Improving Web Services Design Quality Via Dimensionality Reduction

Hanzhang Wang and Marouane Kessentini

Computer and Information Science Department,
University of Michigan, Dearborn, MI, USA
{wanghanz, marouane}@umich.edu

Abstract. In this paper, we propose a dimensionality reduction approach based on PCA-NSGAII to address the Web services modularization problem. Our approach aims at finding the best reduced set of objectives (e.g. quality metrics) that can generate near optimal modularization solutions to fix quality issues in Web services interface. The algorithm starts with a large number of Web service quality metrics as objectives that are reduced based on the correlation between them. This correlation is identified during the execution of the multi-objective algorithm by mining the execution traces of the generated solutions and their evaluations. We evaluated our approach on a set of 22 real world Web services, provided by Amazon and Yahoo. Statistical analysis of our experiments shows that our dimensionality reduction Web services interface modularization approach performed significantly better than the state-of-the-art modularization techniques in terms of generating well-designed Web services interface for users.

1 Introduction

The evolution of Web services may have a negative impact on the design quality of the interface by concatenating many non-cohesive operations that are semantically unrelated, and thus make it unnecessarily complex for users to find relevant operations to be used in their services-based systems. An example of well-known interface design defect is the God object Web service (GOWS) [6,21,24,19,12,5,14,13,15,18,17] which implements many operations related to different business and technical abstractions in a single service interface leading to low cohesion of its operations and high unavailability to end users because it is over-loaded. Indeed, the choice of how operations should be exposed through a service interface can have an impact on the performance, popularity and reusability of the service and it is not a trivial task [16,10,11]. On one hand, Web services interface exposing a high number of operations allow their clients to invoke their interfaces many times which significantly deteriorate the service performance. On the other hand, aggregating several operations of an interface into one large operation will reduce the reusability of the service.

In this work, we start from the hypothesis that there may be correlations among any two or more objectives (e.g. quality metrics) that are used to evaluate Web service modularization solutions. Our approach, based on the PCA-NSGA-II methodology [3,23], aims at finding the best and reduced set of objective that

represents the quality metrics of interest to the domain expert. A regular multi-objective NSGA-II algorithm [4,9,2] with an initial set of exhaustive metrics is executed for a number of iterations then a PCA component analyzes the correlation between the different objectives using the execution traces. The number of objectives maybe reduced during the next iterations based on the PCA results. The process is repeated several times until a maximum number of iterations is reached to generate a set of non-dominated Web services modularization solutions.

We evaluated our approach on a set of 22 real-world Web services, provided by Amazon and Yahoo. Statistical analysis of our experiments shows that our dimensionality reduction reduced significantly the number of objectives on several case studies to a minimum of 4 objectives . It also generates a smaller number of non-dominated solutions and lower execution time comparing to the use of a regular multi-objective algorithm based on NSGA-II [4]. The obtained results provide also evidence to support the claim that our proposal is more efficient, on average, than existing Web services modularization techniques, not based on heuristic search [1,22]. The paper also evaluates the relevance and usefulness of the suggested interface design improvements for web services user.

2 A Dimensionality Reduction Approach for Web Services Remodularization

The general structure of the proposed approach is described in Figure 2. The approach takes as inputs a set of quality metrics, several Web services refactoring types, and a Web service to refactor. The first component consists of a regular execution of NSGA-II during a number of iterations. During this phase, NSGA-II [4] will try to find the non-dominated solutions balancing the initial set containing all the objectives such as improving the quality metrics of the service (Table 1) and minimizing the number of refactorings in the proposed solutions.

After a number of iterations, the second component of the algorithm is executed to analyze the execution traces of the first component (solutions and their evaluations), using PCA [8], to check the correlation between the different objectives. When a correlation between two or more objectives is detected, only one of them is selected for future iterations of the first component. Then, the first component is executed again with the new objective set.

The whole process of these two components continue until a maximum number of iterations is reached. A set of non-dominated refacotoring solutions are proposed to the users with the reduced objectives set to select the best Web service refactorings sequence based on his or her preferences.

3 Experiments

3.1 Research Questions

We designed our experiments to address the following research questions:

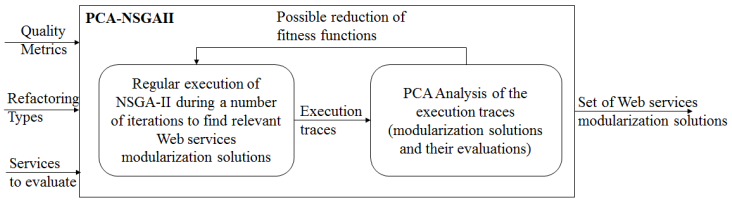

Fig. 1: The proposed approach.

