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

XXIV SEMINÁRIO DE INICIAÇÃO CIENTÍFICA DA UEFS
SEMANA NACIONAL DE CIÊNCIA E TECNOLOGIA - 2020

INVESTIGANDO O IMPACTO DA SUBJETIVIDADE NA DETECÇÃO DE ANOMALIAS DE CÓDIGO COM BASE EM ESTUDOS EXPERIMENTAIS

Carlos André Pereira Tinin[1] e José Amancio Macedo Santos[2]
1. Bolsista PIBIC/FAPESB, Graduando em Engenharia da Computação, Universidade Estadual de Feira de Santana, e-mail: carlostinin@gmail.com
2. Orientador, Departamento de Tecnología, Universidade Estadual de Feira de Santana, e-mail: zeamancio@uefs.br

PALAVRAS-CHAVE: projeto de software; anomalias de código; revisão sistemática.

INTRODUÇÃO

O conceito de anomalias de código tem sido adotado por engenheiros de software, desde a década de 90, como uma forma de identificar problemas em projetos de software. O termo foi cunhado por Riel (1996). Outros importantes autores adotam termos diferentes, mas com o mesmo propósito. Fowler (1999) apresenta um catálogo do que denomina *bad smell* (mau cheiro) para representar problemas em código. Sua apresentação é baseada na percepção dos desenvolvedores do que é um problema de projeto de software. Lanza e Marinescu (2005) adotam o termo *disharmony* (desarmonia) para descrever tais problemas. Entretanto, neste caso, os autores propõem heurísticas baseadas em métricas de software para detecção automática das desarmonias. Neste plano de trabalho adotaremos indistintamente o termo anomalia de código como referência a mau cheio (*bad smell*) e desarmonia (*disharmony*).

A partir do trabalho de Lanza e Marinescu (2005), muitos trabalhos têm sido desenvolvidos devido à possibilidade e identificação de anomalias de forma automática. Em especial o desenvolvimento de ferramentas de detecção automática (CHEN e JIANG 2017; ARCELLI, CORTELLESSA e POMPEO 2018; MOHA e GUÉHÉNEUC Y.-G. 2007) e estudos de correlação entre anomalias e outros atributos de software, como padrões de projetos, manutenibilidade ou bugs, por exemplo (WALTER, FONTANA e FERME, 2018; YAMASHITA e MOONEN, 2013; FONTANA et al., 2015). Um amplo volume de dados está disponível atualmente para a realização de estudos de correlação pela disponibilidade de repositórios de software livre, como SourceForge (https://sourceforge.net/) e o GitHub (https://github.com/). Dessa forma um extenso volume de estudos empíricos têm ajudado a comunidade a compreender o impacto da adoção do conceito de anomalias de código na prática do desenvolvimento de software.

Como resultado de uma ampla gama de estudos correlacionando anomalias de código a atributos de software, um interessante fenômeno passou a ser observado: alguns trabalhos identificaram que o conceito de anomalias não está necessariamente ligado a

problemas de software, como propõe a teoria. Por exemplo, Sjoberg et al. (2013) investigaram o relacionamento entre anomalias de código e esforço de manutenção. Os autores observaram que nenhumas das anomalias estava associada ao esforço de manutenção. Macia et al. (2012) investigaram o relacionamento entre anomalias e problemas de arquitetura de software que surgem durante a evolução dos sistemas. Eles notaram que muitas anomalias estudadas não estavam relacionadas a problemas arquiteturais. Yamashita (2013) investigou a agregação de anomalias e concluíram que elas não são bons indicadores de manutenibilidade.

Santos et al. (2018) sintetizam este conhecimento e apontam a importância da subjetividade como uma das causas da inconsistência dos estudos empíricos sobre anomalias de código com a teoria sobre o tema. Importante destacar que a subjetividade é desconsiderada nos estudos que consideram apenas a detecção automática. Os autores identificaram a necessidade de aprofundar a discussão sobre a relevância da subjetividade na detecção de anomalias de código. Alguns trabalhos têm sido elaborados neste sentido (PALOMBA et al., 2018; SANTOS, ROCHA-JUNIOR e MENDONÇA, 2017; SANTOS e MENDONÇA, 2015), mas o tema ainda carece de um debate mais amplo, uma vez que evidencia um mau direcionamento em um amplo conjunto de pesquisas baseadas exclusivamente na detecção automática. Muito esforço de pesquisa pode estar sendo desperdiçado.

