A Survey on Machine Learning and Antipatterns

Rodger Byrd

I. Abstract

II. Background

A. Code Smells vs Anti Patterns

B. Machine Learning

C. Journal3

D. Topic Map

For my area I'm looking at Anti-Patterns in code, these are also known as code smells. I've also seen them referred to as Atoms of Confusion and nano paterrns. My topic map is included below in figure 1. My thoughts on gaps are some method to demonstrate the found anti-patterns are valid. Also, there seems to be a gap in how to fix anti-patterns and code smells.

E. Notes on Survey Papers,

For my first detailed read I chose a paper called *A survey on software smells* [6]. The following are my raw notes for this paper.

Kent Beck coined the term code smell, I should look at this paper. Contains some ideas for topic map in abstract. Figure 1 in paper has a pretty detailed topic map. Interesting for comparison. Table 4 code smells. Interesting section on whether anti-patterns and smells synonyms, doesn't seem to be consensus. Some papers yes, some no. Smell indicator of problem, anti pattern definitive problem. Current detection methods cause too many false positives? This wasn't as good as the first two papers, they did a very good job of summarizing the current research. This one talked way too much about how they did their analysis instead of the results.

For my next detailed read I chose a paper called *A systematic literature review: Refactoring for disclosing code smells in object oriented software* [8]. The following are my raw notes for this paper.

Refers to code smells and anti-patterns interchangeably. Related work is basically a background. Included 238 of an initial 1053 articles. Wow, that is a ton of papers. The papers they drew from were on the following topics: Refactoring, anti-patterns, code smells, Object Oriented Design and refactoring, fault tolerance and OOD, SW Metrics and anti-patterns, anti-patterns in cross company projects. Provide a lot of detail on the detection methods Interesting to show charts based on the publisher, Conference proceedings and IEEE were top sources.

For my next detailed read I chose a paper called *Smells in software test code: A survey of knowledge in industry and academia* [4]. The following are my raw notes for this paper.

Interesting variation on the idea of anti pattern: "test smells". Poorly designed tests. Really weird figures in this paper, copy paste screenshots out of spreadsheet and websites like google. Just make a table! They are missing the visualization, they are showing tons of tables instead of the results of the research. Missing the analysis?

The files for this latex document are in the github repository located at `https://github.com/rodger79/CS6000`

Relevan papers are referenced in the bibliography below.

F. Journal4

For my learning process so far, I've looked at how to use google scholar to optimize my time looking for the latest research. Up until this class I would go onto the UCCS library website and use the research databases to find papers. It has been very helpful to set up my google scholar profile to automatically show the papers I can download through my UCCS credentails. I have a couple of things on my to-do list. I need to speak with Prof Boult about how to get a private github as a student, as that would be very useful, and I have a handful of questions I would like to get his opinion on. One thing I want to focus on is very recent relevant papers. Fortunately google scholar allows you to researc only papers that have been written since a particular year.

III. Journal 4

A. Who is the Main Character

The main character is the anti-pattern. I need to find a way to clearly define the anti-pattern conceptually.

Listing 1. Example Anti-Pattern

```
#DEFINE M 2+3

int main() {
  int x=5, y;
  y= x * M;
  cout << y << endl;
  return 0;
}
```

There are some authors who define it as interchangeable as a code smell. I don't think that is correct. I think code smells are precursors to anti-patterns. Meaning that an anti-pattern is a known bad problem, where a code smell may lead to a problem. An example anti-pattern is shown in listing 1. Some developers will expect the output to be 25 and some will be 13 because they don't understand intuitively how the macro function works (the correct answer is 13).

On second thought, is the character the developer? And is what is interesting about it their human nature which leads them to confusion? I would guess tha most developers think

