UNIVERSIDADE ESTADUAL DE FEIRA DE SANTANA
Autorizada pelo Decreto Federal nº 77.496 de 27/04/76
Recredenciamento pelo Decreto nº17.228 de 25/11/2016
PRÓ-REITORIA DE PESQUISA E PÓS-GRADUAÇÃO
COORDENAÇÃO DE INICIAÇÃO CIENTÍFICA

XXVI SEMINÁRIO DE INICIAÇÃO CIENTÍFICA DA UEFS
SEMANA NACIONAL DE CIÊNCIA E TECNOLOGIA - 2022

Investigando o impacto da adoção de diferentes heurísticas e thresholds na detecção automática de anomalias de código

Adlla Katarine Aragão Cruz Passos[1]; José Amancio Macedo Santos[2]

1. Bolsista PIBIC/CNPq, Graduando em Engenharia de Computação, Universidade Estadual de Feira de Santana, e-mail: adllakatarine@gmail.com
2. Orientador, Departamento de Tecnologia, Universidade Estadual de Feira de Santana, e-mail: zeamancio@uefs.br

PALAVRAS-CHAVE: code smells; thresholds; detecção automática.

INTRODUÇÃO

A análise estática automatizada de código se tornou um importante pilar das práticas modernas de revisão de código em conjunto com testes e revisão manual (Beller, 2016), tendo como principais objetivos encontrar anomalias (cheiros de código ou *code smells*) e realizar melhorias no código em termos de legibilidade, comentários, consistência e remoção de código morto (Bacchelli; Bird, 2013). A estratégia utilizada neste tipo de análise é o uso de limites métricos. Contudo ajustar esses limites é difícil e quando incorreto, podem conduzir o resultado da análise a muitos falsos positivos e/ou negativos. Com base nisso, o objetivo desta pesquisa é realizar um estudo empírico para compreender o impacto que a adoção de diferentes *thresholds* possuem sobre detecção automática de anomalias de código.

MATERIAL E MÉTODOS OU METODOLOGIA (ou equivalente)

Muitas técnicas para derivar limites métricos foram estudadas, dentre elas: identificando limites para um conjunto de métricas analisando as propriedades estatísticas (Ferreira, 2012); derivando limites relativos com base em uma análise estatística de um corpus de software (Oliveira, 2014). Contudo, as técnicas de Dósea propondo atribuição de papéis de design se mostraram muito promissoras. Ele propõe duas técnicas, sendo uma para derivar limiares métricos genéricos de um benchmark de sistemas desenvolvidos com decisões de projeto semelhantes e outras para derivar limites métricos distintos para cada função de design de classe, além de usar a proposta da outra técnica (Dósea, 2021).

A partir da técnicas de Dósea, três ferramentas de código aberto foram desenvolvidas por ele: *DesignRoleMiner* que implementa uma heurística baseada em palavras-chave para atribuir papéis de design para as classes de um software e que posteriormente foi integrada com sua outra ferramenta *ThresholdTool*, que deriva limiares métricos usando as duas técnicas propostas e mais três de outros autores; e por fim a *ContextSmell* que

usa como entrada os arquivos dos limites gerados pelo *ThresholdTool* e gera uma lista de odores de código identificados pelos limites derivados de cada técnica. Os code smells identificados pela sua última ferramenta são *Long Method, Complex Method, High Efferent Coupling e Many Parameters*.

Outras ferramentas de detecção automática de anomalias de código foram analisadas, sendo elas o *PMD, Checkstyle, SonarQube, Jdeodorant, DECOR, FindBugs e Repository Miner*. Apenas quatro foram testadas (*PMD, Checkstyle, SonarQube, Jdeodorant*) pois, infelizmente, houve dificuldade na instalação e execução das outras. Das quatro ferramentas com êxito na execução, três foram escolhidas para uma análise e comparação de resultados, contudo, apesar do sucesso em relação a execução e resultados dessas ferramentas, é quase inviável seguir com testes de comparação entre elas levando em consideração seus poucos code smells em comum.

RESULTADOS E/OU DISCUSSÃO (ou Análise e discussão dos resultados)

Após todos os testes com as ferramentas de análise estática de código encontradas, foi decidido seguir os estudos com as ferramentas da Tese de Doutorado de Marcos Dósea (Dósea, 2021). Quanto ao objetivo do estudo, foi definido que um conjunto de softwares seriam analisados em diferentes contextos (linhas de código, número de commits, número de contribuidores e tempo de desenvolvimento - alto ou baixo). Este estudo baseado em contextos já foi proposto e analisado em outros estudos, onde apresentou resultados positivos. Por fim, também será feita a união desses contextos para uma análise mais profunda das técnicas.

O dataset escolhido é de um mestrado em fase de conclusão de Elivelton Cerqueira, em que já possui 420 softwares separados nos quatro contextos citados (De Jesus, 2021). Logo, um teste com esses softwares já puderam ser iniciados e um script já foi desenvolvido, em *Python*, para o pré-processamento dos dados gerados a partir das ferramentas *ThresholdTool* e *ContextSmell*.

