008/S/RPL-KCBR/PK.03.08/6/APRIL/2023

PENGARUH *REFACTORING CODE SMELLS* DENGAN *AUTOMATIC STATIC ANALYSIS TOOLS* TERHADAP PENGGUNAAN SUMBER DAYA PERANGKAT LUNAK SELULER ANDROID

SKRIPSI

diajukan untuk memenuhi sebagian syarat untuk memperoleh gelar Sarjana Komputer pada Program Studi S1 Rekayasa Perangkat Lunak

oleh

Fajar Muhammad Al-Hijri
NIM 1909473

PROGRAM STUDI REKAYASA PERANGKAT LUNAK
KAMPUS UPI DI CIBIRU
UNIVERSITAS PENDIDIKAN INDONESIA
2023

PENGARUH *REFACTORING CODE SMELLS* DENGAN *AUTOMATIC STATIC ANALYSIS TOOLS* TERHADAP PENGGUNAAN SUMBER DAYA PERANGKAT LUNAK SELULER ANDROID

Oleh
Fajar Muhammad Al-Hijri
NIM 1909473

Sebuah skripsi yang diajukan untuk memenuhi salah satu syarat memperoleh gelar Sarjana Komputer pada Program Studi S1 Rekayasa Perangkat Lunak

HALAMAN PENGESAHAN

FAJAR MUHAMMAD AL-HIJRI

PENGARUH *REFACTORING CODE SMELLS* DENGAN *AUTOMATIC STATIC ANALYSIS TOOLS* TERHADAP PENGGUNAN SUMBER DAYA PERANGKAT LUNAK SELULER ANDROID

disetujui dan disahkan oleh pembimbing:

Pembimbing I

Mochamad Iqbal Ardimansyah, S.T., M.Kom.

NIP. 920190219910328101

Pembimbing II

Indira Syawanodya, S.Kom., M.Kom.

NIP. 920190219920423201

Mengetahui

Ketua Program Studi Rekayasa Perangkat Lunak

Mochamad Iqbal Ardimansyah, S.T., M.Kom.

NIP. 920190219910328101

HALAMAN PERNYATAAN KEASLIAN SKRIPSI DAN PENYATAAN BEBAS PLAGIARISME

Dengan ini saya menyatakan bahwa skripsi dengan judul "Pengaruh *Refactoring Code Smells* Dengan *Automatic Static Analysis Tools* Terhadap Penggunaan Sumber Daya Perangkat Lunak Seluler Android" ini beserta seluruh isinya adalah benar-benar karya saya sendiri. Saya tidak melakukan penjiplakan atau pengutipan dengan cara-cara yang tidak sesuai dengan etika ilmu yang berlaku dalam masyarakat keilmuan. Atas pernyataan ini, saya siap menanggung risiko/sanksi apabila di kemudian hari ditemukan adanya pelanggaran etika keilmuan atau ada klaim dari pihak lain terhadap keaslian karya saya ini.

Bandung, April 2023

Yang membuat pernyataan,

Fajar Muhammad Al-Hijri

NIM. 1909473

HALAMAN UCAPAN TERIMA KASIH

Dengan mengucapkan puji dan syukur ke hadirat Allah SWT atas segala rahmat dan karunia, penulis ingin menyampaikan rasa terima kasih yang setinggi-tingginya atas dukungan dan bantuan yang diberikan dalam penyelesaikan skripsi ini yang berjudul "Pengaruh *Refactoring Code Smells* Dengan *Automatic Static Analysis Tools* Terhadap Penggunaan Sumber Daya Perangkat Lunak Seluler Android". Skripsi ini adalah salah satu syarat untuk mendapatkan gelar sarjana dan menyelesaikan studi di program studi Rekayasa Perangkat Lunak, Universitas Pendidikan Indonesia.