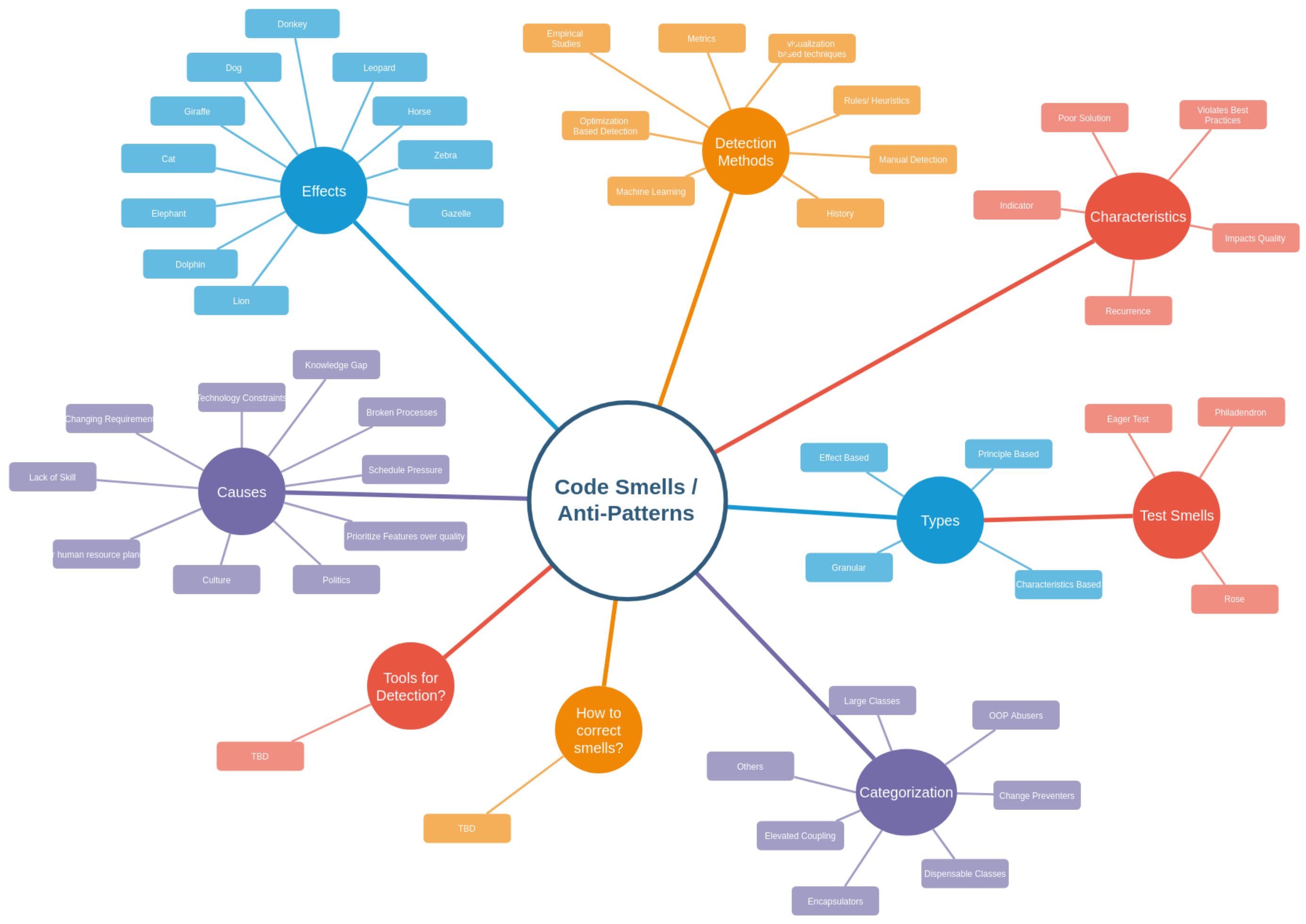

Fig. 1. TopicMap

that a compiler is well-defined and bugs in code must be due to lack of understanding not code structure that is good at confusing human nature.

B. *What Character Traits Make them Interesting*

What's interesting about anti-patterns is that they have a very human aspect to them. They overlap the way humans think with the way code is written. They connect common ways of human misunderstanding to software development. Personally, I think most developers expect code to work as they understand it to. They don't spend a lot of time thinking about how code will work in ways they don't expect. It is in the nature of most engineers to see things in mathematical/binary ways and ignore the human aspects to what they are working on. Example boeing 737 max. Code developers expected pilots to respond with typical emergency procedure, but when the errors occurred they pilots were overwhelmed by the amount of feedback they recieved in the cockpit (need reference).

C. *What do the Character Need to do or Get (Goal)*

The anti-patterns need to be understood and identified. Once that happens developers can change their best practices. The best practices should be created in such a way that the typical misunderstandings of the developer

D. *Why is That Goal Important (motive)*

Note: not many papers about code smells talk about why it is important.

E. *What Conflicts/Problems Block the Character*

The problem is people don't realize they are implementing anti-patterns at the time they are writing code. The interesting things about it are how do we find them. How do we detect them, and what is the fix when we identify the problem.

