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

AVALIAÇÃO DA APLICAÇÃO DOS CONCEITOS DE COESÃO E ACOPLAMENTO NA PROGRAMAÇÃO ORIENTADA A OBJETOS: UM EXPERIMENTO CONTROLADO

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

1. Bolsista PIBIC/CNPq, 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; Acoplamento.

INTRODUÇÃO

O paradigma de Orientação a Objetos (OO) surgiu como uma possível solução para desenvolver sistemas de qualidade com maior rapidez e economia. Os desafios de projetar usando a programação orientada a objetos é conhecido e discutido por várias abordagens. Um deles que é bastante comum no meio acadêmico é o livro de padrões de projeto (GAMMA et al., 2000). Os padrões de projeto são usados para propor soluções que são compartilhadas a fim de solucionar um problema comumente ocorrido no desenvolvimento de um software, de forma que possa ser reusado por outros desenvolvedores. Alguns desses problemas comuns têm implicações e soluções específicas como o padrão de projeto acoplamento e coesão, ambos usados no desenvolvimento orientado a objetos. Segundo McConnell (1993), Coesão se refere a quão "estritamente as operações em uma rotina estão relacionadas". Já Acoplamento se refere a "força de uma conexão entre duas rotinas". É um complemento a coesão. E descreve quão intensamente ela própria está relacionada com outras.

Uma questão pouco estudada está relacionada à compreensão da dificuldade em aplicar os conceitos de coesão e acoplamento corretamente, mesmo havendo referências realizando claramente este mapeamento, como o livro de Larman (2000) - Utilizando UML e Padrões: uma introdução à análise e ao projeto orientado a objetos. Larman (2000) apresenta formas de como usar os padrões de projetos na programação orientada a objetos e apresenta como resolver os problemas conhecido pelos termos de baixa coesão e forte acoplamento. Considerando o uso dessas referências como oráculos, experimentos controlados são um método apropriado para construir evidências sobre o tema. No entanto, a maioria dos experimentos controlados está relacionada à identificação de anomalias, não estudam formas de aplicação dos conceitos para evitar o surgimento das anomalias (SJØBERG et al., 2013; ZHANG; HALL; BADDOO, 2011; SCHUMACHER et al., 2010; MäNTYLä; VANHANEN; LASSENIUS, 2004).

MATERIAL E METODOLOGIA

O experimento foi desenvolvido por meio de uma sequência de atividades de acordo com o cronograma previsto no projeto. Foram envolvidos 17 participantes da Universidade Estadual de Feira de Santana (UEFS), no Brasil: Oito estudantes do mestrado em Ciência da Computação e nove alunos da graduação em Engenharia da

Computação. A participação no experimento foi voluntária. Devido a circunstâncias especiais, todos os participantes são (ou foram) profissionais na área. Perguntamos se eles tinham experiência profissional em desenvolvimento de software (considerando a indústria de software ou acadêmico).
Os participantes deveriam construir um PDV, o qual é um sistema usado para registrar vendas e administrar pagamentos, normalmente usados em lojas de varejo. Este programa deveria utilizar a linguagem Java de programação e atender às especificações apresentadas nos casos de usos propostos. Eles foram encorajados a adotar uma estratégia pessoal para realizar a implementação de cada caso de uso. A partir desta tarefa, comparamos se as escolhas dos participantes estão de acordo com a proposta descritas no livro de Larman. Para analisar os resultados, coletamos o sistema PDV de cada participante depois que eles programaram todos os casos de uso. Cada participante foi pontuado levando em consideração como os métodos, atributos e chamadas foram usados segundo a solução proposta pelo livro de Larman, onde foi usado como um oráculo para orientação e correção dos programas feitos pelos participantes. Para apoiar uma avaliação mais profunda relacionada à aplicação dos conceitos de acoplamento e de coesão, classificamos as questões de acordo com o conceito que eles estão mais ligados. Seis questões ligadas ao acoplamento e seis questões ligadas aos conceitos de coesão.