Dokumen ini memuat informasi lengkap tentang penelitian yang telah dilakukan oleh penulis. Kesuksesan penyelesaian skripsi ini tidak mungkin tercapai tanpa bantuan dan dorongan berbagai pihak yang turut memberikan bimbingan, dukungan, kritik, motivasi, dan saran kepada penulis selama proses penelitian dan penulisan dokumen. Oleh karena itu, penulis ingin mengucapkan terima kasih kepada:

1. Bapak Mochamad Iqbal Ardimansyah, S.T., M.Kom., selaku Dosen Pembimbing Skripsi Utama yang di bawah pengawasannya selalu memberikan bimbingan, saran, dan umpan balik yang bermanfaat.
2. Ibu Indira Syawanodya, S.Kom., M.Kom., selaku Dosen Pembimbing Skripsi Kedua yang di bawah pengawasannya selalu memberikan bimbingan, saran, dan umpan balik yang bermanfaat.
3. Ibu Dian Anggraini, S.ST., M.T., selaku Dosen Pembimbing Akademik yang memberikan bimbingan dan dukungan selama kuliah hingga pada saat penyusunan skripsi ini.
4. Kedua orang tua penulis, yaitu Bapak Ahmad Jukri Karim dan Ibu Fathiah yang selalu memberikan doa dan dukungan, baik secara moril maupun materi sehingga penulis dapat menyelesaikan skripsi ini dengan baik.
5. Teman-teman yang tidak dapat penulis sebutkan satu persatu yang memberikan saran, dukungan, dan motivasi kepada penulis.

Penulis menyadari bahwa dokumen skripsi ini masih terdapat banyak kekurangan. Oleh karena itu, penulis mengharapkan kritik dan saran yang bersifat

membangun agar tidak terjadi kesalahan yang sama pada saat melakukan penelitian di masa depan. Akhir kata, penulis memohon maaf yang sebesar-besarnya apabila terdapat kesalahan baik dari penelitian yang dilakukan maupun cara penulisan dalam dokumen skripsi ini.

Bandung, April 2023
Penulis,

Fajar Muhammad Al-Hijri
NIM. 1909473

ABSTRAK

Perangkat lunak memiliki keterbatasan sumber daya yaitu CPU dan memori. Investigasi penelitian lain mengungkapkan bahwa kode program buruk dapat berdampak pada penurunan kinerja yang berarti meningkatnya konsumsi pada CPU dan memori. Survei penelitian lain menunjukkan bahwa pengguna dapat menghapus pemasangan perangkat lunak karena adanya kinerja yang berlebihan (75,2%) sehingga perangkat lunak tidak berjalan dan konsumsi memori yang besar (42,6%). Kode program buruk disebabkan buruknya praktik penulisan kode dan implementasi yang menyebabkan pemeliharaan perangkat lunak jangka panjang dan berdampak negatif. Dampak negatif mengindikasikan dapat merusak pemeliharaan perangkat lunak dengan mengabaikan pelanggaran aturan yaitu *code smells* yang disebabkan gaya pemrograman yang buruk, kurangnya dokumentasi dan tingginya kompleksitas pada kode program. Sehingga solusi tersebut perlu adanya eksplorasi *code smells* dengan salah satu ASATs yang sudah digunakan oleh 85.000 organisasi bernama SonarQube dan dilanjutkan *refactoring code smells* tunggal dan kumulatif. Topik ini berkaitan dengan pemeliharaan perangkat lunak dengan tujuan untuk menganalisis *code smells* dan *refactoring* serta membandingkan setiap perangkat lunak Android versi orisinal dan versi *refactoring* dengan aspek yang diuji mencakup *Fixed Detection Ratio* (FDR), perubahan relatif, penggunaan CPU dan memori menggunakan pendekatan *Design Research Methodology* (DRM). *Code smells* yang diteliti mencakup *Blocker*, *Critical*, *Major*, *Minor*, HashMap *Usage*, *Member Ignoring Method*, dan *Slow Loop*. Hasil penelitian yang telah dilakukan membuktikan adanya penurunan intensitas *code smells* pada Calculator (60%), Todolist (71%), Openflood (93%) dan penurunan konsumsi penggunaan CPU yang signifikan pada *Member Ignoring Method* (-7,7%) dan *Critical* (-9,90%). Selain itu, penurunan konsumsi memori berdampak lebih signifikan pada setiap perangkat lunak versi *refactoring* tunggal maupun kumulatif.