F. *How do they Create Risk and Danger*

Because we know that the anti-patterns cause confusion in the developer, they create risk because they create unstable code. This is risk for the owner of the software and the customer of the developer who uses code

G. *What Does the Character Do (Struggles) to Reach Goal*

This is something I think is interesting, is how are these problems identified. It isnt' enought to just use expert advice/-knowledge. There must be a way to mathematically demonstrate or by experimentation that particular code patterns cause problems.

H. What Sensory Details Will Make the Story Seem Real

Real world examples of the problem

References

[1] M. Mantyla, J. Vanhanen, and C. Lassenius, "A taxonomy and an initial empirical study of bad smells in code," in *International Conference on Software Maintenance, 2003. ICSM 2003. Proceedings.*, Sep. 2003, pp. 381–384.

[2] R. Arcoverde, A. Garcia, and E. Figueiredo, "Understanding the Longevity of Code Smells: Preliminary Results of an Explanatory Survey," in *Proceedings of the 4th Workshop on Refactoring Tools*, ser. WRT '11. New York, NY, USA: ACM, 2011, pp. 33–36, event-place: Waikiki, Honolulu, HI, USA. [Online]. Available: http://doi.acm.org/10.1145/1984732.1984740

[3] A. Yamashita and L. Moonen, "Do developers care about code smells? An exploratory survey," Oct., pp. 242–251.

[4] V. Garousi and B. Kk, "Smells in software test code: A survey of knowledge in industry and academia," *Journal of Systems and Software*, vol. 138, pp. 52 – 81, 2018. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0164121217303060

[5] N. Yoshioka, H. Washizaki, and K. Maruyama, "A survey on security patterns," *Progress in Informatics*, no. 5, p. 35, Mar. 2008. [Online]. Available: http://www.nii.ac.jp/pi/n5/5_35.html

[6] T. Sharma and D. Spinellis, "A survey on software smells," *Journal of Systems and Software*, vol. 138, pp. 158 – 173, 2018. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0164121217303114

[7] D. Gopstein, J. Iannacone, Y. Yan, L. DeLong, Y. Zhuang, M. K.-C. Yeh, and J. Cappos, "Understanding Misunderstandings in Source Code," in *Proceedings of the 2017 11th Joint Meeting on Foundations of Software Engineering*, ser. ESEC/FSE 2017. New York, NY, USA: ACM, 2017, pp. 129–139, event-place: Paderborn, Germany. [Online]. Available: http://doi.acm.org/10.1145/3106237.3106264

[8] S. Singh and S. Kaur, "A systematic literature review: Refactoring for disclosing code smells in object oriented software," *Ain Shams Engineering Journal*, vol. 9, no. 4, pp. 2129 – 2151, 2018. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S2090447917300412

[9] Z. Li, T.-H. P. Chen, J. Yang, and W. Shang, "Dlfinder: Characterizing and Detecting Duplicate Logging Code Smells," in *Proceedings of the 41st International Conference on Software Engineering*, ser. ICSE '19. Piscataway, NJ, USA: IEEE Press, 2019, pp. 152–163, event-place: Montreal, Quebec, Canada. [Online]. Available: https://doi.org/10.1109/ICSE.2019.00032

[10] M. S. Haque, J. Carver, and T. Atkison, "Causes, Impacts, and Detection Approaches of Code Smell: A Survey," in *Proceedings of the ACMSE 2018 Conference*, ser. ACMSE '18. New York, NY, USA: ACM, 2018, pp. 25:1–25:8, event-place: Richmond, Kentucky. [Online]. Available: http://doi.acm.org/10.1145/3190645.3190697

[11] F. A. Fontana, V. Lenarduzzi, R. Roveda, and D. Taibi, "Are architectural smells independent from code smells? An empirical study," *Journal of Systems and Software*, vol. 154, pp. 139 – 156, 2019. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0164121219301013

[12] B. Walter, F. A. Fontana, and V. Ferme, "Code smells and their collocations: A large-scale experiment on open-source systems," *Journal of Systems and Software*, vol. 144, pp. 1 – 21, 2018. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0164121218301109

[13] S. S. Afjehei, T.-H. P. Chen, and N. Tsantalis, "iPerfDetector: Characterizing and detecting performance anti-patterns in iOS applications," *Empirical Software Engineering*, Apr. 2019. [Online]. Available: https://doi.org/10.1007/s10664-019-09703-y

[14] F. Tian, P. Liang, and M. A. Babar, "How Developers Discuss Architecture Smells? An Exploratory Study on Stack Overflow," in *2019 IEEE International Conference on Software Architecture (ICSA)*, Mar. 2019, pp. 91–100.

