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

O PAPEL DOS DESENVOLVEDORES NOS CODE SMELLS EM PROJETOS OPEN SOURCE

Bernardo Oliveira Rosa[1]; José Amancio Macedo Santos[2]

1. Bolsista PIBIC/CNPq, Graduando em Engenharia de Computação Universidade Estadual de Feira de Santana, e-mail: bernardoorosa@gmail.com

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

PALAVRAS-CHAVE: Code Smells; Repositórios de Código Aberto; Engenharia de Software Experimental.

1. INTRODUÇÃO

Este trabalho visa contribuir com evidências experimentais para a discussão sobre *Code Smell* (FOWLER, 1999) e a utilidade real de sua identificação e estudo para a indústria de software. *Code smells,* ou, numa tradução literal, "mal cheiros de código", é um conceito utilizado na Programação Orientada a Objetos para determinar se um projeto pode vir a ter problemas no sua evolução.

Diferentes autores propõem diferentes heurísticas para a identificação de *Code Smells,* algumas mais relacionada a observação direta, outras à detecção automática a partir de métricas extraídas do projeto. Muitas ferramentas têm sido produzidas no intuito de detectar automaticamente (CHEN; JIANG, 2017) *Code Smells* e utilizá-los na indústria de software para evitar o acúmulo de dívida técnica. É questionável, no entanto, se tais esforços de fato contribuem para o sucesso dos projetos de desenvolvimento de software (SJØBERG, et al., 2013).

Desta forma, este trabalho visa investigar a correlação entre desenvolvedores e *Code Smells* em projetos de código aberto publicados em plataformas como o GitHub[1]. Seguindo a linha de estudos similares sobre o tema, pretende-se contribuir para a discussão através de experimentos (BASILI, 1993) de coleta de dados em repositórios de código e correlacionando com os *Code Smells* encontrados nos projetos.

[1]https://github.com

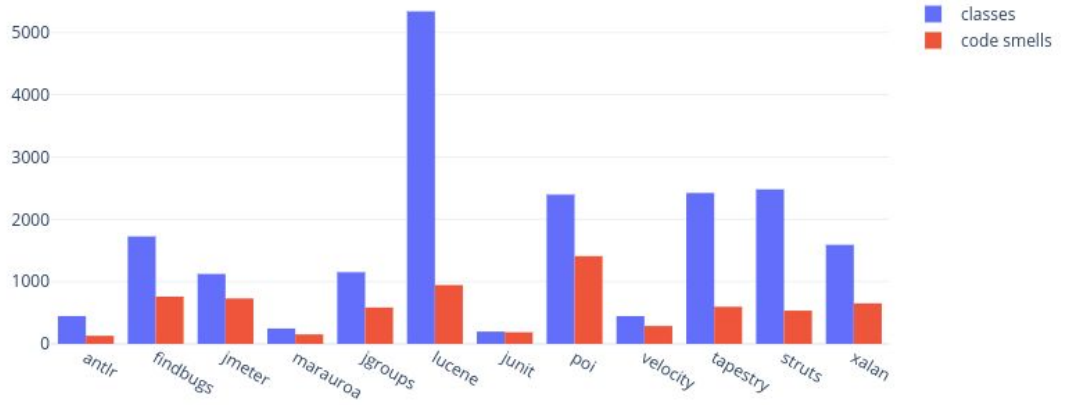
2. MATERIAL E MÉTODOS

A seguir indicaremos os métodos pelos quais obtivemos os dados para a análise, ou seja, a realização prática do experimento.

2.1. SISTEMAS ESCOLHIDOS

A escolha de sistemas para análise dependeu principalmente da disponibilidade de sistemas e dados confiáveis sobre as ações de desenvolvedores em tais sistemas.

