Universidade Federal de Juiz de Fora

Instituto de Ciências Exatas

Programa de Pós-graduação em Ciência da Computação

Vinicius Junqueira Schettino

Uma ferramenta para recomendação de revisores de código para apoiar a colaboração em Desenvolvimento Distribuído de Software

Juiz de Fora

2018

Vinicius Junqueira Schettino

Uma ferramenta para recomendação de revisores de código para apoiar a colaboração em Desenvolvimento Distribuído de Software

Dissertação apresentada ao Programa de Pós-graduação em Ciência da Computação da Universidade Federal de Juiz de Fora, como requisito parcial para obtenção do título de Mestre em Ciência da Computação.

Orientador: Marco Antônio Pereira Araújo

Juiz de Fora

2018

Ficha catalográfica elaborada através do Modelo Latex do CDC da UFJF
com os dados fornecidos pelo(a) autor(a)

Junqueira Schettino, Vinicius.
Uma ferramenta para recomendação de revisores de código para apoiar a colaboração em Desenvolvimento Distribuído de Software / Vinicius Junqueira Schettino. – 2018.
21 f.

Orientador: Marco Antônio Pereira Araújo
Dissertação (Mestrado) – Universidade Federal de Juiz de Fora, Instituto de Ciências Exatas. Programa de Pós-graduação em Ciência da Computação, 2018.

1. Palavra-chave. 2. Palavra-chave. 3. Palavra-chave. I. Pereira Araújo, Marco Antônio, orient. II. Título.

Vinicius Junqueira Schettino

Uma ferramenta para recomendação de revisores de código para apoiar a colaboração em Desenvolvimento Distribuído de Software

Dissertação apresentada ao Programa de Pós-graduação em Ciência da Computação da Universidade Federal de Juiz de Fora, como requisito parcial para obtenção do título de Mestre em Ciência da Computação.

Aprovada em:

BANCA EXAMINADORA

Prof. Dr. Marco Antônio Pereira Araújo - Orientador
Universidade Federal de Juiz de Fora

Professor Dr. ??
Universidade ???

Professor Dr. ??
Universidade ??

AGRADECIMENTOS

De acordo com a Associação Brasileira de Normas Técnicas - 14724 (2011, p. 1) Agradecimentos é o “texto em que o autor faz agradecimentos dirigidos àqueles que contribuíram de maneira relevante à elaboração do trabalho.”

“Texto em que o autor apresenta uma citação, seguida de autoria, relacionada com a matéria tratada no corpo do trabalho”
(ASSOCIAÇÃO BRASILEIRA DE NORMAS TÉCNICAS, 2011, p. 2)
A epígrafe elaborada conforme NBR 10520 (Epígrafe - Opcional)

RESUMO

De acordo com a Associação Brasileira de Normas Técnicas - 6028 (2003, p. 2) “o resumo deve ressaltar o objetivo, método e as conclusões do documento (...) Deve ser composto de uma sequência de frases concisas, afirmativas e não de enumeração de tópicos. Recomenda-se o uso de parágrafo único.” O resumo deve ter de 150 a 500 palavras.

Palavras-chave: Palavra-chave. Palavra-chave. Palavra-chave.

ABSTRACT

...

Key-words: ...

LISTA DE ILUSTRAÇÕES

LISTA DE ABREVIATURAS E SIGLAS

UFJF	Universidade Federal de Juiz de Fora
DDS	Desenvolvimento Distribuído de Software

SUMÁRIO

1 INTRODUÇÃO

O *code review* é considerado como uma das principais técnicas de diminuição de defeitos de software [1]. Nela, o autor de uma alteração na base de código de um projeto submete tal conteúdo ao crivo de um conjunto de pares técnicos, que irão revisar sua estrutura com base em um lista de regras e convenções previamente definida. Diferentes aspectos relacionados ao autor, ao revisor e ao processo de revisão em si estão diretamente relacionados à eficiência da prática. Autores relatam relação da diminuição da incidência de *anti-patterns* [2] de acordo com o nível de participação dos envolvidos e cobertura do código revisado [3, 4, 5]. Reputação [6] e experiência [7] do revisor também parecem impactar nos efeitos do *code review*