[15] C. Vassallo, S. Proksch, H. C. Gall, and M. Di Penta, "Automated Reporting of Anti-patterns and Decay in Continuous Integration," in *Proceedings of the 41st International Conference on Software Engineering*, ser. ICSE '19. Piscataway, NJ, USA: IEEE Press, 2019, pp. 105–115, event-place: Montreal, Quebec, Canada. [Online]. Available: https://doi.org/10.1109/ICSE.2019.00028

[16] D. Taibi, V. Lenarduzzi, and C. Pahl, *Microservices Anti Patterns: A Taxonomy*, 2019.

[17] A. Tahir, A. Yamashita, S. Licorish, J. Dietrich, and S. Counsell, "Can You Tell Me if It Smells?: A Study on How Developers Discuss Code Smells and Anti-patterns in Stack Overflow," in *Proceedings of the 22Nd International Conference on Evaluation and Assessment in Software Engineering 2018*, ser. EASE'18. New York, NY, USA: ACM, 2018, pp. 68–78, event-place: Christchurch, New Zealand. [Online]. Available: http://doi.acm.org/10.1145/3210459.3210466

[18] R. Ibrahim, M. Ahmed, R. Nayak, and S. Jamel, "Reducing redundancy of test cases generation using code smell detection and refactoring," *Journal of King Saud University - Computer and Information Sciences*, 2018. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1319157818300296

[19] H. Brabra, A. Mtibaa, F. Petrillo, P. Merle, L. Sliman, N. Moha, W. Gaaloul, Y.-G. Guhneuc, B. Benatallah, and F. Gargouri, "On semantic detection of cloud API (anti)patterns," *Information and Software Technology*, vol. 107, pp. 65 – 82, 2019. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S095058491830226X

[20] S. Hussain, J. Keung, M. K. Sohail, A. A. Khan, G. Ahmad, M. R. Mufti, and H. A. Khatak, "Methodology for the quantification of the effect of patterns and anti-patterns association on the software quality," *IET Software*, vol. 13, no. 5, pp. 414–422, 2019.

[21] Y. Lyu, D. Li, and W. G. J. Halfond, "Remove RATs from Your Code: Automated Optimization of Resource Inefficient Database Writes for Mobile Applications," in *Proceedings of the 27th ACM SIGSOFT International Symposium on Software Testing and Analysis*, ser. ISSTA 2018. New York, NY, USA: ACM, 2018, pp. 310–321, event-place: Amsterdam, Netherlands. [Online]. Available: http://doi.acm.org/10.1145/3213846.3213865

[22] A. Abadi, M. Abadi, and I. Ben-Harrush, "Fixing anti-patterns in javascript," US Patent US9 983 975B2, May, 2018. [Online]. Available: https://patents.google.com/patent/US9983975B2/en

[23] M. Kessentini, R. Mahaouachi, and K. Ghedira, "What you like in design use to correct bad-smells," *Software Quality Journal*, vol. 21, no. 4, pp. 551–571, Dec. 2013. [Online]. Available: https://doi.org/10.1007/s11219-012-9187-6

[24] A. Kaur, K. Kaur, and S. Jain, "Predicting software change-proneness with code smells and class imbalance learning," in *2016 International Conference on Advances in Computing, Communications and Informatics (ICACCI)*, Sep. 2016, pp. 746–754.

[25] N. Pritam, M. Khari, L. H. Son, R. Kumar, S. Jha, I. Priyadarshini, M. Abdel-Basset, and H. V. Long, "Assessment of Code Smell for Predicting Class Change Proneness Using Machine Learning," *IEEE Access*, vol. 7, pp. 37 414–37 425, 2019.

[26] A. Kaur, S. Jain, and S. Goel, "SP-J48: a novel optimization and machine-learning-based approach for solving complex problems: special application in software engineering for detecting code smells," *Neural Computing and Applications*, Apr. 2019. [Online]. Available: https://doi.org/10.1007/s00521-019-04175-z

[27] A. Maiga, N. Ali, N. Bhattacharya, A. Saban, Y. Guhneuc, and E. Aimeur, "SMURF: A SVM-based Incremental Anti-pattern Detection Approach," in *2012 19th Working Conference on Reverse Engineering*, Oct. 2012, pp. 466–475.