Kata kunci: Pemeliharaan Perangkat Lunak; *Refactoring*; *Code Smells*; Penggunaan Sumber Daya; Android

ABSTRACT

Software has resource limitations, namely CPU and memory. Other research investigations have revealed that bad programming code can impact performance, resulting in increased CPU and memory consumption. Another survey has shown that users may uninstall software due to excessive performance issues (75.2%) resulting in the software crashes, and high memory consumption (42.6%). Poor programming code is caused by bad coding practices and implementation, leading to long-term software maintenance issues and negative impacts. Negative impacts indicate that ignoring rule violations, such as code smells caused by poor programming styles, lack of documentation, and high code complexity, can damage software maintenance. Therefore, a solution is needed to explore code smells using one of the ASATs named SonarQube, which is already used by 85,000 organizations, followed by single and cumulative code smell refactoring. This topic relates to software maintenance with the aim of analyzing code smells and refactoring, and comparing each Android software's original and refactored versions with aspects tested including Fixed Detection Ratio (FDR), relative change, CPU and memory usage using the Design Research Methodology (DRM) approach. The investigated code smells include Blocker, Critical, Major, Minor, HashMap Usage, Member Ignoring Method, and Slow Loop. The results of the conducted research prove a decrease in code smell intensity in Calculator (60%), Todolist (71%), and Openflood (93%), and significant decreases in CPU usage in Member Ignoring Method (-7.7%) and Critical (-9.90%). In addition, the decrease in memory consumption has a more significant impact on each software's single and cumulative refactored versions.

Keywords: Software Maintenance; Refactoring; Code Smells; Resource Usage; Android

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

DAFTAR LAMPIRAN

DAFTAR PUSTAKA

Al-Hijri, F. M. (2023). *ObreusDroid* (1.0). [*Online*]. Diakses dari https://zenodo.org/record/7665495.

Alkandari, M. A., Kelkawi, A., & Elish, M. O. (2021). An Empirical Investigation on the Effect of Code Smells on Resource Usage of Android Mobile Applications. *IEEE Access*, *9*, 61853–61863.

Amalfitano, D., Riccio, V., Tramontana, P., & Fasolino, A. R. (2020). Do Memories Haunt You? An Automated Black Box Testing Approach for Detecting Memory Leaks in Android Apps. *IEEE Access*, *8*, 12217–12231.

Amrit, C., & Meijberg, Y. (2018). Effectiveness of Test-Driven Development and Continuous Integration: A Case Study. *IT Professional*, *20*(1), 27–35.

Anwar, H., Pfahl, D., & Srirama, S. N. (2019). Evaluating the Impact of Code Smell Refactoring on the Energy Consumption of Android Applications. *2019 45th Euromicro Conference on Software Engineering and Advanced Applications (SEAA)*, 82–86.

Blasquez, I., & Leblanc, H. (2018). Experience in learning test-driven development: Space invaders project-driven. *Annual Conference on Innovation and Technology in Computer Science Education, ITiCSE*, 111–116.

Blessing, L. T. M., & Chakrabarti, A. (2009). *DRM, a Design Research Methodology*.

Bluman, A. G. (2012). *Elementary Statistics: A Step by Step Approach (8th ed.)*. McGraw-Hill.

Brian, C., & Jacob, W. (2007). Secure Programming with Static Analysis. In *Pearson*.

Carette, A., Younes, M. A. A., Hecht, G., Moha, N., & Rouvoy, R. (2017). Investigating the energy impact of Android smells. *2017 IEEE 24th International Conference on Software Analysis, Evolution and Reengineering (SANER)*, 115–126.