Intrinsecamente colaborativa, a atividade hoje é exercida com suporte de ferramentas computacionais específicas [8], principalmente no desenvolvimento distribuído. Dentro de workflows de trabalho descentralizados [9], a prática funciona como um *gateway* de qualidade que busca garantir que apenas alterações aderentes aos padrões de qualidade do projeto serão incorporados à codebase principal. Esta etapa do desenvolvimento se torna uma oportunidade para disseminação de conhecimento, embate de ideias e discussão de melhores práticas entre profissionais de experiência e visões diferentes.

Tais aspectos configuram o Desenvolvimento Distribuído de Software (DDS), onde equipes de desenvolvimento se encontram espalhadas por organizações e espaços geográficos distintos. Este novo ramo da Engenharia de Software vem modificando a relação entre empresas e sistemas, principalmente em relação às estratégias de negócios [10]. As próprias relações de negócios fomentam a distribuição das equipes, procurando diminuição dos custos e a incorporação de mão de obra qualificada que pode estar em qualquer lugar do planeta.

Neste contexto, porém, os os desafios à colaboração co-localizada são potencializados e as soluções tradicionais não são suficientes para fomentar esta aspecto das atividades distribuídas [11]. Casey [12] mostra que, com a distribuição geográfica dos times, diversos outros desafios, antes considerados colaterais ou resolvidos, emergem de forma a ameaçar a colaboração entre os membros da equipe: barreiras culturais, temporais e geográficas; reengenharia dos processos de desenvolvimento; resistência em compartilhar informações e conhecimento com os pares distribuídos; entre outros desafios.

Estes desafios do Desenvolvimento Distribuído de Software (DDS) afetam o *code review* de duas formas distintas. Primeiro, o processo de revisão pode se tornar menos eficiente quando a colaboração é afetada, devido aos baixos níveis de participação e cobertura. O mesmo vale para a disseminação do conhecimento, que fica prejudicada. Outro desafio que se forma é a escolha do revisor adequado para aquele *patch*. Com um vasto número de opções e pouca informação disponível sobre seus aspectos técnicos

e gerenciais (e.g. tempo disponível) já que não há contato co-localizado entre eles, o a natureza distribuída deste tipo de desenvolvimento dificulta o processo de escolha do revisor, o que também pode impactar a eficiência do revisor.

Uma possível solução, visando amparar o desafio da colaboração e evitando o *overhead* da escolha do revisor, seria manter grupos bem testados e experientes exercendo as atividades de revisão. Ou ainda, fixar, dentro de cada equipe de desenvolvimento, quem são os responsáveis por revisão e pela submissão dos *patches*, evitando a diversificação das relações de trabalho.

Contudo, estudos recentes demonstram que a fixação de grupos e responsabilidades pode não ser benéfica para o processo de desenvolvimento. Scott Page [13] argumenta que a diversidade de experiências, visões e especilidades fazem com que grupos sejam mais eficientes. Já Prikladnicki et al. [14] apontam índicios de que a formação de grupos temporários em detrimento ou em conjunto com permanentes é um fator de eficiência em projeots de software:

> "Although old colleagues bring knowledge of the development process and prior norms from previous teams, new members bring fresh ideas that could promote project performance and creativity. Old colleagues might not do so and might not give new members a chance to implement their ideas."

Essa visão aponta que a formação dinâmica dos grupos de trabalho em desenvolvimento de software potencializa a disseminação do conhecimento, um dos objetivos primários do *code review* [8].