RESULTADOS E DISCUSSÃO

As principais descobertas de nossa análise foram feitas a partir de discursões sobre duas questões: “Como as características dos desenvolvedores impactam na aplicação dos conceitos de coesão e acoplamento?” E se “Existe uma diferença relevante no nível de correção considerando as questões de acoplamento e coesão de forma independente?”
A descoberta relacionada à primeira pergunta “Como as características dos desenvolvedores impactam na aplicação dos conceitos de coesão e acoplamento?” E se o conhecimento prévio sobre os conceitos de coesão e acoplamento podem impactar na atividade de programar, em termos de qualidade de projeto. Pelo menos a variável “conhecimento prévio” merece uma investigação mais aprofundada. Baseamos essa descoberta no resultado observado na Figura 1. Eles indicam que os programas dos participantes que têm conhecimento prévio sobre os conceitos de acoplamento e coesão tem mais classes coesas e menos acopladas, segundo o oráculo. Essas evidências permitem conjecturar sobre a relevância do treinamento de desenvolvedores, incluindo a educação formal. Também é possível que os participantes que declaram não ter conhecimento prévio sobre os conceitos de acoplamento e coesão podem ter sido introduzidos a eles sem serem assimilados apropriadamente. Muitos problemas podem ser considerados vindos desta conjectura. Por exemplo, a relevância de estudos focados em avaliação cognitiva ou na educação formal de desenvolvedores. Outra descoberta está relacionada à variável experiência do desenvolvedor: anos de experiência no desenvolvimento de software não afetam na qualidade do projeto durante a atividade de programação. De alguma forma, isso reforça a relevância do treinamento de desenvolvedores. Alguns podem especular que os desenvolvedores não programam software melhor com o tempo, em termos de qualidade de projeto. É importante destacar que a descoberta é apresentada no contexto do experimento, e em relação a construir classes mais coesas e menos acopladas. Isto não é uma negação ao valor da experiência no desenvolvimento de software.

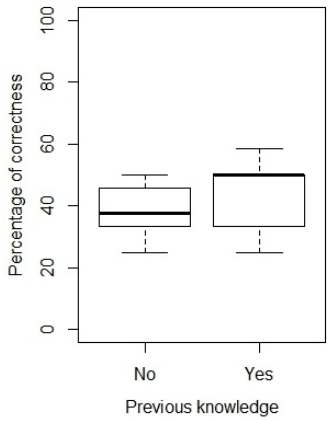

Figura 1: Distribuição de acertos (em porcentagem) agrupada por conhecimento prévio

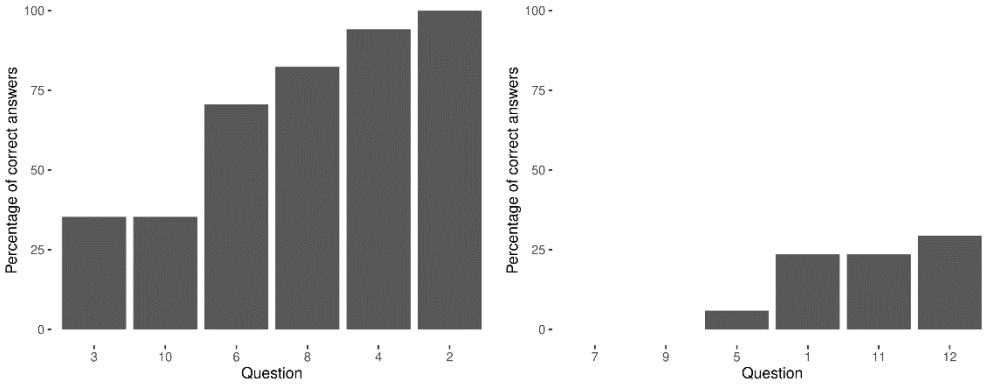

Figura 2: Porcentagem de acertos para questões de coesão (lado esquerdo) comparado com questões de acoplamento (lado direito)

