 IEEE International Conference on*, pages 586–593. IEEE, 2016.

[20] A. Yugov. Approach to anti-pattern detection in service-oriented software systems. , 28(2), 2016.