Existem alguns trabalhos congêneres que demonstram métodos de recomendação de revisores. Esses trabalhos foram estudados e levados em consideração para escrita do presente texto. Além disso, foram revisadas pesquisas que apontam caracterísitcas de revisões, revisores e autores que possivelmente potencializam a colaboração. Tais aspectos são apresentados e discutidos no capítulo 3.

As principais lacunas deixadas pelos trabalhos anteriores estão relacionadas aos objetivos e à avaliação dos métodos propostos, principalmente em DDS. Primeiramente, não há relato de método de recomendação de revisores de código com o objetivo específico de potencializar a colaboração. Por isso, métodos já propostos não utilizam métricas nem variáveis de entrada relacionadas aos aspectos de cooperação, coordenação e comunicação, como por exemplo a abordagem 3C em DDS [15].

A segunda lacuna é observável na avaliação dos modelos de avaliação. Os trabalhos encontrados se limitam a comparar seus resultados com métricas relacionadas à proximidade dos mesmos com a indicação manual do revisor. Ou seja, a eficência é tida de acordo com a interseção entre o recomendado automaticamente e por decisão de um especialista, geralmente um desenvolvedor. Este modelo assume que o responsável pela indicação

manual tem os subsídios naturais para fazer uma boa escolha. Em DDS isso pode não ser verdade, uma vez que fatores como diferenças culturais, de horário, geográficas e de maturidade podem diminuir a compreensão do indicador e propiciar a escolha inadequada do revisor. Por isso, no contexto apresentado, outras formas de avaliação podem ser mais apropriadas. Tais discussão são extendidas no capítulo 2.

Expostas os desafios que o Desenvolvimento Distribuído de Software impõe sobre a escolha do revisor de código, a importância da escolha do revisor adequado do ponto de vista de colaboração e a motivação da formação de grupos heterogêneos e dinâmicos, sumariza-se o objetivo deste trabalho. De acordo com a abordagem QGM (Goal/Question/Metric) proposta por Basili et al. [16], postula-se o objetivo do trabalho como: **Implementar** um método de recomendação de revisores **com o objetivo de** potencializar a colaboração **em relação aos aspectos** de coordenação **do ponto de vista** de revisores e autores **no contexto de** desenvolvimento distribuído de software.

A principal hipótese que norteia o andamento desta proposta, e que será revisitada e discutida nos capítulos derradeiros é:

- O método de recomendação apresentado pode potencializar a colaboração entre revisores e autores.

O uso de ferramentas computacionais para o processo de revisão de código se tornou prática comum nos últimos anos [8]. O GitHub, possivelmente a maior plataforma de código do planeta, é uma plataforma rica em repositórios de projetos de software. Muitos são de código aberto, disponíveis para mineração. Discussões sobre o *workflow* de trabalho na ferramenta em contraponto à métodos tradicionais de revisão podem ser vistas na seção 2.1. São 24 milhões de usuários, 67 milhões de projetos e 47 milhões de revisões[1], também chamadas de *pull requests* no modelo de desenvolvimento "*pull based*" [17]. Essa abordagem é explorada em pormenores na seção 2.1.3.

Esta característica permitiu a extração e análise automatizadas das informações sobre as revisões em projetos de código aberto, através de APIs disponibilizadas para este fim. Foram as extraídas métricas apontadas como relevantes para nossos objetivos pela literatura relacionada. A arquitetura que embasa a extração e análise destes dados com objetivo de recomendação é explicada no capítulo 4.

A avaliação da eficiência do método proposto apresenta particularidades em relação à trabalhos relacionados, devido ao enfoque em colaboração no contexto de DDS. O método de avaliação é devidamente discutido e aplicado no capítulo 5, incluindo a apresentação dos experimentos e a revisitação da hipótese levantada neste capítulo. Por fim, o capítulo 6

[1] https://octoverse.github.com/

é dedicado ao fechamento do trabalho, assim como à sugestação de trabalhos futuros e à discussão de ameaças a validade e generalização dos resultados apresentados.