- **RQ1.**: To what extent can the proposed dimensionality reduction approach recommends useful Web service refactorings?
- **RQ2.**: To what extent does the proposed dimensionality reduction approach reduce the number of objectives while recommending useful refactorings?
- **RQ3.**: How does the proposed dimensionality reduction approach perform compared to other existing Web services modularization techniques not based on computational search [1,22]?

To answer **RQ1.**, we considered both automatic and manual validations to evaluate the usefulness of the proposed Web service refactorings. For the automatic validation we compared the proposed Web service refactorings with the expected ones. The expected refactorings are suggested by users (e.g. subjects of our study) to fix existing Web service design defects as detailed later.

$$RE_{recall} = \frac{\mid suggested\ Web\ service\ refactorings \cap expected\ Web\ service\ refactorings \mid}{\mid expected\ Web\ service\ refactorings \mid} \in [0,1] \quad (1)$$

$$RE_{precision} = \frac{\mid suggested\ Web\ service\ refactorings \cap expected\ Web\ service\ refactorings \mid}{\mid suggested\ refactorings \mid} \in [0,1] \quad (2)$$

For the manual validation, we asked groups of potential users of our tool to manually evaluate whether the suggested refactorings are feasible and efficient at improving the services quality and achieving their maintainability objectives. We define the metric Manual Correctness (MC) that corresponds to the number of meaningful refactorings divided by the total number of suggested refactorings. MC is given by the following equation:

$$MC_{manualcorrectness} = \frac{\mid relevant\ Web\ service\ refactorings \mid}{\mid suggested\ Web\ service\ refactorings \mid} \in [0,1] \quad (3)$$

We have also evaluated the ability of our approach to fix design defects, detailed in Section 2, using the measure NF that corresponds to the number of fixed defects divided by the total number of defects. The defects are detected using a set of rules defined in our previous work [20].

To answer **RQ2**, we compared the number of objectives (*NOB*), precision, recall and manual correctness of our approach to a regular multi-objective algorithm (NSGAII) using the same fitness functions adaptation.

To answer **RQ3**, We compared our results with a recent state-of-the art approaches by [1,22]. Athanasopoulos et al. proposed a Web service refactoring approach based on a greedy algorithm to refactor and split Web service interfaces based on different cohesion measures. Ouni et al. proposed a graph decomposition approach for Web services remodularization using coupling and cohesion metrics.

3.2 Experimental Setup

To answer all the above research questions, we conducted our experiment on a benchmark of 22 real-world services provided by Amazon[1] and Yahoo[2]. We selected services with interfaces exposing at least 10 operations. We chose these Web services because their WSDL interfaces are publicly available, and they were previously studied in the literature [1] [7]. Table 1 presents our used benchmark.

Our evaluation involved 14 independent volunteer participants including 6 industrial developers and 8 graduate students. In particular, 3 senior developers from *Browser Kings*[3], 3 developers from *Accunet Web Services*[4], 3 MSc and 5 PhD candidates in Software Engineering. We first gathered information about the participant's background. All participants are familiar with service-oriented development and SOAP Web services with an experience ranging from 4 to 9 years. The participants were unaware of the techniques to be evaluated neither the particular research questions, in order to guarantee that there will be no bias in their judgment.

3.3 Results

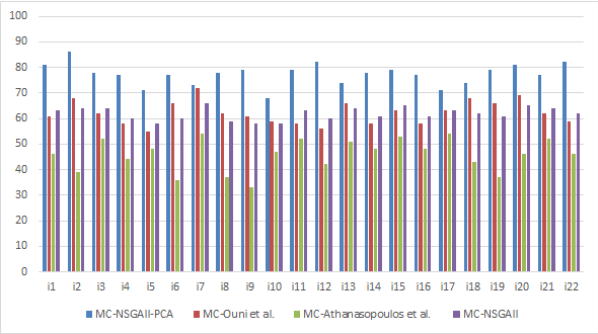

Fig. 2: Median manual correctness value over 30 runs on all the Web services using the different techniques with a 95% confidence level ($\alpha < 5\%$).

[1] `http://aws.amazon.com/`
[2] `developer.searchmarketing.yahoo.com/docs/V6/reference/`
[3] `http://www.browserkings.com`
[4] `http://www.accunet.us`

Table 1: Amazon and Yahoo benchmark overview.