[28] L. Kumar and A. Sureka, "An Empirical Analysis on Web Service Anti-pattern Detection Using a Machine Learning Framework," in *2018 IEEE 42nd Annual Computer Software and Applications Conference (COMPSAC)*, vol. 01, Jul. 2018, pp. 2–11.

[29] D. D. Nucci, F. Palomba, D. A. Tamburri, A. Serebrenik, and A. D. Lucia, "Detecting code smells using machine learning techniques: Are we there yet?" in *2018 IEEE 25th International Conference on Software Analysis, Evolution and Reengineering (SANER)*, Mar. 2018, pp. 612–621.

[30] R. Malhotra, "A systematic review of machine learning techniques for software fault prediction," *Applied Soft Computing*, vol. 27, pp. 504 – 518, 2015. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1568494614005857

[31] F. A. Fontana and M. Zanoni, "Code smell severity classification using machine learning techniques," *Knowledge-Based Systems*, vol. 128, pp. 43 – 58, 2017. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0950705117301880

[32] F. Arcelli Fontana, M. V. Mntyl, M. Zanoni, and A. Marino, "Comparing and experimenting machine learning techniques for code smell detection," *Empirical Software Engineering*, vol. 21, no. 3, pp. 1143–1191, Jun. 2016. [Online]. Available: https://doi.org/10.1007/s10664-015-9378-4

[33] *Findings from FUMEC University Provides New Data on Machine Learning (Machine Learning Techniques for Code Smells Detection: a Systematic Mapping Study)*, 2019.

[34] U. Azadi, F. A. Fontana, and M. Zanoni, “Poster: machine learning based code smell detection through WekaNose,” in *2018 IEEE/ACM 40th International Conference on Software Engineering: Companion Proceedings (ICSE-Companion)*. IEEE, 2018, pp. 288–289.

[35] M. Kessentini, “Understanding the Correlation between Code Smells And Software Bugs,” 2019.

[36] A. Barbez, F. Khomh, and Y.-G. Guhneuc, “A Machine-learning Based Ensemble Method For Anti-patterns Detection,” *CoRR*, vol. abs/1903.01899, 2019. [Online]. Available: http://arxiv.org/abs/1903.01899

[37] S. Saluja and U. Batra, “Assessing Quality by Anti-pattern Detection in Web Services,” Social Science Research Network, Rochester, NY, SSRN Scholarly Paper ID 3350876, Mar. 2019. [Online]. Available: https://papers.ssrn.com/abstract=3350876

[38] W. Song, C. Zhang, and H. Jacobsen, “An Empirical Study on Data Flow Bugs in Business Processes,” *IEEE Transactions on Cloud Computing*, pp. 1–1, 2018.

[39] M. I. Azeem, F. Palomba, L. Shi, and Q. Wang, “Machine learning techniques for code smell detection: A systematic literature review and meta-analysis,” *Information and Software Technology*, vol. 108, pp. 115 – 138, 2019. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0950584918302623

[40] V. Garousi, B. Kucuk, and M. Felderer, “What We Know About Smells in Software Test Code,” *IEEE Software*, vol. 36, no. 3, pp. 61–73, May 2019.

[41] D. Arcelli, V. Cortellessa, and D. D. Pompeo, “Performance-driven software model refactoring,” *Information and Software Technology*, vol. 95, pp. 366 – 397, 2018. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0950584917301787

[42] S. Fakhoury, V. Arnaoudova, C. Noiseux, F. Khomh, and G. Antoniol, “Keep it simple: Is deep learning good for linguistic smell detection?” in *2018 IEEE 25th International Conference on Software Analysis, Evolution and Reengineering (SANER)*, Mar. 2018, pp. 602–611.

[43] Z. Li, X.-Y. Jing, and X. Zhu, “Progress on approaches to software defect prediction,” *IET Software*, vol. 12, no. 3, pp. 161–175(14), Jun. 2018. [Online]. Available: https://digital-library.theiet.org/content/journals/10.1049/iet-sen.2017.0148

[44] J. Carver, B. Penzenstadler, and A. Serebrenik, “Software Analysis, Evolution, and Reengineering, and ICT Sustainability,” *IEEE Software*, vol. 35, no. 4, pp. 78–80, Jul. 2018.