Choi, W., Necula, G., & Sen, K. (2013). Guided GUI testing of Android apps with minimal restart and approximate learning. *ACM SIGPLAN Notices*, *48*(10), 623–639.

Cruz, L., & Abreu, R. (2019). Improving Energy Efficiency Through Automatic Refactoring. *Journal of Software Engineering Research and Development*, *7*, 2.

Cruz, L., & Abreu, R. (2017). Performance-Based Guidelines for Energy Efficient Mobile Applications. *2017 IEEE/ACM 4th International Conference on Mobile Software Engineering and Systems (MOBILESoft)*, 46–57.

De Andrade Gomes, P. H., Garcia, R. E., Spadon, G., Eler, D. M., Olivete, C., & Correia, R. C. M. (2017). Teaching software quality via source code inspection tool. *Proceedings - Frontiers in Education Conference*, *FIE*, *2017-Octob*, 1–8.

Etemadi Someoliayi, K., Harrand, N. Y. M., Larsen, S., Adzemovic, H., Luong Phu, H., Verma, A., Madeiral, F., Wikstrom, D., & Monperrus, M. (2022). Sorald: Automatic Patch Suggestions for SonarQube Static Analysis Violations. *IEEE Transactions on Dependable and Secure Computing*, 1–17.

Fanta, R., & Rajlich, V. (1998). Reengineering object-oriented code. *Proceedings. International Conference on Software Maintenance (Cat. No. 98CB36272)*, 238–246.

Fowler, M. (2019). *Refactoring Improving the Design of Existing Code Second Edition*.

Gao, X., Gu, Z., Li, Z., Jamjoom, H., & Wang, C. (2019). Houdini's Escape: Breaking the Resource Rein of Linux Control Groups. *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*, 1073–1086.

Georgiou, N., Konstantinidis, A., & Papadopoulos, H. (2016). *Malware Detection with Confidence Guarantees on Android Devices* (pp. 407–418).

Guerra-Manzanares, A., & Välbe, M. (2022). Cross-device behavioral consistency: Benchmarking and implications for effective android malware detection. *Machine Learning with Applications*, *9*, 100357.

Habchi, S., Moha, N., & Rouvoy, R. (2021). Android code smells: From introduction to refactoring. *Journal of Systems and Software*, *177*, 110964.

Habchi, S., Rouvoy, R., & Moha, N. (2019). On the Survival of Android Code Smells in the Wild. *2019 IEEE/ACM 6th International Conference on Mobile*

Software Engineering and Systems (MOBILESoft), 87–98.

Hall, T., Zhang, M., Bowes, D., & Sun, Y. (2014). Some code smells have a significant but small effect on faults. *ACM Transactions on Software Engineering and Methodology*, *23*(4).

Hamdi, O., Ouni, A., AlOmar, E. A., O Cinneide, M., & Mkaouer, M. W. (2021). An Empirical Study on the Impact of Refactoring on Quality Metrics in Android Applications. *2021 IEEE/ACM 8th International Conference on Mobile Software Engineering and Systems (MobileSoft)*, 28–39.

Hecht, G., Moha, N., & Rouvoy, R. (2016). An empirical study of the performance impacts of Android code smells. *Proceedings of the International Conference on Mobile Software Engineering and Systems*, 59–69.

Huseynov, F. (2020). Understanding Usage Behavior of Different Mobile Application Categories Based on Personality Traits. *Interacting with Computers*, *32*(1), 66–80.

Ickin, S., Petersen, K., & Gonzalez-Huerta, J. (2017). *Why Do Users Install and Delete Apps? A Survey Study* (pp. 186–191).

Imbugwa, G. B., De Araújo, L. J. P., Khazeev, M., Enombe, E., Saliu, H., & Mazzara, M. (2021). A case study comparing static analysis tools for evaluating SwiftUI projects. *Journal of Physics: Conference Series*, *2134*(1).