A análise comparativa entre as ferramentas observadas também pode ser considerada como uma importante contribuição deste trabalho. As observações representam um insumo essencial para avanços na área, uma vez que evidencia aspectos importantes que precisam ser considerados para adoção do conceito de code smells na prática de desenvolvimento de software. Devido ao caráter exploratório do trabalho, resultados desta natureza são importantes achados para que os objetivos de comparação dos limiares adotados por diferentes ferramentas de detecção de smells possam ser, então, avaliados de forma mais ampla.

A **Tabela 1** apresenta mais informações a respeito dos testes com as sete *ferramentas de análise* encontradas nos artigos.

Ferramenta	Download	Dificuldades	Resultados
PMD	Plug-in do Eclipse ou download pelo site.	Não houve.	Gera arquivos em vários formatos.
Checkstyle	Plug-in do Eclipse ou download pelo github.	Não houve.	Gráficos e lista no console sem opção de download.
SonarQube	Plug-in do Eclipse, versão na nuvem (***SonarCloud***) ou download pelo site.	Executar o plug-in e instalar a ferramenta na máquina. Não houve dificuldades com o *SonarCloud*.	O *SonarCloud* possui disponibilidade para visualização na própria plataforma.
Jdeodorant	Plug-in do Eclipse.	Não houve.	Gera arquivos em formato txt.
DECOR	Download do **Ptidej**, que contém o DECOR.	Inicialização da ferramenta.	Disponibilidade para visualização na própria plataforma.
FindBugs	Plug-in do Eclipse ou download pelo site.	Não gerou resultados.	Gera arquivos em formato xml.
Repository Miner	Download pelo github.	Inicialização da ferramenta.	Gera arquivos.

Tabela 1. Detalhes das tentativas de instalação/execução das ferramentas.

CONSIDERAÇÕES FINAIS (ou Conclusão)

A partir do estudo a respeito de detecção automática de anomalias de código e das técnicas utilizadas, foi possível encontrar uma técnica promissora com duas ferramentas para que fosse possível planejar e iniciar um estudo em conjunto com outro estudo sobre contextos. Utilizando um dataset de um mestrado contendo 420 softwares já separados em contextos (linhas de código, número de commits, número de contribuidores e tempo de desenvolvimento - alto ou baixo) e um script de pré-processamento dos dados já implementado será possível dar continuidade e concluir o projeto.

REFERÊNCIAS

FOWLER, M.: Refactoring: Improving the Design of Existing Code. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA (1999)

LANZA, M., MARINESCU, R., DUCASSE, S.: Object-Oriented Metrics in Practice. Springer, Secaucus, NJ, USA (2005)

FERREIRA, K.A., BIGONHA, M., BIGONHA, R.D., MENDES, L.F., & ALMEIDA, H.C. (2012). Identifying thresholds for object-oriented software metrics. J. Syst. Softw., 85, 244-257.

BACCHELLI, A.; BIRD, C. Expectations, outcomes, and challenges of modern code review. In: International Conference on Software Engineering (ICSE). Piscataway, NJ, USA: IEEE, 2013. p. 712–721. ISBN 978-1-4673-3076-3.

P. OLIVEIRA, M. T. VALENTE and F. P. LIMA, "Extracting relative thresholds for source code metrics," 2014 Software Evolution Week - IEEE Conference on Software Maintenance, Reengineering, and Reverse Engineering (CSMR-WCRE), 2014, pp. 254-263.

LAVAZZA, L.; MORASCA, S. An empirical evaluation of distribution-based thresholds for internal software measures. In: International Conference on Predictive Models and Data Analytics in Software Engineering (PROMISE). New York, NY, USA: ACM, 2016. p. 6:1–6:10. ISBN 978-1-4503-4772-3.

BELLER, M. et al. Analyzing the state of static analysis: A large-scale evaluation in open source software. In: Conference on Software Analysis, Evolution, and Reengineering (SANER). [S.l.]: IEEE, 2016. v. 1, p. 470–481.

SOBRINHO, E. V. d. P.; LUCIA, A. D.; MAIA, M. d. A. A systematic literature review on bad smells — 5 w's: which, when, what, who, where. Transactions on Software Engineering (TSE), IEEE, p. 1–1, 2018. ISSN 0098-5589.

DÓSEA, Marcos Barbosa. Design-sensitive metric thresholds based on design roles. 2021. 155 f. Tese (Doutorado) - Curso de Ciência da Computação, Instituto de Computação, Universidade Federal da Bahia, Salvador, Bahia, 2021.

DE JESUS, Elivelton Cerqueira. O impacto de fatores contextuais na incidência de code smells: um estudo exploratório baseado em mineração de repositórios de software. 46 f. Qualificação (Mestrado) - Programa de Pós Graduação em Ciência da Computação (PGCC), Universidade Estadual de Feira de Santana, Feira de Santana, Bahia, 2021.