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

XXIII SEMINÁRIO DE INICIAÇÃO CIENTÍFICA DA UEFS
SEMANA NACIONAL DE CIENTÍFICA E TECNOLÓGICA - 2019

INVESTIGANDO A IMPLEMENTAÇÃO DE CONCEITOS DE COESÃO E ACOPLAMENTO NA PRÁTICA DE DESENVOLVIMENTO DE SOFTWARE ORIENTADO A OBJETOS

Gabriel Yago de O. Moreira[1]; José A. M. Santos [2]

1. Bolsista PROBIC, Graduando em Engenharia de Computação, Universidade Estadual de Feira de Santana, e-mail: gyagom@gmail.com

2. Orientador, Departamento de Tecnologia, Universidade Estadual de Feira de Santana, e-mail: zeamancio@uefs.br

PALAVRAS-CHAVE: Anomalias de código. Projeto de software, Coesão e acoplamento.

INTRODUÇÃO

O conceito de anomalia de código vem sendo aceito desde a década de 90 como um indicativo de problemas em projetos de software. Os trabalhos de Riel (1996), Fowler (1999) e Lanza e Marinescu (2005) representam uma teoria sobre o tema. Essa teoria é formada por uma ampla discussão sobre heurísticas que direcionam desenvolvedores para a detecção de anomalias (FOWLER, 1999). Estas heurísticas baseiam-se em princípios do paradigma da Orientação a Objetos (OO), tais como coesão, acoplamento, herança, etc. Os autores consideram que anomalias de código surgem quando desenvolvedores desconsideram estes princípios durante o processo de elaboração de um software.

Na prática, o surgimento de anomalias de código está diretamente relacionado a aspetos humanos. Uma vez que o processo de desenvolvimento de software ainda não é manufaturado (na imensa maioria dos casos), os desenvolvedores são responsáveis por codificar, muitas vezes aplicando os princípios do paradigma OO de forma inadequada. O controle sobre as ações dos desenvolvedores representa um custo significativo dentro deste processo. Uma forma de mitigar este problema é compreender que aspectos levam ao desenvolvedor desconsiderar os princípios da OO durante as atividades de programação. Essa compreensão requer em avaliação empírica (SCHUMACHER et al., 2010; ZHANG; HALL; BADDOO, 2011; SJØBERG et al., 2013). Alguns estudos têm sido realizados visando compreender a relação entre fatores humanos e a existência de anomalias de código (YAMASHITA; MOONEN, 2013; PALOMBA et al., 2014; FU; SHEN, 2015; SANTOS; ROCHA-JUNIOR; MENDONÇA, 2017).

MATERIAL E MÉTODOS OU METODOLOGIA

O presente estudo foi desenvolvido por meio de uma sequência de atividades de acordo com o cronograma previsto no projeto. Primeiramente, foram convidados alunos dos cursos de graduação e mestrado na área de informática da Universidade Estadual de Feira de Santana –UEFS para participarem voluntariamente da pesquisa. Esses alunos foram orientados a assistirem a uma breve apresentação sobre a importância do experimento

controlado em Engenharia de Software e explicação do método do experimento controlado. Em seguida foram encaminhados ao preenchimento do formulário de consentimento e caracterização, que continuamente seguiu-se da leitura de um resumo sobre coesão e acoplamento. Na leitura do resumo, foram preservados os contextos acerca dos conceitos estudados com a finalidade de evitar ideias enviesadas por parte dos participantes e ou influenciá-los, portanto, foi dado ênfase aos exemplos.

A primeira fase do experimento foi conduzida durante o segundo semestre de 2018. Os alunos foram perguntados se queriam participar de um experimento prático para implementação de coesão e acoplamento no código fonte em um sistema de PDV, na qual tivemos nove participantes. Antes de executarmos o experimento pela primeira vez, um estudante cursando a mesma graduação que o primeiro havia executado um piloto durante o segundo semestre de 2018. A partir desse piloto, acrescentamos algumas configurações como adicionar um breve comentário sobre sua descrição em cada classe. Além disso, verificar se duas horas seria tempo suficiente para executar o experimento. Conseguinte na segunda etapa, realizada no primeiro semestre de 2019, os alunos também estavam cientes de participar de um experimento para avaliar a coesão e o acoplamento, onde participaram quatro estudantes. Ambos os experimentos duraram cerca de duas horas com um total de treze participantes.