Service interface	Provider
AutoScalingPortType	Amazon
MechanicalTurkRequesterPortType	Amazon
AmazonFPSPorttype	Amazon
AmazonRDSv2PortType	Amazon
AmazonVPCPortType	Amazon
AmazonFWSInboundPortType	Amazon
AmazonS3	Amazon
AmazonSNSPortType	Amazon
ElasticLoadBalancingPortType	Amazon
MessageQueue	Amazon
AmazonEC2PortType	Amazon
KeywordService	Yahoo
AdGroupService	Yahoo
UserManagementService	Yahoo
TargetingService	Yahoo
AccountService	Yahoo
AdService	Yahoo
CompaignService	Yahoo
BasicReportService	Yahoo
TargetingConverterService	Yahoo
ExcludedWordsService	Yahoo
GeographicalDictionaryService	Yahoo

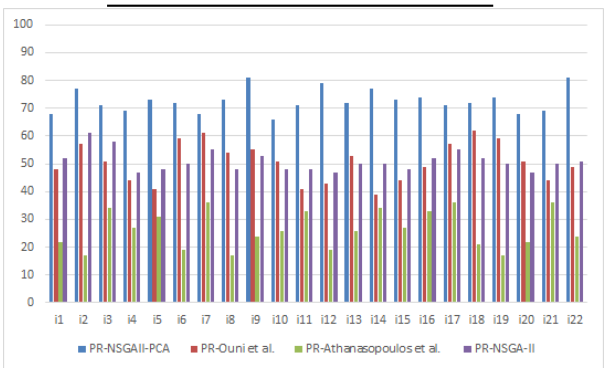

Fig. 3: Median precision value over 30 runs on all the Web services using the different techniques with a 95% confidence level ($\alpha < 5\%$).

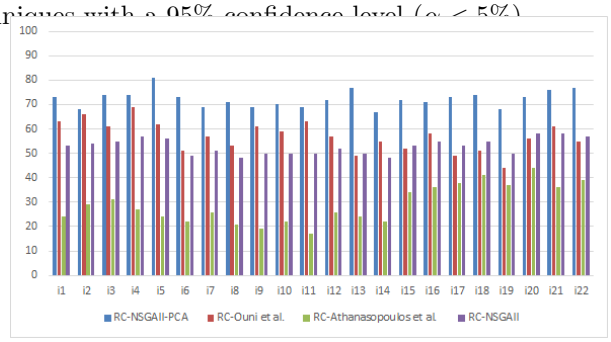

Fig. 4: Median recall value over 30 runs on all the Web services using the different techniques with a 95% confidence level ($\alpha < 5\%$).

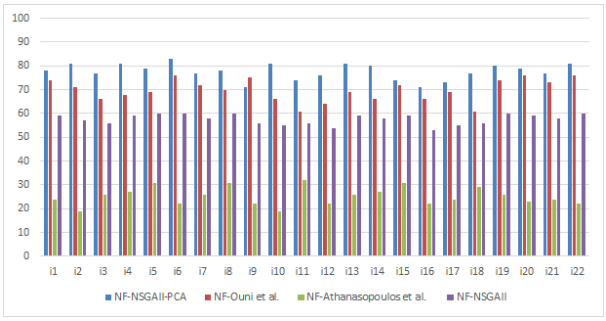

Fig. 5: Median number of fixed design defects value over 30 runs on all the Web services using the different techniques with a 95% confidence level ($\alpha < 5\%$).

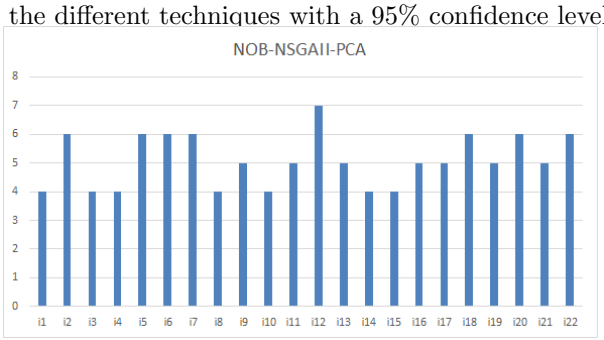

Fig. 6: Median number of objectives value over 30 runs on all the Web services using NSGAII-PCA.

