A Study on Detection of Anti-Patterns in Object-Oriented Systems

Harvinder Kaur
University Institute of Engineering & Technology
Panjab University
Chandigarh

Puneet Jai Kaur
University Institute of Engineering & Technology
Panjab University
Chandigarh

ABSTRACT
Software quality is an important issue in the development of software systems. The extent to which the software possesses a desired set of quality attributes such as testability, performance, maintainability, and manageability indicates the success of the design and the overall quality of the software system. These attributes are adversely affected by anti-patterns. These design smells, the symptoms of code smells, are introduced during software development that constrains the evolution of system by making it difficult for engineers to bring changes. Researchers and practitioners put a great effort to detect these anti-patterns to reduce costs, effort and resources. Their detection is important because it allows refactoring or removing them from systems. Consequently, it improves software quality and usability. This paper discusses various manual, semi-automated and SVM based anti-pattern detection techniques for object-oriented systems, so that researchers can get a clear and concise view about them. The limitations and advantages (over previous approaches) of some detection techniques are also compared in this paper.

Keywords
Designs Smells, Code Smells, Anti-pattern, Maintainability, Testing, Detection Techniques.

1. INTRODUCTION
Software testability is a significant attribute of software quality that facilitates testing activities and reduces testing effort and cost. The importance of software quality lies in the complexity and popularity of the software systems. Anti-patterns negatively affect the testability and hence quality of the object-oriented systems (Brown et al [1]). Anti-pattern classes require much greater testability effort than non anti-pattern classes. Their early detection and correction is necessary to comfort the development and maintenance process.

An anti-pattern is a generally used, but largely ineffective solution to a problem. We can also call them design smells. These "poor solutions" disrupt the development and maintenance activities by increasing the difficulty to understand the source code. The reasons of introducing these design smells during development include: time pressure, lack of understanding, communication, and skills. It is important to detect the occurrence of code smells because they are indicators of possible presence of design smells. In order to improve the quality and maintenance costs, anti-patterns should be detected first.

Organization of Paper: The paper is organized as: Section 2 describes the background of anti-patterns and the problems related to them. Section 3 provides a concise description of various anti-pattern detection approaches. Limitations and advantages of some currently used techniques for detecting design smells are compared. Section 4 concludes the paper.

2. BACKGROUND
2.1 Problems with Anti-pattern approaches
Dhambri et al. [2] communicate six problems which must be taken care of for an anti-pattern detection technique to be efficient. These problems are concerned with:

- Unconfined specifications of anti-patterns.
- The trends of evolution.
- Efforts required by the quality analysts to retrieve, Organize, and use the detected candidate anti-patterns.

We now revise and describe these problems.

Problem 1: Uncertainty of making the decision that whether the participant classes exist as real anti-patterns or not. Non anti-pattern classes may show all the symptoms of an anti-pattern class. The analysts' interference is significant to validate the results of detection due to ambiguous process of detection. Therefore, a detection technique should address the issue.

Problem 2: Inadequacy of listing a large set of candidates. A detection technique can be discarded due to the results containing a lot of false positives. The purpose of a detection technique is to allow the quality analyst for a manual assessment. Therefore, a list of the candidates of a detection approach should be made available.

Problem 3: Complexity of considering the opinion of a quality analyst. Given a certain set of symptoms, a design, which is interpreted as good or poor by the quality analysts', depends on their background and contextual knowledge.

Problem 4: Need to consider the detection framework. Some design fundamentals may be violated by the classes which are supposed to be "good" by many quality analysts in a given environment .An organisation or application domain can be the examples of the context.

Problem 5: Significance of using thresholds to deal with quantitative data. Different values may be explained in a different way by quality analysts', given their different backgrounds, even if there are consents on utmost values.

Problem 6: Trouble of improving and using semantic data. Some anti-pattern's symptoms like, a class accomplishing a single action, involves semantic data. An example of such an anti-pattern is Functional Decomposition. Semantic data should be taken into consideration by an automated detection technique.

3. ANTI-PATTERN DETECTION APPROACHES

3.1 Manual Techniques

Travassos et al. [3] introduced a technique to identify design smells considering manual review and reading techniques. Only manual inspection is done which does not comprise specification of smells. This technique is not automated and cannot be applied to broad systems. They ignore cases 1,3,4,6 due to analyst supervision throughout the detection process, but come across cases 2,5.

Marinescu [4] proposed a metric-based approach with detection techniques, executed in the IPLASMA tool, to detect design smells. To obtain 10 relevant anti-patterns, 10 detection techniques were illustrated. This approach has two shortcomings: a) requires vast knowledge of metric-based rules to successfully detect an anti-pattern. b) different threshold values result into distinct outcomes.