ANÁLISE DOS RESULTADOS

Foram definidas doze questões de acordo com o livro de Larman para corrigir as implementações feitas a partir dos sujeitos.

Objetivo 1, comparou-se o sucesso dos sujeitos em relação a sua experiência em algum projeto, no que tange a quantidade de anos. Para a análise, identificamos dois grupos de sujeitos que se classificaram sobre a experiência em algum projeto: três anos ou mais para o primeiro grupo e menos de três anos para o outro grupo. A análise da experiência descrita acima examina se a experiência influencia a taxa de resposta correta ou errada de coesão e acoplamento.

O objetivo 2, comparou-se o sucesso dos assuntos relativos ao conhecimento. Para a análise, identificamos dois grupos de sujeitos que classificaram seus conhecimentos de coesão e acoplamento: especialista ou bom grupo, baixo ou nenhum outro.

A análise para o conhecimento descrito examina se o conhecimento influencia a taxa de resposta correta ou incorreta para coesão e acoplamento.

Descobrimos que entre a boa e baixa experiência dos participantes não faz uma grande diferença na obtenção de mais respostas corretas. Embora, como era esperado, os participantes com mais tempo de experiência foram um pouco melhores que os participantes com pouco ou nenhum tempo de experiência em algum projeto.

A Figura 1 e a Figura 2 mostram dois gráficos de barras separados por conhecimento relacionado as respostas corretas e erradas para acoplamento e coesão. Nas questões gerais, a taxa do sujeito para as respostas corretas e erradas foi entre oito por cento a noventa e dois por cento para o acoplamento e para a coesão, a taxa estava entre trinta por cento a setenta por cento. No que diz respeito aos sujeitos com conhecimento especialista/bom, a coesão parece ter uma taxa de acerto mais alta, enquanto o acoplamento tem uma taxa de erro mais alta. O mesmo pode ser visto para indivíduos com conhecimento baixo/nenhum. A taxa de diferença entre conhecimento

especialista/bom e baixo/nenhum é de quatro por cento para respostas corretas e erradas, no acoplamento é cinco por cento para o correto e errado

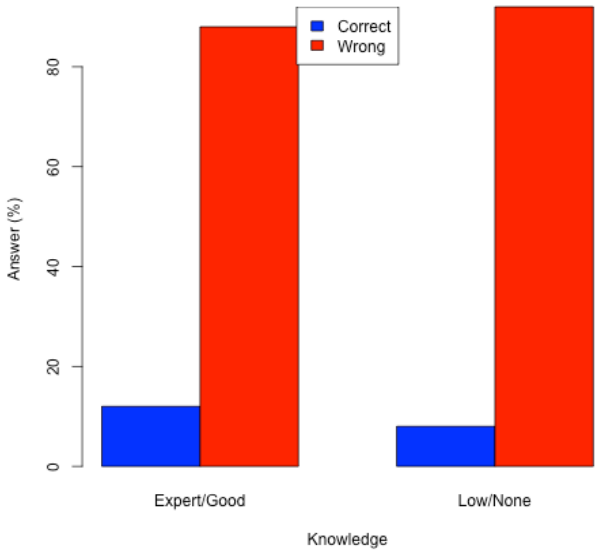

Figura 1: Média das respostas corretas e erradas da taxa dos participantes com conhecimento de acoplamento