As métricas sobre code smells, sobre quais classes do sistema faziam parte de code smells, foi principalmente extraídas do artigo "'*Code Smells* and their collocations: A large-scale experiment on open-source systems" que investiga a detecção de smells, sua classificação e a relação entre diferentes smells num projeto. O estudo analisou 92 sistemas, usando 9 diferentes ferramentas de detecção automáticas. O resultado do procedimento, publicado no *Elsevier's Journal of Systems & Software,* foi disponibilizado pelos autores de maneira organizada e bem documentada.
Utilizaram o *QC* (Qualita Corpus[1]), uma coleção de diferentes sistemas escritos em Java (linguagem de programação orientada a objetos) disponibilizados pelos pesquisadores e amplamente utilizados para trabalhos no campo da engenharia de software empírica (TEMPERO, et al., 2010). A versão utilizada foi a "Release 20130901", que representa a data em que esta versão foi lançada, e indica a versão e data de lançamento de cada sistema utilizado. O QC disponibilizou 112, o estudo investiga 92 deles. Entre esses 92 sistemas alguns encontramos no github com a mesma versão disponível no github. Usamos essa abordagem porque ela nos permitiu obter uma quantidade razoável de sistemas, 14 no total, com um conjunto de dados de code smells já disponível. A quantidade de sistemas foi drasticamente reduzida a partir do conjunto de dados smells porque os sistemas QC são mais antigos e não hospedados em plataformas modernas de compartilhamento de repositórios como o Github, por isso não nos permite extrair facilmente informações do desenvolvedor.
[1]http://java.labsoft.dcc.ufmg.br/qualitas.class/index.ht



2.2. CONJUNTO DE DADOS DO DESENVOLVEDOR:
Para coletar dados sobre qual desenvolvedor fez uma mudança em cada classe dentro de um período de tempo, a primeira versão e a data de lançamento usada no *QC* usamos os repositórios de código aberto hospedados no github. Procuramos a versão específica utilizada na detecção de *Code Smells*: para confirmar que a versão que estávamos recebendo do github era a mesma usada no conjunto de dados, fomos ao catálogo *QC* para a versão 20130901 e comparamos cada número de versão com projetos com o mesmo nome no Github: encontramos 14 sistemas disponíveis que atendem a esse critério em repositórios abertos no Github.

2.3. *CRAWLING* GITHUB:

Para obter as informações sobre um desenvolvedor fazendo alterações em alguma classe, rastreamos a página de repositórios na versão principal e coletamos as informações de alteração feita pela desenvolvedor. Depois de rastrear a ramificação principal dos repositórios, foi possível obter o *HTML* da página, que usamos para extrair informações sobre cada confirmação em cada repositório, ramificando os desenvolvedores responsáveis e os arquivos que eles alteraram.
Usamos a própria página *HTML* para pegar os dados usando a biblioteca cheerio para *NodeJS*[1], navegamos a página através de sua própria estrutura *HTML*. Usamos o nome e os nomes de classe da unidade *HTML* para coletar os dados que desejávamos: o nome de usuário e os arquivos alterados pelo desenvolvedor.
[1]https://nodejs.org/

RESULTADOS E/OU DISCUSSÃO (ou Análise e discussão dos resultados)
A próxima pergunta a ser feita é: quem é responsável por fazer cada um desses *Code Smells*? Como podemos vincular as classes de *Code Smell* ao desenvolvedor responsável? Bem, cada *Code Smell* é associado a uma classe ou a um método em uma classe ou ao conjunto de diferentes classes juntas. Em qualquer caso, é possível apontar para uma classe e dizer que ela tem ou é parte de um *Code Smell*.
Apontar o dedo para um determinado desenvolvedor e dizer que ele é responsavelmente por fazer com que a classe tenha um *Code Smell* é algo que depende dele fazer pelo menos uma modificação no código da classe dentro do período de tempo a partir da primeira versão até a versão na qual os *Code Smells* foram detectados. Porque um desenvolvedor mexendo com uma classe com *Code Smells* geralmente significa que ele ou ela fez o *Code Smell* ou deixou a classe suscetível a isso, ou não corrigiu um *Code Smell pré-existente*. Além disso, o número de desenvolvedores envolvidos em cada classe individualmente tende a ser pequeno.

