**METHODOLOGY**

To identify the research works on mining software repositories, we adopted the search and selection process that involved renowned digital libraries, i.e., IEEE Xplore, ACM, Springer, Google Scholar, and ScienceDirect. Such libraries are selected because they are reputable and publish a significant number of SE-related journals and conference proceedings. For example, ACM and IEEE publish proceedings of the Conference on Mining Software Repositories. To query these libraries, we used pre-defined keywords including "Refactoring", "Design Flaws", "Code Smells", "Change History", "Commit Messages", "Software History", and "Mining Software Repositories". Each library was searched by these keywords individually and in different combinations to find research works that fall within the scope of our study. The search strings were composed to contain important keywords that appear in most of the papers of our interest. For example, we formulated the following query strings to search the libraries.

1. "Mining change history to detect code smells"
2. "Mining software repositories to drive refactoring"
3. "Mining, code smells, refactoring, change history"

The following table highlights the results per query for each digital library. However, it is worth noting that, ACM was returning a big number of results and therefore we had to select the top few papers that appeared more relevant to our study.

| SNo | Query | Digital Library | Number of outcomes |
|---|---|---|---|
| 1 | Mining change history to detect code smells | IEEE Xplore | 7 |
| | | ACM | 21,161 |
| | | Springer | 844 |
| | | ScienceDirect | 169 |
| 2 | Mining software repositories to drive refactoring | IEEE Xplore | 6 |
| | | ACM | 243,894 |
| | | Springer | 882 |
| | | ScienceDirect | 102 |
| 3 | Mining, code smells, refactoring, change history | IEEE Xplore | 4 |
| | | ACM | 289,333 |
| | | Springer | 404 |
| | | ScienceDirect | 95 |

**LIST OF PAPERS COMPOSING OUR DATASET**


1. AlOmar, E. A., Mkaouer, M. W., & Ouni, A. (2021). Toward the automatic classification of Self-Affirmed Refactoring. *Journal of Systems and Software*, *171*. https://doi.org/10.1016/J.JSS.2020.110821
2. Babii, H., Prenner, J. A., Stricker, L., Karmakar, A., Janes, A., & Robbes, R. (2021). Mining Software Repositories with a Collaborative Heuristic Repository. *Proceedings - International Conference on Software Engineering*, 106–110. https://doi.org/10.1109/ICSE-NIER52604.2021.00030
3. Behl, D., Handa, S., & Arora, A. (2014). A bug Mining tool to identify and analyze security bugs using Naive Bayes and TF-IDF: A Comparative Analysis. *ICROIT 2014 - Proceedings of the 2014 International Conference on Reliability, Optimization and Information Technology*, 294–299. https://doi.org/10.1109/ICROIT.2014.6798341
4. Chatzigeorgiou, A., & Manakos, A. (2013). Investigating the evolution of code smells in object-oriented systems. *Innovations in Systems and Software Engineering 2013 10:1*, *10*(1), 3–18. https://doi.org/10.1007/S11334-013-0205-Z
5. Choudhary, A., & Singh, P. (2016). *Minimizing Refactoring Effort through Prioritization of Classes based on Historical, Architectural and Code Smell Information*.
6. D'Ambros, M., Gall, H., Lanza, M., & Pinzger, M. (2008). Analysing Software Repositories to Understand Software Evolution. *Software Evolution*, 37–67. https://doi.org/10.1007/978-3-540-76440-3_3
7. Elmishali, A., Sotto-Mayor, B., Roshanski, I., Sultan, A., & Kalech, M. (2021). BEIRUT: Repository Mining for Defect Prediction. *Proceedings - International Symposium on Software Reliability Engineering, ISSRE*, *2021-October*, 47–56. https://doi.org/10.1109/ISSRE52982.2021.00018
8. Fu, S., & Shen, B. (2015). Code Bad Smell Detection through Evolutionary Data Mining. *International Symposium on Empirical Software Engineering and Measurement*, *2015-Novem*, 41–49. https://doi.org/10.1109/ESEM.2015.7321194
9. Keivanloo, I., Zhang, F., & Zou, Y. (2015). Threshold-free code clone detection for a large-scale heterogeneous Java repository. *2015 IEEE 22nd International Conference on Software Analysis, Evolution, and Reengineering, SANER 2015 - Proceedings*, 201–210. https://doi.org/10.1109/SANER.2015.7081830
10. Kessentini, M., Dea, T. J., & Ouni, A. (2017). A context-based refactoring recommendation approach using simulated annealing. *Proceedings of the Genetic and Evolutionary Computation Conference, GECCO 2017, Berlin, Germany, July 15-19, 2017*, 1303–1310. https://doi.org/10.1145/3071178.3071334
11. Krasniqi, R., & Cleland-Huang, J. (2020). Enhancing Source Code Refactoring Detection with Explanations from Commit Messages. *SANER 2020 - Proceedings of the 2020 IEEE 27th International Conference on Software Analysis, Evolution, and Reengineering*, 512–516. https://doi.org/10.1109/SANER48275.2020.9054816
12. Lozano, A., Wermelinger, M., & Nuseibeh, B. (2007). Assessing the impact of bad smells using historical information. *International Workshop on Principles of Software Evolution (IWPSE)*, 31–34. https://doi.org/10.1145/1294948.1294957
13. Luaphol, B., Polpinij, J., & Kaenampornpan, M. (2022). Mining Bug Report Repositories to Identify Significant Information for Software Bug Fixing. *Applied Science and Engineering Progress*, *15*(3). https://doi.org/10.14416/J.ASEP.2021.03.005
14. Mani, S. (2014). Improving enterprise software maintenance efficiency through mining software repositories in an industry context. *36th International Conference on Software Engineering, ICSE Companion 2014 - Proceedings*, 706–709. https://doi.org/10.1145/2591062.2591085
15. Nyamawe, A. S. (2022). Mining commit messages to enhance software refactorings