ISO/IEC. (2016). ISO/IEC 25023:2016 Systems and software engineering — Systems and software quality requirements and evaluation (SQuaRE) — Measurement of quality in use. In *Iso* (Issue March, p. 41).

ISO/IEC. (2022). *INTERNATIONAL STANDARD ISO / IEC Software Engineering — Software Life Cycle Processes — Maintenance 14764*.

Kaur, S., Awasthi, L. K., & Sangal, A. L. (2021). A Brief Review on Multi-objective Software Refactoring and a New Method for Its Recommendation. *Archives of Computational Methods in Engineering*, *28*(4), 3087–3111.

Khanam, Z., & Ahsan, M. N. (2017). Evaluating the effectiveness of test driven development: Advantages and pitfalls. *International Journal of Applied Engineering Research*, *12*(18), 7705–7716.

Kwan Kim, D. (2017). Towards Performance-Enhancing Programming for Android Application Development. *International Journal of Contents*, *13*(4), 39–46.

Lacerda, G., Petrillo, F., Pimenta, M., & Guéhéneuc, Y. G. (2020). Code smells and refactoring: A tertiary systematic review of challenges and observations. *Journal of Systems and Software*, *167*.

Lee, J., Raja, A. V., & Gao, D. (2019). SplitSecond: Flexible Privilege Separation of Android Apps. *2019 17th International Conference on Privacy, Security and Trust (PST)*, 1–10.

Lenarduzzi, V., Saarimäki, N., & Taibi, D. (2020). Some SonarQube issues have a significant but small effect on faults and changes. A large-scale empirical study. *Journal of Systems and Software*, *170*, 110750.

Mannan, U. A., Ahmed, I., Almurshed, R. A. M., Dig, D., & Jensen, C. (2016). Understanding code smells in android applications. *Proceedings - International Conference on Mobile Software Engineering and Systems, MOBILESoft 2016*, 225–236.

Marcilio, D., Bonifacio, R., Monteiro, E., Canedo, E., Luz, W., & Pinto, G. (2019). Are static analysis violations really fixed? a closer look at realistic usage of sonarqube. *IEEE International Conference on Program Comprehension*, *2019-May*, 209–219.

Marcilio, D., Furia, C. A., Bonifácio, R., & Pinto, G. (2020). SpongeBugs: Automatically generating fix suggestions in response to static code analysis warnings. *Journal of Systems and Software*, *168*, 110671.

Mitra, D., Arora, M., Rakhra, M., Kumar, C. R., Reddy, M. L., Reddy, S. P. K., Kumar, C., & Shabaz, M. (2022). A Hybrid Framework to Control Software Architecture Erosion for Addressing Maintenance Issues | Annals of the Romanian Society for Cell Biology. *Annals of the Romanian Society for Cell Biology*, *25*(4), 2974–2021.

Nanthaamornphong, A., & Carver, J. C. (2018). Test-Driven Development in HPC Science: A Case Study. *Computing in Science & Engineering*, *20*(5), 98–113.

Nass, M., Alegroth, E., & Feldt, R. (2019). Augmented testing: Industry feedback to shape a new testing technology. *Proceedings - 2019 IEEE 12th International Conference on Software Testing, Verification and Validation Workshops, ICSTW 2019*, 176–183.

Nikolic, D., Stefanovic, D., Dakic, D., Sladojevic, S., & Ristic, S. (2021). Analysis

of the Tools for Static Code Analysis. *2021 20th International Symposium INFOTEH-JAHORINA, INFOTEH 2021 - Proceedings*, *March*, 17–19.

Oliveira, J., Viggiato, M., Santos, M., Figueiredo, E., & Marques-Neto, H. (2018). *An Empirical Study on the Impact of Android Code Smells on Resource Usage*. 314–359.

Oo, T., Liu, H., & Nyirongo, B. (2018). Dynamic Ranking of Refactoring Menu Items for Integrated Development Environment. *IEEE Access*, *6*, 76025–76035.