Likewise, Munro [5] presented metric-based heuristics to detect anti-patterns. He also proposed a template to characterize code smells systematically to overcome the limitation of text-based descriptions. The template includes three portions: name of code smell, text-based descriptions of its attributes, and heuristics to detect them. To describe the choice of metrics and thresholds to detect code smells, he performed an empirical study. The work presented by Marinescu [4] and Munro [5] only addressed cases 5, 6. It does not deal with the uncertainty, case 1.They lack any ranked lists of candidates, case 2.Their approaches does not support any of these cases: 1) Context of the programs. 2) Quality analysts' interpretation, case 3 and 4.

Alikacem and Sahraoui [6] presented an ad hoc domain specific language (DSL) that allows the specification of fuzzy-logic rules to express the thresholds of rules conditions. This language detects the defiance of quality principles and smells in object-oriented systems. The author did not verify their approach on real program and only addresses case 5.

To study legacy code, Ciupke [7] introduced a technique which specifies design problems as queries. He located the occurrences of these problems in a model derived from the source code. Table 1 represents the problems addressed by manual approaches.

3.2 Semi-automated Techniques

Dhambri et al. [2] (Manual approach) and Simon et al. [8] proposed some visualisation techniques to determine a trade-off between manual inspections (time-consuming and non-representative) and fully automated approaches (systematic and productive).They did not address cases 1,2.

Lanza and Marinescu [9] and van Emden and Moonen [10] presented approaches to perform fully automatic detection that do not deal with uncertainty and long lists, cases 1 and 2.Their approaches use visualisation techniques to show the results of detection. They do not take into account quality analysts' interpretation and thresholds, cases 3 and 4.

Rao and Reddy [11] came up with the use of design propagation probability matrices (DCPP matrix).They made use of change propagation between the design of artifacts to define Shotgun Surgery and Divergent Change anti-patterns. Their matrix model the uncertainty of the detection technique. It considers neither quality analysts' interpretation, nor the context of systems, cases 3 and 4 respectively. It compares candidates to fixed templates characterizing the anti-patterns and does not use thresholds, case 2. Semantic data is not used, case 6 and cannot report ranked lists, case 5.

Moha et al. [12] introduced a DSL based on set of rules (metrics, relation between classes) that describes the character of each anti-pattern. They defined a platform for automatic conversion of rule cards into detection algorithms and also proposed some algorithms [13]. All the available occurrences of well-known anti-patterns like Blob, Functional Decomposition, Spaghetti Code, and Swiss Army were identified with the result of 100% recall and precision between 41% and 88% with the average of 60%.Their detection approach confronted cases 4, 5, 6 but does not manage the uncertainty for the solution, case 1. It was not concerned with ranked list of candidates, case 2, and quality analysts' interpretations is not supported , case 3.

Table 1.Summarizes the problems addressed by some Manual Techniques tion

Problems	Manual Approaches					
	Travassos et al. (1999)	**Marinescu (2004)**	**Munro (2005)**	**Alikacem & Sahraoui (2006)**	**Dhambri et. al (2008)**	**Ciupke (2010)**
Problem 1	Y					
Problem 2						
Problem 3	Y				Y	Y
Problem 4	Y				Y	Y
Problem 5		Y	Y	Y	Y	
Problem 6	Y	Y	Y			Y

Expert), a Goal Question Metric (GQM) based approach. This

Table 2. Summarizes the problems addressed by some Semi-automated Techniques.

Problems	Semi-automated Approaches					
	Simon et al. (2001)	**van Emden & Moonen (2002)**	**Lanza & Marinescu (2006)**	**Rao and Reddy (2008)**	**Moha (2010b)**	**F.Khomh et al. (2011)**
Problem 1				Y		Y
Problem 2						Y
Problem 3	Y					
Problem 4	Y				Y	Y
Problem 5	Y	Y	Y		Y	Y
Problem 6					Y	Y

technique of detection takes into account anti-pattern definitions to build Bayesian Belief Networks (BBNs).It is highly uncertain to discover the occurrence of an anti-pattern. The outcome of BBNs is the probability that a class is an anti-pattern or not which addresses case 1. These probabilities allow the quality analysts' to rank the classes with respect to their probabilities.

Therefore, case 2 is also taken care of. It takes into account the judgement of quality analysts' when data is not available or need to standardize to some other context, case 3. The previous results, which are obtained in context of a given module and verified by a quality analyst, can be used to instruct a BBN.

Consequently, BBNs avoid cases 3, 4, and 5. Table 2 represents the problems addressed by various semi-automated approaches.