recommendation: A machine learning approach. *Machine Learning with Applications*, *9*, 100316. https://doi.org/10.1016/J.MLWA.2022.100316
16. Nyamawe, A. S., Bakhti, K., & Sandiwarno, S. (2021). Identifying rename refactoring opportunities based on feature requests. *International Journal of Computers and Applications*. https://doi.org/10.1080/1206212X.2021.1922151
17. Nyamawe, A. S., Liu, H., Niu, N., Umer, Q., & Niu, Z. (2019). Automated recommendation of software refactorings based on feature requests. *Proceedings of the IEEE International Conference on Requirements Engineering*, *2019-Septe*, 187–198. https://doi.org/10.1109/RE.2019.00029
18. Nyamawe, A. S., Liu, H., Niu, N., Umer, Q., & Niu, Z. (2020). Feature requests-based recommendation of software refactorings. *Empirical Software Engineering 2020 25:5*, *25*(5), 4315–4347. https://doi.org/10.1007/S10664-020-09871-2
19. Olbrich, S., Cruzes, D. S., Basili, V., & Zazworka, N. (2009). The evolution and impact of code smells: A case study of two open source systems. *2009 3rd International Symposium on Empirical Software Engineering and Measurement, ESEM 2009*, 390–400. https://doi.org/10.1109/ESEM.2009.5314231
20. Ouni, A., Kessentini, M., Ó Cinnéide, M., Sahraoui, H., Deb, K., & Inoue, K. (2017). MORE: A multi-objective refactoring recommendation approach to introducing design patterns and fixing code smells. *Journal of Software: Evolution and Process*, *29*(5), e1843. https://doi.org/10.1002/SMR.1843
21. Ouni, A., Kessentini, M., Sahraoui, H., & Hamdi, M. S. (2013a). The use of development history in software refactoring using a multi-objective evolutionary algorithm. *GECCO 2013 - Proceedings of the 2013 Genetic and Evolutionary Computation Conference*, 1461–1468. https://doi.org/10.1145/2463372.2463554
22. Ouni, A., Kessentini, M., Sahraoui, H., Inoue, K., & Hamdi, M. S. (2015). Improving multi-objective code-smells correction using development history. *Journal of Systems and Software*, *105*, 18–39. https://doi.org/10.1016/J.JSS.2015.03.040
23. Palomba, F., Bavota, G., Di Penta, M., Oliveto, R., De Lucia, A., & Poshyvanyk, D. (2013). Detecting bad smells in source code using change history information. *2013 28th IEEE/ACM International Conference on Automated Software Engineering, ASE 2013 - Proceedings*, 268–278. https://doi.org/10.1109/ASE.2013.6693086
24. Palomba, F., Bavota, G., Di Penta, M., Oliveto, R., Poshyvanyk, D., & De Lucia, A. (2015). Mining version histories for detecting code smells. *IEEE Transactions on Software Engineering*, *41*(5), 462–489. https://doi.org/10.1109/TSE.2014.2372760
25. Palomba, F., Zaidman, A., Oliveto, R., & De Lucia, A. (2017). An Exploratory Study on the Relationship between Changes and Refactoring. *IEEE International Conference on Program Comprehension*, 176–185. https://doi.org/10.1109/ICPC.2017.38
26. Peruma, A., Mkaouer, M. W., Decker, M. J., & Newman, C. D. (2019). Contextualizing rename decisions using refactorings and commit messages. *Proceedings - 19th IEEE International Working Conference on Source Code Analysis and Manipulation, SCAM 2019*, 74–85. https://doi.org/10.1109/SCAM.2019.00017
27. Peruma, A., Mkaouer, M. W., Decker, M. J., & Newman, C. D. (2020). Contextualizing rename decisions using refactorings, commit messages, and data types. *Journal of Systems and Software*, *169*, 110704. https://doi.org/10.1016/J.JSS.2020.110704
28. Peters, R., & Zaidman, A. (2012). Evaluating the lifespan of code smells using software repository mining. *Proceedings of the European Conference on Software Maintenance and Reengineering, CSMR*, 411–416. https://doi.org/10.1109/CSMR.2012.79
29. Ratzinger, J., Sigmund, T., Vorburger, P., & Gall, H. C. (2007a). Mining Software Evolution to Predict Refactoring. *First International Symposium on Empirical Software Engineering and Measurement, ESEM*, 354— – 363. https://sci-hub.se/10.1109/ESEM.2007.9
30. Rebai, S., Kessentini, M., Alizadeh, V., Sghaier, O. Ben, & Kazman, R. (2020). Recommending