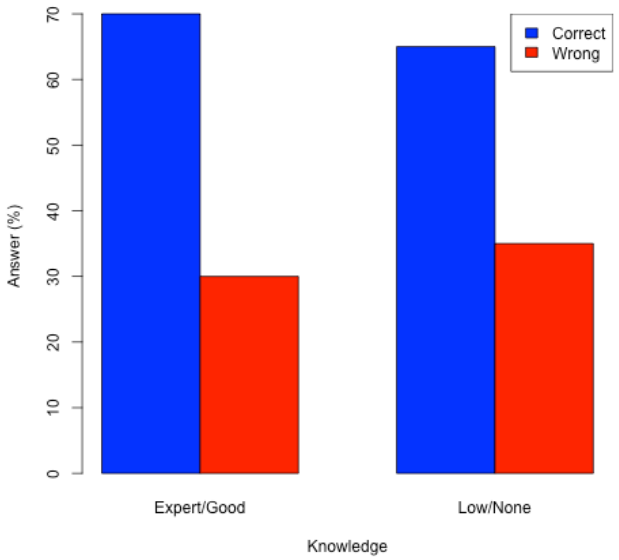

Figura 2: Média das respostas corretas e erradas da taxa dos participantes no conhecimento de coesão

Constatamos que, entre quem era especialista/bom e quem tinha baixo conhecimento dos sujeitos não fazia grande diferença em obter mais respostas corretas. Embora, como se esperava, os sujeitos que eram especialistas ou bons em coesão ou acoplamento eram um pouco melhores que os sujeitos com pouco conhecimento.

Portanto, embora os resultados das experiências não tenham apresentado diferença estatisticamente significante, encontramos uma grande diferença entre coesão e acoplamento. Desta forma, os participantes se saíram melhor em coesão em vez de acoplamento.

CONSIDERAÇÕES FINAIS

O resultado desta pesquisa aumentou o volume de dados empíricos sobre o tema anomalias de código. Portanto, foi possível a ampliação do grau de compreensão sobre a viabilidade em adotar os conceitos de anomalias de código em processos de desenvolvimento de software. Essa compreensão permitiu-nos redirecionar pesquisas na área que desconsideram aspectos práticos do desenvolvimento de software, bem como a produção de ferramentas de suporte.

Portanto, pode-se afirmar que os resultados alcançados contribuem para a confirmação de que a hipótese testada de pôr em prática o princípio dos paradigmas de programação Orientada a Objetos (OO) não é uma tarefa trivial, pois requer conhecimento sólido dos princípios envolvidos e de experiência na área. Vale salientar que este tema ainda é pouco explorado em Engenharia de Software, o que pode direcionar estudos para caminhos inadequados, caso não se tenha um bom volume de estudos sobre o assunto.

REFERÊNCIAS

AHMED, I. et al. An empirical study of design degradation: How software projects get worse over time. In: 2015 ACM/IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM). [S.l.: s.n.], 2015. p. 1–10. ISSN 1949-3770.

BASILI, V. R. The experimental paradigm in software engineering. In: Proceedings of the International Workshop on Experimental Software Engineering Issues: Critical Assessment and Future Directions. [S.l.: s.n.], 1993. p. 3–12.

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

FU, S.; SHEN, B. Code bad smell detection through evolutionary data mining. In: 2015 ACM/IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM). [S.l.: s.n.], 2015. p. 1–9. ISSN 1949-3770.

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

LINARES-VáSQUEZ, M. et al. Domain matters: Bringing further evidence of the relationships among anti-patterns, application domains, and quality-related metrics in java mobile apps. In: Proceedings of the 22Nd International Conference on Program Comprehension. New York, NY, USA: ACM, 2014. (ICPC 2014), p. 232–243. ISBN 978-1-4503-2879-1.

MäNTYLä, M. V.; VANHANEN, J.; LASSENIUS, C. Bad smells humans as code critics. In: Proc. of the 20th IEEE International Conference on Software Maintenance (ICSM). [S.l.: s.n.], 2004. p. 399–408.

PALOMBA, F. et al. Do they really smell bad? a study on developers' perception of bad code smells. In: Proc. of the 30th IEEE International Conference on Software Maintenance and Evolution (ICSME). [S.l.: s.n.], 2014.

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.