We reported the results of our empirical qualitative evaluation in Figure 4 (MC). As reported in Figure 4, most of the Web services modularization solutions recommended by our approach were correct and approved by developers. On average, for the different Web services, 78% of the created port types and applied changes to the initial design are considered as correct, improve the quality, and are found to be useful by the software developers of our experiments. The highest MC score is 84% and was achieved for the Web service GeographicalDictionary, while the lowest score was 67% for AmazonVPCPortType. Thus, this finding indicates that the results are independent of the size of the Web services and the number of recommended changes to the initial design.

Since the manual correctness MC metric just evaluates the correctness and not the relevance of the recommended solutions, we also compared the proposed modularization changes with some expected ones defined manually by the different groups for the different Web services. Figures 5 and 6 summarize our findings. We found that a considerable number of proposed port types, with an average of more than 76% in terms of precision and recall, were already created by the users manually (expected port types). The recall scores are higher than precision ones since we found that the port types sug-gested manually by developers could be further decomposed, if necessary. This was confirmed by the qualitative evaluation (MC). In addition, we found that the slight deviation with

the expected design is not related to incorrect changes but to the fact that the developers have different scenarios/contexts in using the different operations.

We evaluated also the ability of our approach to fix several types of design defects and to improve the service interface design quality as described in Figure 7 that depicts the percentage of fixed defects (NF). It is higher than 77% on all the 22 Web services, which is an acceptable score since developers may reject or modify some design changes that fix some de-fects because they do not consider some of them as very important (their goal is not to fix all design defects in the Web service interface) or because they wanted to focus on improving the cohesion and minimize coupling. Some Web service interfaces, such as AmazonFWSInboundPortType, have a higher percentage of fixed code smells with an average of more than 83%.

To summarize and answer RQ1, the experimentation results confirm that our approach helps the participants to restructure their Web service interface design efficiently by finding the relevant portTypes and improve the quality of all the 22 Web services.

Results for RQ2. Figure 8 shows that our approach significantly reduced the number of objectives when executed on all the systems. The number of objectives were reduced to only four in several services. The reduced objectives may show the importance of coupling and cohesion when identifying refactoring recommendations since they were identified in all the 22 services after the reduction of objectives.The number of changes was also selected for all the services after the reduction step. Combined with the results of RQ1, it is clear that the proposed NSGAII-PCA formulation successfully reduced the number of objectives while generating useful Web services refactoring recommendations.

Results for RQ3. Figures 4,5,6 and 7 confirm the average superior performance of our approach compared to the two existing fully automated Web service modularization techniques [1,22] and also the multi-objective approach combining all the metrics together without the use of the PCA component. Figure 4 shows that our approach provides significantly higher manual correctness results (MC) than all other approaches having MC scores respectively between 48% and 64%, on average as MC scores on the different Web services. The same observation is valid for the precision and recall as described in Figures 5 and 6. The outperformance of our technique in terms of percentage of fixed defects, as described in Figure 7, can be explained by the fact that the main goal of existing studies is not to mainly fix these defects. Existing work are mainly limited to the coupling and cohesion metrisc which may not be sufficient to guide the modularization of Web services. In conclusion, our approach provides better results, on average, than all existing fully-automated Web services modularization techniques (answer to RQ3).

4 Conclusion

In this paper, we proposed a dimensionality reduction approach for multi-objective Web services remodularization that adjusts the number of considered objec-

tives during the search for near optimal solutions. The execution traces of the multi-objective algorithm are analyzed using a PCA component to find potential correlation between the objectives (e.g. quality metrics). To evaluate the effectiveness of our tool, we conducted a human study on a set of users who evaluated the tool and compared it with the state-of-the-art Web services modularization techniques. Our evaluation results provide strong evidence that our technique successfully reduced the initial set of large number of objectives/quality metrics. The results also show that our approach outperforms several of existing Web services modularization techniques, not based on heuristic search [1,22].

