Clean CaDET
Clean Code and Design Educational Tool

REPORT ON THE DETECTION METHODS FOR TARGET SMELLS – Y2

Executive Summary

In this document, we review the Artificial-Intelligence-based detectors for code smell detection. We focus on our target smells: Refused (Parent) Bequest and Data Class. The reviewed approaches will serve as a guideline for designing our code smell detection methodologies. We also list the publicly available resources we may use to implement or train our code smell detectors.

Key Takeaways

1. *We list the publicly available resources for detecting our target smells (Refused (Parent) Bequest, Data Class): datasets, tools for preprocessing and feature extraction, and implementations of the existing detection approaches*
2. *We present data collected in our systematic literature review: the most commonly used detection approaches, datasets, and metrics.*
3. *In our previous report [1], we left Feature Envy code smell for a later phase of our project. Feature Envy will not be described in this report, as we already covered it in the previous one.*

Contact
Jelena Slivka
slivkaje@uns.ac.rs
https://clean-cadet.github.io/

1. Introduction

In our previous report [2], we have identified Refused (Parent) Bequest and Data Class as the most harmful types of code smells we plan to address in our project's second phase. In addition, we will cover the Feature Envy code smell, identified in the report [1]. In this document, we review the artificial intelligence approaches for the detection of Refused (Parent) Bequest and Data class smells. Detection techniques for Feature Envy were covered in the report [1]. We aim to identify:

- the most commonly used techniques for the detection of these smells,
- the limitations of the existing methods and possible ways to overcome them,
- publicly available resources we can use for their detection.

We will place a large part of our efforts on developing machine learning (ML) techniques for code smell detection. We performed a SLR of newly published approaches and a meta-analysis of their reported performances. In this document, we will present data extracted from our SLR.
In Sections 2 and 3, we give an overview of the techniques directed at detecting Data Class and Refused (Parent) Bequest code smells. In addition to detection techniques, we present the most commonly used datasets and structural code metrics in the literature.

2. Data Class detection

The authors of the paper [3] experimented with 16 different machine learning techniques for code smell detection. Data Class is one type of code smell that they detected. They considered 74 systems written in Java from the Qualitas Corpus (20120401r) collected by Tempero et al. [4]. Systems have different sizes and belong to different application domains. The data set contains 140 positive and 280 negative instances for Data Class code smell.

Selected machine learning classifiers are available in the Weka tool [5]: Support Vector Machines (SMO, LibSVM), Decision Trees (J48), Random Forest, Naive Bayes, and JRip. The choice of the algorithms was made by selecting different algorithms representing different approaches and possibly already exploited in the literature for similar purposes.

The authors computed a large set of object-oriented metrics used as independent variables in machine learning approaches. They used iPlasma [6], Fluid Tool [7], and Anti-Pattern Scanner [8] as reference tools (detection rules). Following the output of the code smell detection tools, the reported code smell candidates were manually evaluated and assigned severities. Severity values used in this paper are 0 (no smell), 1 (non-severe smell), 2 (smell), 3 (severe smell). They implemented 10-fold cross-validation.

In addition to the papers listed in the systematic literature review from Azeem et al. [9], we analyzed and collected data from primary studies of our SLR. We searched for definitions, heuristics, rules, metrics, and detection approaches that the collected papers observed.

The most commonly used techniques for Data Class detection are SVM, Multi-layer Perceptron, Decision Tree, Random Forest, Naïve Bayes, KNN, and Logistic Regression. All primary studies used structural code metrics, except for one study which included process metrics as well. Some of the most commonly used structural code metrics are WOC (Weight Of Class), NOAM (Number Of Accessor Methods), NOAP (Number of Public Attributes), WMCNAMM (Weighted Methods Count of Not Accessor or Mutator Methods), and LCOM (Lack of cohesion between methods).

The Fontana et al. dataset [3] stood out as the most used for Data class detection, with 74 open-source systems written in Java.

3. Refused (Parent) Bequest detection

In the paper [10] authors study the effectiveness of the Decision Tree algorithm to recognize code smells, among which is Refused (Parent) Bequest. The algorithm was applied on the dataset containing 4 open-source projects with 7952 classes. The information about the existence of code smells was derived from the datasets used in [11]. Regarding code metrics, 62 software metrics were obtained using CKJM [12] (18 metrics) and POM [13] (44 metrics) tools. To assess the statistical validity of the analysis, the authors followed the 10-fold crossover validation.