Dessa forma, torna-se necessária uma análise sistemática do material existente sobre as anomalias de código, possibilitando a elaboração de uma síntese desse material para solucionar o problema da falta de compreensão sobre o assunto. Posteriormente, realizar experimentos envolvendo hipóteses identificadas durante a elaboração da síntese.

MATERIAL E MÉTODOS OU METODOLOGIA (ou equivalente)

No primeiro momento, para aquisição do conhecimento sobre o assunto, foram reunidos materiais relevantes sobre o tema. Em seguida, foi feita a leitura e análise de cada um desses trabalhos, para então classificá-los de acordo a natureza da pesquisa. Durante a análise desses materiais, surgiu a problemática do estudo de *test smells* (que são anomalias de código que se encontram em códigos de testes de software) utilizando abordagens de estudo dos *code smells*.

O trabalho de Palomba et al. (2016), serviu como guia para o estudo dos *test smells*, na sua pesquisa, através do detector de anomalias de teste, desenvolvido por Bavota et al. (2015), ele realiza um experimento observando algumas métricas de análise de software sobre um *dataset* de projetos de código aberto existentes. Posteriormente, ele responde quatro perguntas conclusivas sobre os *test smells* e sua difusão em projetos de software.

O objetivo desse trabalho seria a repetição desse experimento em um ambiente de testes diferente, utilizando os projetos do repositório *qualita corpus* e analisando mais métricas de análise de características de software do que foram utilizadas no trabalho de Palomba et al. (2016).

A segunda parte do projeto foi a experimentação, e para conseguir um resultado mais amplo com respostas mais decisivas, seria necessário realizar o experimento com mais ferramentas e dados do que foram utilizados pelos outros pesquisadores. Foram

reunidas as ferramentas Evosuite e Randoop para geração automática de classes de testes, e o Test Smell Detector para análise das características de software e a identificação das anomalias de testes.

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

Devido a problemas que surgiram durante o projeto, não foi possível a realização do experimento, e foi finalizada a pesquisa.

CONSIDERAÇÕES FINAIS (ou Conclusão)

Apesar do trabalho não ter sido concluído, essa análise é importante e necessária para o avanço do conhecimento da engenharia de software em geral, e as pesquisas relacionadas a anomalias de códigos de testes.

REFERÊNCIAS

ARCELLI, D.; CORTELLESSA, V.; POMPEO, D. D. Performance-driven software model refactoring. Information and Software Technology, v. 95, p. 366 – 397, 2018. ISSN 0950-5849.

ARCELLI, F. et al. Design pattern detection in java systems: A dynamic analysis based approach. In: MACIASZEK, L. A.; GONZÁLEZ-PÉREZ, C.; JABLONSKI, S. (Ed.). Evaluation of Novel Approaches to Software Engineering. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010. p. 163–179. ISBN 978-3-642-14819-4.

CHEN, B.; JIANG, Z. M. J. Characterizing and detecting anti-patterns in the logging code. In: Proceedings of the 39th International Conference on Software Engineering. Piscataway, NJ, USA: IEEE Press, 2017. (ICSE '17), p. 71–81. ISBN 978-1-5386-3868-2.

FONTANA, Francesca Arcelli; FERME, Vincenzo; ZANONI, Marco. Poster: Filtering Code Smells Detection Results. 2015 Ieee/acm 37th Ieee International Conference On Software Engineering, [s.l.], p.803-804, maio 2015. IEEE.

FONTANA, Francesca Arcelli et al. Comparing and experimenting machine learning techniques for code smell detection. Empirical Software Engineering, [s.l.], v. 21, n. 3, p.1143-1191, 6 jun. 2015. Springer Nature.

FOWLER, M. Refactoring: improving the design of existing code. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 1999. ISBN 0-201-48567-2.