A principal descoberta da segunda questão "Existe uma diferença relevante no nível de correção considerando as questões de acoplamento e coesão de forma independente?" É que a aplicação do conceito de acoplamento é mais difícil do que a aplicação do conceito de coesão para uma melhor qualidade de projeto em sistemas OO. Isso é evidente pela maior porcentagem de programas alinhados com o oráculo para questões relacionadas à coesão do que para questões relacionadas ao conceito de acoplamento. Esta descoberta é reforçada pela observação dos programas dos participantes agrupados por variáveis de conhecimento prévio. A Figura 2 mostra que há mais participantes aplicando o conceito de coesão apropriadamente do que aplicando o conceito de acoplamento. Especulamos que a aplicação do conceito de coesão está relacionada a uma (ou poucas) classe (s). Por outro lado, a aplicação do conceito de acoplamento depende de pensar sobre outras classes. Para aprofundar esta análise, consideramos algumas questões e suas soluções.

As principais descobertas do experimento são que o conhecimento prévio sobre os conceitos de acoplamento e coesão, parecem ser mais relevantes do que a experiência do desenvolvedor para uma melhor qualidade do projeto; A aplicação da coesão é significativamente mais fácil do que a aplicação do conceito de acoplamento. Essas descobertas permitem conjecturas sobre a relevância dos estudos focados no treinamento de desenvolvedores (incluindo aspectos formais) e cognitivos na compreensão e programação de princípios básicos de OO. Apesar das dificuldades em abordar aspectos cognitivos que normalmente são desconsiderados na área de engenharia de software, este experimento indica que mais esforço nessa área é importante.

CONSIDERAÇÕES FINAIS

De acordo com o projeto definido no plano de trabalho, executamos as etapas para a pesquisa do conteúdo, informações de como proceder para fazer um experimento, preparação de questionários para conhecer os participantes focando em suas experiências e conhecimento no assunto, a procura de um oraculo que seria como um guia para a correção do código gerado pelos participantes, execução do experimento, correção e avaliação do desempenho e a analise baseada em suas experiências e conhecimento. Para a realização da análise foi feito replicação do experimento a fim de aumentar os dados colhidos do primeiro experimento. E os resultados apontaram para a confirmação da hipótese testada que é tentar pôr em prática os princípios dos paradigmas de programação Orientada a Objetos (OO) a qual não é uma tarefa trivial, pois requer conhecimento sólido dos princípios envolvidos na experiência. Tal tema é pouco explorado em Engenharia de Software, o que pode direcionar estudos para caminhos inadequados.

REFERÊNCIAS

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.

C. Larman. 2002. Applying UML and Patterns: An Introduction to Object-Oriented Analysis and Design and the Unified Process. Prentice-Hall, Inc., NJ, United States

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

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.

MCCONNELL, S. Code Complete: A practical Handbook of Software Construction, Redmond: Microsoft Press, 1993, p. 81, 87.

R. Johnson E. Gamma, R. Helm and J. Vlissides. 1994. Design Patterns: Elements of Reusable Object-Oriented Software. Addison-Wesley Professional, U.S.

SCHUMACHER, J. et al. Building empirical support f
or automated code smell detection. In: Proc. of the 4th International Symposium on Empirical Software Engineering and
Measurement (ESEM). [S.l.: s.n.], 2010. p. 1–10.

SHALLOWAY, ALAN.; TROTT, JAMES R. Design Patterns Explained: A New Perspective on Object-Oriented Design 1st. ed. Addison-Wesley Professional, 2002.

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.

SOMMERVILLE, IAN. Software Engineering 9 . ed. Addison-Wesley Boston Columbus,2011.

WOHLIN, C. et al. Experimentation in Software Engineering. [S.l.]: Springer Berlin Heidelberg, 2012.

ZHANG, M.; HALL, T.; BADDOO, N. Code bad smells: A review of current knowledge. Software Maintenance and Evolution: Research and Practice, John Wiley & Sons, Inc., New York, NY, USA, v. 23, n. 3, p. 179–202, abr. 2011. ISSN 1532-060X.