References

1. Athanasopoulos, D., Zarras, A.V., Miskos, G., Issarny, V.: Cohesion-Driven Decomposition of Service Interfaces Without Access to Source Code. IEEE Transactions on Services Computing 8(JUNE), 1–18 (2015)
2. Bechikh, S., Kessentini, M., Said, L.B., Ghédira, K.: Chapter four-preference incorporation in evolutionary multiobjective optimization: A survey of the state-of-the-art. Advances in Computers 98, 141–207 (2015)
3. Deb, K., Saxena, D.: Searching for Pareto-optimal solutions through dimensionality reduction for certain large-dimensional multi-objective optimization problems. In: 2006 IEEE Congress on Evolutionary Computation (CEC'2006). pp. 3353–3360. IEEE (Jul 2006)
4. Deb, K., Jain, H.: An evolutionary many-objective optimization algorithm using reference-point-based nondominated sorting approach, part i: Solving problems with box constraints. IEEE Transactions on Evolutionary Computation 18(4), 577–601 (2014)
5. ben Fadhel, A., Kessentini, M., Langer, P., Wimmer, M.: Search-based detection of high-level model changes. In: 2012 28th IEEE International Conference on Software Maintenance (ICSM). pp. 212–221. IEEE (2012)
6. Fleck, M., Troya, J., Kessentini, M., Wimmer, M., Alkhazi, B.: Model transformation modularization as a many-objective optimization problem. IEEE Transactions on Software Engineering 43(11), 1009–1032 (2017)
7. Fokaefs, M., Mikhaiel, R., Tsantalis, N., Stroulia, E., Lau, A.: An empirical study on web service evolution. In: IEEE International Conference on Web Services (ICWS). pp. 49–56 (July 2011)
8. Jackson, J.: A Users Guide to Principal Components. John Wiley and Sons, New York (1991)
9. Kalboussi, S., Bechikh, S., Kessentini, M., Said, L.B.: Preference-based many-objective evolutionary testing generates harder test cases for autonomous agents. In: International Symposium on Search Based Software Engineering. pp. 245–250. Springer (2013)
10. Kessentini, M., Bouchoucha, A., Sahraoui, H., Boukadoum, M.: Example-based sequence diagrams to colored petri nets transformation using heuristic search. Modelling Foundations and Applications pp. 156–172 (2010)
11. Kessentini, M., Langer, P., Wimmer, M.: Searching models, modeling search: On the synergies of sbse and mde. In: Proceedings of the 1st International Workshop on Combining Modelling and Search-Based Software Engineering. pp. 51–54. IEEE Press (2013)
12. Kessentini, M., Mahaouachi, R., Ghedira, K.: What you like in design use to correct bad-smells. Software Quality Journal 21(4), 551–571 (2013)
13. Kessentini, M., Sahraoui, H., Boukadoum, M.: Example-based model-transformation testing. Automated Software Engineering 18(2), 199–224 (2011)

14. Kessentini, M., Sahraoui, H., Boukadoum, M., Wimmer, M.: Search-based design defects detection by example. In: International Conference on Fundamental Approaches to Software Engineering. pp. 401–415. Springer, Berlin, Heidelberg (2011)
15. Kessentini, M., Wimmer, M., Sahraoui, H., Boukadoum, M.: Generating transformation rules from examples for behavioral models. In: Proceedings of the Second International Workshop on Behaviour Modelling: Foundation and Applications. p. 2. ACM (2010)
16. Král, J., Zemlicka, M.: Popular soa antipatterns. In: Computation World: Future Computing, Service Computation, Cognitive, Adaptive, Content, Patterns. pp. 271–276 (2009)
17. Mansoor, U., Kessentini, M., Maxim, B.R., Deb, K.: Multi-objective code-smells detection using good and bad design examples. Software Quality Journal 25(2), 529–552 (2017)
18. Mansoor, U., Kessentini, M., Wimmer, M., Deb, K.: Multi-view refactoring of class and activity diagrams using a multi-objective evolutionary algorithm. Software Quality Journal 25(2), 473–501 (2017)
19. Mkaouer, M.W., Kessentini, M., Cinnéide, M.Ó., Hayashi, S., Deb, K.: A robust multi-objective approach to balance severity and importance of refactoring opportunities. Empirical Software Engineering 22(2), 894–927 (2017)
20. Ouni, A., Kessentini, M., Inoue, K., O Cinneide, M.: Search-based web service antipatterns detection. IEEE Transactions on Services Computing PP(99) (2015)
21. Ouni, A., Kula, R.G., Kessentini, M., Ishio, T., German, D.M., Inoue, K.: Search-based software library recommendation using multi-objective optimization. Information and Software Technology 83, 55–75 (2017)
22. Ouni, A., Salem, Z., Inoue, K., Soui, M.: Sim: An automated approach to improve web service interface modularization. In: Web Services (ICWS), 2016 IEEE International Conference on. pp. 91–98. IEEE (2016)
23. Saxena, D.K., Duro, J.A., Tiwari, A., Deb, K., Zhang, Q.: Objective reduction in many-objective optimization: Linear and nonlinear algorithms. IEEE Transactions on Evolutionary Computation 17(1), 77–99 (Feb 2013)
24. Wang, H., Kessentini, M., Ouni, A.: Bi-level identification of web service defects. In: International Conference on Service-Oriented Computing. pp. 352–368. Springer, Cham (2016)