G. Bavota, A. Qusef, R. Oliveto, A. De Lucia, and D. Binkley. Are test smells really harmful? an empirical study. Empirical Software Engineering, 20(4):1052–1094, 2015.

LANZA, M.; MARINESCU, R. Object-Oriented Metrics in Practice. Secaucus, NJ, USA: Springer-Verlag New York, Inc., 2005. ISBN 3540244298.

MACIA, Isela et al. On the Relevance of Code Anomalies for Identifying Architecture Degradation Symptoms. 2012 16th European Conference On Software Maintenance And Reengineering, [s.l.], p.277-286, mar. 2012. IEEE.

MOHA, N.; GUÉHÉNEUC Y.-G., Y.-G. Ptidej and decor: Identification of design patterns and design defects. In: Companion to the 22Nd ACM SIGPLAN Conference on Object-oriented Programming Systems and Applications Companion. New York,

NY, USA: ACM, 2007. (OOPSLA ’07), p. 868–869. ISBN 978-1-59593-865-7.
PALOMBA, Fabio et al. How do community smells influence code smells? Proceedings Of The 40th International Conference On Software Engineering Companion Proceeedings - Icse '18, [s.l.], p.240-241, 2018. ACM Press.
PALOMBA, Fábio; NUCCI, Dario di; PANICHELLA, Annibale; OLIVETO, Rocco; LUCIA, Andrea de. On the Diffusion of Test Smells in Automatically Generated Test Code: An Empirical Study. Proceedings Of The 9Th International Workshop On Search-Based Software Testing. P. 5-14. Maio 2016.
RIEL, A. J. Object-Oriented Design Heuristics. 1st. ed. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 1996.
SANTOS, J. A. et al. The problem of conceptualization in god class detection: agreement, strategies and decision drivers. Journal of Software Engineering Research and Development (JSERD), v. 2, n. 11, p. 1–33, 2014.
SANTOS, J. A. et al. A systematic review on the code smell effect. Journal Of Systems And Software, [s.l.], v. 144, p.450-477, out. 2018. Elsevier BV.
SANTOS, J. A.; MENDONÇA, M. G. de. Exploring decision drivers on god class detection in three controlled experiments. Proceedings Of The 30th Annual Acm Symposium On Applied Computing - Sac '15, [s.l.], p.1472-1479, 2015. ACM Press.
SANTOS, J. A. M; ROCHA-JUNIOR, J. B.; MENDONÇA, M. G. de. Investigating factors that affect the human perception on god class detection: an analysis based on a family of four controlled experiments. Journal Of Software Engineering Research And Development, [s.l.], v. 5, n. 1, 28 nov. 2017. Springer Nature.
SJØBERG, D. et al. Quantifying the effect of code smells on maintenance effort. IEEE Transactions on Software Engineering, v. 39, n. 8, p. 1144–1156, 2013. SJØEBERG, D. et al. A survey of controlled experiments in software engineering. Ieee Transactions On Software Engineering, [s.l.], v. 31, n. 9, p.733-753, set. 2005. Institute of Electrical and Electronics Engineers (IEEE).
SJOBERG, D.; DYBA, T.; JORGENSEN, M. The Future of Empirical Methods in Software Engineering Research. Future Of Software Engineering (fose '07), [s.l.], p.358-378, maio 2007.
WALTER, Bartosz; FONTANA, Francesca Arcelli; FERME, Vincenzo. Code smells and their collocations: A large-scale experiment on open-source systems. Journal Of Systems And Software, [s.l.], v. 144, p.1-21, out. 2018. Elsevier BV.
YAMASHITA, Aiko; MOONEN, Leon. Do developers care about code smells? An exploratory survey. 2013 20th Working Conference On Reverse Engineering (wcre), [s.l.], p.242-251, out. 2013. IEEE.
YAMASHITA, Aiko. How Good Are Code Smells for Evaluating Software Maintainability? Results from a Comparative Case Study. 2013 Ieee International Conference On Software Maintenance, [s.l.], p.566-571, set. 2013. IEEE.