[45] M. A. Zaidi and R. Colomo-Palacios, “Code Smells Enabled by Artificial Intelligence: A Systematic Mapping,” in *Computational Science and Its Applications ICCSA 2019*, S. Misra, O. Gervasi, B. Murgante, E. Stankova, V. Korkhov, C. Torre, A. M. A. Rocha, D. Taniar, B. O. Apduhan, and E. Tarantino, Eds. Cham: Springer International Publishing, 2019, pp. 418–427.

[46] G. M. Ubayawardana and D. D. Karunaratna, “Bug Prediction Model using Code Smells,” in *2018 18th International Conference on Advances in ICT for Emerging Regions (ICTer)*, Sep. 2018, pp. 70–77.

[47] X. Ban, S. Liu, C. Chen, and C. Chua, “A performance evaluation of deep-learnt features for software vulnerability detection,” *Concurrency and Computation: Practice and Experience*, vol. 31, no. 19, p. e5103, 2019. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1002/cpe.5103

[48] L. Pellegrini and V. Lenarduzzi, “Are Code Smells the Root Cause of Faults?: A Continuous Experimentation Approach,” in *Proceedings of the 19th International Conference on Agile Software Development: Companion*, ser. XP ’18. New York, NY, USA: ACM, 2018, pp. 28:1–28:3, event-place: Porto, Portugal. [Online]. Available: http://doi.acm.org/10.1145/3234152.3234153

[49] T. Sharma, “Detecting and Managing Code Smells: Research and Practice,” in *Proceedings of the 40th International Conference on Software Engineering: Companion Proceeedings*, ser. ICSE ’18. New York, NY, USA: ACM, 2018, pp. 546–547, event-place: Gothenburg, Sweden. [Online]. Available: http://doi.acm.org/10.1145/3183440.3183460

[50] V. K. Kulamala, A. S. C. Teja, A. Maru, Y. Singla, and D. P. Mohapatra, “Predicting Software Reliability using Computational Intelligence Techniques: A Review,” in *2018 International Conference on Information Technology (ICIT)*, Dec. 2018, pp. 114–119.

[51] N. Tsuda, H. Washizaki, Y. Fukazawa, Y. Yasuda, and S. Sugimura, “Machine Learning to Evaluate Evolvability Defects: Code Metrics Thresholds for a Given Context,” in *2018 IEEE International Conference on Software Quality, Reliability and Security (QRS)*, Jul. 2018, pp. 83–94.

[52] K. Karauzovi-Hadiabdi and R. Spahi, “Comparison of Machine Learning Methods for Code Smell Detection Using Reduced Features,” in *2018 3rd International Conference on Computer Science and Engineering (UBMK)*, Sep. 2018, pp. 670–672.

[53] J. A. Reshi and S. Singh, “Investigating the Role of Code Smells in Preventive Maintenance,” *Journal of Information Technology Management*, vol. 10, no. 4, pp. 41–63, 2019.

[54] H. Liu, Z. Xu, and Y. Zou, “Deep Learning Based Feature Envy Detection,” in *Proceedings of the 33rd ACM/IEEE International Conference on Automated Software Engineering*, ser. ASE 2018. New York, NY, USA: ACM, 2018, pp. 385–396, event-place: Montpellier, France. [Online]. Available: http://doi.acm.org/10.1145/3238147.3238166

[55] B. Grodniyomchai, K. Chalapat, K. Jitkajornwanich, and S. Jaiyen, “A Deep Learning Model for Odor Classification Using Deep Neural Network,” in *2019 5th International Conference on Engineering, Applied Sciences and Technology (ICEAST)*, Jul. 2019, pp. 1–4.

[56] H. Liu, J. Jin, Z. Xu, Y. Bu, Y. Zou, and L. Zhang, “Deep Learning Based Code Smell Detection,” *IEEE Transactions on Software Engineering*, pp. 1–1, 2019.

[57] P. Kokol, M. Zorman, G. Zlahtic, and B. Zlahtic, *Code smells*, 2018.

[58] K. Alkharabsheh, J. A. Taboada, Y. Crespo, and T. Alzu’bi, “Improving Design Smell Detection for Adoption in Industry,” in *2018 8th International Conference on Computer Science and Information Technology (CSIT)*, Jul. 2018, pp. 213–218.

[59] N. Kamaraj and A. Ramani, “Search-Based Software Engineering Approach for Detecting Code-Smells with Development of Unified Model for Test Prioritization Strategies,” *International Journal of Applied Engineering Research*, vol. 14, no. 7, pp. 1599–1603, 2019.