2.1. Cruzando os dados
Podemos extrair de registros de alteração em código os arquivos (e, portanto, as classes) alterados pelo desenvolvedor naquele registro. Extraindo todos os dados de um registro de alteração de um determinado projeto, podemos ter uma idéia do que um certo desenvolvedor fez em um determinado projeto. Os dados sobre *Code Smells* nos oferecem a outra peça que precisamos para analisar o impacto de um determinado desenvolvedor na saúde de um projeto. Uma lista de quais classes contêm *Code Smells* nos projetos analisados funcionará, pois pode ser relacionada à tabela um para nos mostrar quais classes têm cheiros e quais não têm.
O projeto e a classe precisam corresponder entre as duas listas para determinar se é a mesma classe. As classes que não aparecem na lista de classes com *Code Smells* são consideradas sem *Code Smells*, então as que aparecem são com *Code Smells*.

Para comparar os desenvolvedores, contamos quantas classes eles fizeram alterações e quantas dessas continham *Code Smells*, sendo capazes de calcular a porcentagem de classes que continham *Code Smells* do total de classes que eles alteraram. Isso nos permitiu colocar os desenvolvedores em dois grupos, dependendo de quantos *Code Smells* eles criaram, desenvolvedores bons e ruins. Adotamos dois limiares nesta

análise. Desenvolvedores ruins com mais de 70% de *Code Smells (Imagem 3)* e muito ruins com mais de 80% (Imagem 4) das classes com *Code Smells*.

Nesse ponto, é possível pensar em muitas coisas que podem justificar um desenvolvedor ser ruim, como estar em um projeto que já era muito ruim. Então para testar essa hipótese precisávamos testar uma coisa nova, precisávamos saber a porcentagem de classes fedorentas que um determinado projeto tem, então responder a pergunta "é esse desenvolvedor realmente ruim ou o fato de que eles foram considerados ruins é apenas a conseqüência de estar em um projeto que já tem muitos cheiros? ". Assim, calculamos a porcentagem de *Code Smells* em um projeto, a partir do gráfico da Figura 1, classificando-os como bons ou ruins, analisamos quantos desenvolvedores bons ou ruins estavam em projetos bons ou ruins.

CONSIDERAÇÕES FINAIS (ou Conclusão)

As inconsistências nos dados obtidos tornaram a análise difícil, havia alterações com atribuição a desenvolvedores não identificáveis (sem usuário no Github), o próximo passo na análise, que não foi feito por conta do tempo, seria analisar as causas desses dados específicos não terem e seu impacto na análise como um todo. A unidade de código a qual um *Code Smell* se refere: métodos, classes ou conjuntos de classes, também é um elemento que pode ser debatido, se é possível atribuir um *Code Smell* diretamente a um desenvolvedor que alterou uma classe afetada por ele. Mais dados sobre os projetos e desenvolvedores devem ser coletados para compor a análise, como tempo de cada desenvolvedor no projeto e número de desenvolvedores no projeto. Dessa forma o estudo ainda passar por melhoria na análise e coleta de dados antes de se tornar uma real contribuição para a discussão da comunidade científica acerca do tema, trazendo todos os dados possíveis que possam de alguma forma afetar a relação entre desenvolvedores e *Code Smells*.

REFERÊNCIAS

FOWLER, Martin. Chapter 3: Bad Smells in Code. *In*: FOWLER, Martin; BECK, Kent. **Refactoring: Improving the Design of Existing Code**. USA: Addison-Wesley Professional, 1999.

TEMPERO, Ewan; AL., Et. The Qualitas Corpus: A Curated Collection of Java Code for Empirical Studies. **2010 Asia Pacific Software Engineering Conference**, Sydney, NSW, Australia, 2010. Disponível em: https://ieeexplore.ieee.org/abstract/document/5693210. Acesso em: 12 ago. 2019.

OLSTON, Christopher; NAJORK, Marc. Web Crawling. **Foundations and Trends® in Information Retrieval: Vol. 4: No. 3, pp 175-246**, USA, 2010. Disponível em: https://www.nowpublishers.com/article/Details/INR-017. Acesso em: 12 ago. 2019.

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.

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.

WALTER, Bartosz; FONTANAB, Francesca; FERMEC, Vincenzo. Code smells and their collocations: A large-scale experiment on open-source systems. **Journal of Systems and Software, vol 44, p1-21**, [*S. l.*], 2018.