Palomba, F., Di Nucci, D., Panichella, A., Zaidman, A., & De Lucia, A. (2017). Lightweight detection of Android-specific code smells: The aDoctor project. *2017 IEEE 24th International Conference on Software Analysis, Evolution and Reengineering (SANER)*, 487–491.

Pantiuchina, J., Zampetti, F., Scalabrino, S., Piantadosi, V., Oliveto, R., Bavota, G., & Penta, M. Di. (2020). Why Developers Refactor Source Code. *ACM Transactions on Software Engineering and Methodology*, *29*(4), 1–30.

Pombo, N., & Martins, C. (2021). Test driven development in action: Case study of a cross-platform web application. *EUROCON 2021 - 19th IEEE International Conference on Smart Technologies, Proceedings*, *July*, 352–356.

Rasool, G., & Arshad, Z. (2017). A Lightweight Approach for Detection of Code Smells. *Arabian Journal for Science and Engineering*, *42*(2), 483–506.

Romano, S., Zampetti, F., Baldassarre, M. T., Di Penta, M., & Scanniello, G. (2022). Do Static Analysis Tools Affect Software Quality when Using Test-driven Development? *International Symposium on Empirical Software Engineering and Measurement*, 80–91.

Şanlıalp, İ., Öztürk, M. M., & Yiğit, T. (2022). Energy Efficiency Analysis of Code Refactoring Techniques for Green and Sustainable Software in Portable Devices. *Electronics*, *11*(3), 442.

Shapiro, A. S. S., & Wilk, M. B. (1965). Biometrika Trust An Analysis of Variance Test for Normality (Complete Samples) Published by : Oxford University Press on behalf of Biometrika Trust Stable. *Biometrika*, *52*(3), 591–611.

Sonarqube. (t.t.). *Install the server*. [*Online*]. Diakses dari https://docs.sonarqube.org/latest/setup-and-upgrade/install-the-server.

Sonarsource. (t.t.). *Java static code analysis*. [*Online*]. Diakses dari https://rules.sonarsource.com/java/type/Code Smell.

Stefanović, D., Nikolić, D., Dakić, D., Spasojević, I., & Ristić, S. (2020). Static code analysis tools: A systematic literature review. *Annals of DAAAM and Proceedings of the International DAAAM Symposium*, *31*(1), 565–573.

Sutino, Q. L., Maryamah, & Rochimah, S. (2018). Android Quality Measurement in Metrics and Findings. *2018 Electrical Power, Electronics, Communications, Controls and Informatics Seminar (EECCIS)*, 365–370.

Telemaco, U., Oliveira, T., Alencar, P., & Cowan, D. (2020). A Catalogue of Agile Smells for Agility Assessment. *IEEE Access*, *8*, 79239–79259.

Venters, C. C., Capilla, R., Betz, S., Penzenstadler, B., Crick, T., Crouch, S., Nakagawa, E. Y., Becker, C., & Carrillo, C. (2018). Software sustainability: Research and practice from a software architecture viewpoint. *Journal of Systems and Software*, *138*, 174–188.

Verdecchia, R., Aparicio Saez, R., Procaccianti, G., & Lago, P. (2018). *Empirical Evaluation of the Energy Impact of Refactoring Code Smells*. 365–345.

Wang, J., Huang, Y., Wang, S., & Wang, Q. (2022). Find Bugs in Static Bug Finders. *IEEE International Conference on Program Comprehension*, *2022-March*, 516–527.

Wen, E., Cao, J., Shen, J., & Liu, X. (2018). Fraus: Launching Cost-efficient and Scalable Mobile Click Fraud Has Never Been So Easy. *2018 IEEE Conference on Communications and Network Security (CNS)*, 1–9.

Wohlin, C., Runeson, P., Höst, M., Ohlsson, M. C., Regnell, B., & Wesslén, A. (2012). Experimentation in Software Engineering. In *Formal Specification Techniques for Engineering Modular C Programs* (Vol. 6). Springer Berlin Heidelberg.