[60] A. Gupta, B. Suri, V. Kumar, S. Misra, T. Blaauskas, and R. Damaeviius, “Software code smell prediction model using Shannon, Rnyi and Tsallis entropies,” *Entropy*, vol. 20, no. 5, p. 372, 2018.

[61] M. Lafi, J. W. Botros, H. Kafaween, A. B. Al-Dasoqi, and A. Al-Tamimi, “Code Smells Analysis Mechanisms, Detection Issues, and Effect on Software Maintainability,” in *2019 IEEE Jordan International Joint Conference on Electrical Engineering and Information Technology (JEEIT)*, Apr. 2019, pp. 663–666.

[62] R. Spahi and K. Kara\djuzovi-Hadiabdi, “Class Level Code Smell Detection using Machine Learning Methods,” *book of*, p. 74, 2018.

[63] F. Ferreira, L. L. Silva, and M. T. Valente, *Software Engineering Meets Deep Learning: A Literature Review*, 2019.

[64] H. Foidl and M. Felderer, “Risk-based Data Validation in Machine Learning-based Software Systems,” in *Proceedings of the 3rd ACM SIGSOFT International Workshop on Machine Learning Techniques for Software Quality Evaluation*, ser. MaLTeSQuE 2019. New York, NY, USA: ACM, 2019, pp. 13–18, event-place: Tallinn, Estonia. [Online]. Available: http://doi.acm.org/10.1145/3340482.3342743

[65] J. A. Reshi and S. Singh, *Predicting Software Defects through SVM: An Empirical Approach*, 2018.

[66] Y. Wang, S. Hu, L. Yin, and X. Zhou, “Using Code Evolution Information to Improve the Quality of Labels in Code Smell Datasets,” in *2018 IEEE 42nd Annual Computer Software and Applications Conference (COMPSAC)*, vol. 01, Jul. 2018, pp. 48–53.

[67] T. Sharma, V. Efstathiou, P. Louridas, and D. Spinellis, *On the Feasibility of Transfer-learning Code Smells using Deep Learning*, 2019.

[68] J. Rubin, A. N. Henniche, N. Moha, M. Bouguessa, and N. Bousbia, “Sniffing Android Code Smells: An Association Rules Mining-Based Approach,” in *2019 IEEE/ACM 6th International Conference on Mobile Software Engineering and Systems (MOBILESoft)*, May 2019, pp. 123–127.

[69] A. Kaur and S. Singh, “International Journal of Applied Engineering Research.” [Online]. Available: https://www.ripublication.com/ijaer18/ijaerv13n11_176.pdf

[70] F. Pecorelli, F. Palomba, D. Di Nucci, and A. De Lucia, “Comparing Heuristic and Machine Learning Approaches for Metric-based Code Smell Detection,” in *Proceedings of the 27th International Conference on Program Comprehension*, ser. ICPC ’19. Piscataway, NJ, USA: IEEE Press, 2019, pp. 93–104, event-place: Montreal, Quebec, Canada. [Online]. Available: https://doi.org/10.1109/ICPC.2019.00023

[71] P. Kriens and T. Verbelen, *Software Engineering Practices for Machine Learning*, 2019.

[72] B. Chernis and R. Verma, “Machine Learning Methods for Software Vulnerability Detection,” in *Proceedings of the Fourth ACM International Workshop on Security and Privacy Analytics*, ser. IWSPA ’18. New York, NY, USA: ACM, 2018, pp. 31–39, event-place: Tempe, AZ, USA. [Online]. Available: http://doi.acm.org/10.1145/3180445.3180453

[73] Y. Xiong, B. Wang, G. Fu, and L. Zang, “Learning to Synthesize,” in *Proceedings of the 4th International Workshop on Genetic Improvement Workshop*, ser. GI ’18. New York, NY, USA: ACM, 2018, pp. 37–44, event-place: Gothenburg, Sweden. [Online]. Available: http://doi.acm.org/10.1145/3194810.3194816

[74] M. Hirsch, A. Rodriguez, J. M. Rodriguez, C. Mateos, and A. Zunino, “Spotting and Removing WSDL Anti-pattern Root Causes in Code-first Web Services Using NLP Techniques: A Thorough Validation of Impact on Service Discoverability,” *Computer Standards & Interfaces*, vol. 56, pp. 116 – 133, 2018. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0920548917300892