They presented results obtained for the DT algorithm C5.0 [14] as average precision, recall and F-measure values. Comparison with rule-based technique (iPlasma [6]) indicates that the DT algorithm can perform more effectively than pre-defined sets of rules. Similar conclusions are made when comparing the DT algorithm with the SVM. In terms of recall, the Bayesian Beliefs Networks algorithm performs more effectively.

The following paragraph present the conclusion drawn from the data collected from the papers of our SLR.

The most commonly used Refused (Parent) Bequest detection techniques are Decision Tree, Random Forest, and Naïve Bayes. The authors of the examined papers used only structural code metrics for detection; the most commonly used metrics are BUR (Base class Usage Ratio) and BOvR (Base class Overriding Ratio).

4. References

[1] Deliverables for the Clean-CaDET project funded by the Science Fund of the Republic of Serbia, Grant No. 6521051, AI - Clean CaDET https://github.com/Clean-CaDET/Deliverables/blob/main/work_package_1/D1.2.%20Report%20on%20the%20state-of-the-art%20AI-based%20detectors%20for%20target%20smells.pdf

[2] Deliverables for the Clean-CaDET project funded by the Science Fund of the Republic of Serbia, Grant No. 6521051, AI - Clean CaDET https://github.com/Clean-CaDET/Deliverables/blob/main/work_package_1/D1.1.%20Report%20on%20the%20selected%20target%20smells%20-%20Y2.pdf

[3] Arcelli Fontana, F., Mäntylä, M.V., Zanoni, M. *et al.* Comparing and experimenting machine learning techniques for code smell detection. *Empir Software Eng* **21,** 1143–1191 (2016).

[4] E. Tempero, C. Anslow, J. Dietrich, T. Han, J. Li, M. Lumpe, H. Melton, and J. Noble, “The qualitas corpus: A curated collection of java code for empirical studies,” in Proceedings of the 17th Asia Pacific Software Engineering Conference. IEEE Computer Society, December 2010, pp. 336–345.

[5] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I. H. Witten, “The weka data mining software: an update,” SIGKDD Explor. Newsl., vol. 11, pp. 10–18, November 2009.

[6] C. Marinescu, R. Marinescu, P. F. Mihancea, and R. Wettel, “iplasma: An integrated platform for quality assessment of object-oriented design,” in In ICSM (Industrial and Tool Volume. Society Press, 2005, pp. 77–80.

[7] K. Nongpong, “Integrating ”code smells” detection with refactoring tool support,” Ph.D. dissertation, University of Wisconsin Milwaukee, August 2012. [Online]. Available: http://dc.uwm.edu/etd/13

[8] S. Olbrich, D. Cruzes, and D. I. K. Sjoberg, “Are all code smells harmful? a study of god classes and brain classes in the evolution of three open source systems,” in IEEE International Conference on Software Maintenance (ICSM 2010), 2010, p. 10.

[9] Azeem, Muhammad Ilyas et al. “Machine learning techniques for code smell detection: A systematic literature review and meta-analysis.” Inf. Softw. Technol. 108 (2019): 115-138

[10] L. Amorim, E. Costa, N. Antunes, B. Fonseca and M. Ribeiro, "Experience report: Evaluating the effectiveness of decision trees for detecting code smells," 2015 IEEE 26th International Symposium on Software Reliability Engineering (ISSRE), 2015, pp. 261-269, doi: 10.1109/ISSRE.2015.7381819.

[11] F. Khomh, M. D. Penta, Y.-G. Gu´eh´eneuc, and G. Antoniol, “An exploratory study of the impact of antipatterns on class change- and fault-proneness,” Empirical Softw. Engg., vol. 17, no. 3, pp. 243–275, Jun. 2012.

[12] M. Jureczko and D. Spinellis, Using Object-Oriented Design Metrics to Predict Software Defects, ser. Monographs of System Dependability. Wroclaw, Poland: Oficyna Wydawnicza Politechniki Wroclawskiej, 2010, vol. Models and Methodology of System Dependability, pp. 69–81.

[13] Y.-G. Guehneuc, H. Sahraoui, and F. Zaidi, “Fingerprinting design patterns,” 2013 20th Working Conference on Reverse Engineering (WCRE), vol. 0, pp. 172–181, 2004.

[14] R. C. Barros, M. P. Basgalupp, A. C. P. L. F. de Carvalho, and A. A. Freitas, “A survey of evolutionary algorithms for decision-tree induction,” IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews, vol. 42, no. 3, pp. 291–312, MAY 2012.

The Project is funded by the Science Fund of the Republic of Serbia, Grant No. 6521051, AI - Clean CaDET

Science Fund
of the Republic of Serbia