refactorings via commit message analysis. *Information and Software Technology*, *126*. https://doi.org/10.1016/J.INFSOF.2020.106332
31. Silva, D., & Valente, M. T. (2017). RefDiff: Detecting Refactorings in Version Histories. *IEEE International Working Conference on Mining Software Repositories*, 269–279. https://doi.org/10.1109/MSR.2017.14
32. Soares, G., Catão, B., Varjão, C., Aguiar, S., Gheyi, R., & Massoni, T. (2011). Analyzing refactorings on software repositories. *Proceedings - 25th Brazilian Symposium on Software Engineering, SBES 2011*, 164–173. https://doi.org/10.1109/SBES.2011.21
33. Thung, F., Wang, S., Lo, D., & Lawall, J. (2013). Automatic recommendation of API methods from feature requests. *2013 28th IEEE/ACM International Conference on Automated Software Engineering, ASE 2013 - Proceedings*, 290–300. https://doi.org/10.1109/ASE.2013.6693088
34. TIAN, Y. (2017). Mining software repositories for automatic software bug management from bug triaging to patch backporting. *Dissertations and Theses Collection*. https://ink.library.smu.edu.sg/etd_coll_all/26
35. Tsantalis, N., & Chatzigeorgiou, A. (2011). Ranking refactoring suggestions based on historical volatility. *Proceedings of the European Conference on Software Maintenance and Reengineering, CSMR*, 25–34. https://doi.org/10.1109/CSMR.2011.7
36. Tsantalis, N., Mansouri, M., Eshkevari, L. M., Mazinanian, D., & Dig, D. (2018). *Accurate and efficient refactoring detection in commit history*. 483–494. https://doi.org/10.1145/3180155.3180206
37. Tufano, M., Palomba, F., Bavota, G., Olivetox, R., Di Penta, M., De Lucia, A., & Poshyvanyk, D. (2015). When and why your code starts to smell bad. *Proceedings - International Conference on Software Engineering*, *1*, 403–414. https://doi.org/10.1109/ICSE.2015.59
38. Wang, W., & Godfrey, M. W. (2014). Recommending clones for refactoring using design, context, and history. *Proceedings - 30th International Conference on Software Maintenance and Evolution, ICSME 2014*, 331–340. https://doi.org/10.1109/ICSME.2014.55
39. Weißgerber, P., & Diehl, S. (2006). Identifying refactorings from source-code changes. *Proceedings - 21st IEEE/ACM International Conference on Automated Software Engineering, ASE 2006*, 231–240. https://doi.org/10.1109/ASE.2006.41
40. Xu, S., Sivaraman, A., Khoo, S. C., & Xu, J. (2017). GEMS: An Extract Method Refactoring Recommender. *Proceedings - International Symposium on Software Reliability Engineering, ISSRE*, *2017-October*, 24–34. https://doi.org/10.1109/ISSRE.2017.35
41. Ying, A. T. T., Murphy, G. C., Ng, R., & Chu-Carroll, M. C. (2004a). Predicting source code changes by mining change history. *IEEE Transactions on Software Engineering*, *30*(9), 574–586. https://doi.org/10.1109/TSE.2004.52
42. Zimmermann, T., Weißgerber, P., Diehl, S., & Zeller, A. (2004). Mining version histories to guide software changes. *Proceedings - International Conference on Software Engineering*, *26*, 563–572. https://doi.org/10.1109/ICSE.2004.1317478