Table 3. Compares limitations and advantages of some anti-pattern detection techniques

Techniques	Limitations	Advantages over previous approaches
DETEX (DECOR) [12]	• Low precision & recall rate when implemented on subgroups of the system • Rule-based • Excessive number of faulty positives visible • Unranked outcomes	• an accomplished routine with a description language • a clear detection programme • verification of the technique for detection • thorough processing of technique
BDTEX [14]	• only provides a possibility that a class exists as an anti-pattern • Involves huge extent of unpredictability	• Outcomes were ranked. • Operate in case of omitted data. • Allows quality analysts' to use their knowledge. • Present better precision, recall, and utility (by quality analysts) than DECOR.
SMURF [15]	• To instruct the classifier, it demands tagged data.	• Incremental • Both inter and intra system applicability. • Allows users feedback • Good precision and recall in comparison to BDTEX and DETEX. • Users feedback enhances SMURF efficiency

3.3 SVM based Techniques

Maiga et.al [15] also proposed SMURF. This approach also uses a machine learning technique (SVM) using polynomial kernel as a basis for detection, but it takes into account the practitioners' feedback. SMURF is designed to work on both inter and intra system configurations. It results in a better precision and recall for both the system as a whole or subsystems. This approach is incremental and reveals that the efficiency of this technique can be improved by taking into account practitioners' feedback. Table 3 compares the limitations and advantages of some detection techniques.

4. CONCLUSION

The presence of design smells adversely affects the evolution of object-oriented systems. To improve the software testability and quality, anti-pattern detection is important. From the study on anti-pattern detection techniques, this paper concludes that the current state-of-art SMURF is the most accurate technique for detecting design smells till now [15]. In comparison to previous manual and semi-automated approaches (DETEX and BDTEX) for anti-pattern detection, SMURF has a much better precision and recall as it overcomes all the limitations of previous approaches.

5. REFERENCES

[1] Brown, W. J., Malveau, R. C., Brown , W. H., McCormick III , H. W., Mowbray, T. J. 1998. AntiPatterns: Refactoring Software, Architectures, and Projects in Crisis.1st ed. John Wiley and Sons.

[2] Dhambri, K., Sahraoui, H., Poulin, P. 2008. Visual detection of design anomalies. In Proceedings of the 12th European Conference on Software Maintenance and Reengineering.IEEE Computer Society, pp. 279–283.

[3] Travassos, G., Shull, F., Fredericks , M., Basili , V. R. 1999.Detecting defects in object-oriented designs: using reading techniques to increase software quality. In Proceedings of the 14th Conference on Object-Oriented Programming, Systems, Languages, and Applications.ACM Press,pp. 47–56.

[4] Marinescu, R. 2004. Detection strategies: metrics-based rules for detecting design flaws.In Proceedings of the 20th International Conference on Software Maintenance.IEEE Computer Society Press, pp. 350–359.

[5] Munro, M. J. 2005. Product metrics for automatic identification of "bad smell" design problems in java source-code.In Proceedings of the 11th International Software Metrics Symposium.IEEE Computer Society Press, pp.15.

[6] Alikacem, E., Sahraoui, H.Détection d'anomalies utilisant un langage de description de règle de qualité.In: Rousseau, R., Urtado, C., Vauttier, S. (Eds.), actes du 12e colloque Langages, Modèles, Objets. Hermès Science Publications, pp. 185–200.

[7] Ciupke, O. 1999. Automatic detection of design problems in object-oriented reengineering. In Proceeding of 30th Conference on Technology of Object-Oriented Languages and Systems. IEEE Computer Society Press, pp. 18–32.

[8] Simon, F., Steinbrückner, F., Lewerentz, C. 2001. Metrics based refactoring.In Proceedings of the Fifth European Conference on Software Maintenance and Reengineering (CSMR'01). IEEE Computer Society , pp. 30-38.

[9] Lanza, M., Marinescu, R. 2006.Object-Oriented Metrics in Practice. Springer Berlin Heidelberg,

[10] van Emden, E., Moonen, L. 2002. Java quality assurance by detecting code smells,In Proceedings of the 9th Working Conference on Reverse Engineering (WCRE'02).IEEE Computer Society Press.

[11] Rao, A. A., Reddy, K. N. 2008. Detecting bad smells in object oriented design using design change propagation probability matrix.In Proceedings of the International MultiConference of Engineers and Computer Scientists.

[12] Moha, N., Guéhéneuc, Y. -G., Duchien, L., Meur, A. -F. L. 2010. DECOR: a method for the specification and detection of code and design smells. IEEE Transactions on Software Engineering(2010a), vol. 36, no.1, pp. 20–36.

[13] Moha, N., Guéhéneuc, Y. -G., Meur, A. -F. L., Duchien, L., Tiberghien, A. 2010. From a domain analysis to the specification and detection of code and design smells. Formal Aspects of Computing (FAC), vol. 22, no. 3-4, 2010b, pp. 345-361.

[14] Khomh, F., Vaucher, S., Gu´eh´eneuc, Y. -G., Sahraoui, H. 2011. Bdtex: A gqm-based bayesian approach for the detection of antipatterns. J. Syst. Softw., vol. 84, no. 4, pp. 559–572.

[15] Maiga, A. et al. 2012. SMURF: a SVM based incremental anti-pattern detection approach.In Proceedings of the 19th Working Conference on Reverse Engineering (WCRE). IEEE Computer Society Press.