2 PRESSUPOSTOS TEÓRICOS

2.1 *Code review*

(justificar o porquê da escolha do GitHub)

2.1.1 Relevância

2.1.2 Histórico

2.1.3 Pull Based Method

(literalmente mostrar como funciona)

2.2 Modelo 3c

2.3 Desenvolvimento Distribuído de Software

3 REFERENCIAL TEÓRICO

3.1 Revisão sistemática da literatura

3.2 Métricas relevantes para recomendação do revisor

3.3 Métricas de avaliação dos resultados

3.4 Outros trabalhos relevantes

4 SOLUÇÃO DESENVOLVIDA

(explicações técnicas, MER, tecnologias, etc)

5 AVALIAÇÃO DA SOLUÇÃO

5.1 Processo de Avaliação

5.2 Apresentação dos resultados

5.3 Discussão dos resultados

6 CONCLUSÃO

6.1 Ameaças

6.2 Trabalhos futuros

6.3 Considerações finais

REFERÊNCIAS

[1] BOEHM, B.; BASILI, V. R. Software defect reduction top 10 list. *Computer*, IEEE Computer Society Press, Los Alamitos, CA, USA, v. 34, n. 1, p. 135–137, 12 2001. ISSN 0018-9162.

[2] KEMERER, C. F.; PAULK, M. C. The impact of design and code reviews on software quality: An empirical study based on psp data. *IEEE Transactions on Software Engineering*, v. 35, n. 4, p. 534–550, 07 2009. ISSN 0098-5589.

[3] MENEELY, A. et al. An empirical investigation of socio-technical code review metrics and security vulnerabilities. In: . [S.l.: s.n.], 2014. p. 37–44.

[4] MORALES, R.; MCINTOSH, S.; KHOMH, F. Do code review practices impact design quality? a case study of the qt, vtk, and itk projects. *2015 IEEE 22nd International Conference on Software Analysis, Evolution and Reengineering (SANER)*, IEEE Computer Society, Los Alamitos, CA, USA, v. 00, p. 171–180, 2015.

[5] BAVOTA, G.; RUSSO, B. Four eyes are better than two: On the impact of code reviews on software quality. In: . [S.l.: s.n.], 2015. p. 81–90.

[6] BAYSAL, O. et al. The influence of non-technical factors on code review. In: . [S.l.: s.n.], 2013. p. 122–131.

[7] KONONENKO, O. et al. Investigating code review quality: Do people and participation matter? In: . [S.l.: s.n.], 2015. p. 111–120.

[8] BACCHELLI, A.; BIRD, C. Expectations, outcomes, and challenges of modern code review. In: *Proceedings of the 2013 International Conference on Software Engineering*. [S.l.]: IEEE Press, 2013. (ICSE '13), p. 712–721. ISBN 978-1-4673-3076-3.

[9] GOUSIOS, G.; STOREY, M.-A.; BACCHELLI, A. Work practices and challenges in pull-based development: The contributor's perspective. In: IEEE. *Software Engineering (ICSE), 2016 IEEE/ACM 38th International Conference on*. [S.l.], 2016. p. 285–296.

[10] AUDY, J. L. N.; PRIKLADNICKI, R. *Desenvolvimento distribuído de software*. [S.l.]: Elsevier, 2007.

[11] COSTA, A. M. Nicolaci-da; PIMENTEL, M. Sistemas colaborativos para uma nova sociedade e um novo ser humano. *Sistemas colaborativos. PIMENTEL, M.; FUKS, H.(Orgs.). Rio de Janeiro: Elsevier*, 2011.

[12] CASEY, V. Virtual software team project management. *Journal of the Brazilian Computer Society*, Springer, v. 16, n. 2, p. 83–96, 2010.

[13] PAGE, S. E. *The difference: How the power of diversity creates better groups, firms, schools, and societies*. [S.l.]: Princeton University Press, 2008.

[14] PRIKLADNICKI, R. et al. The best software development teams might be temporary. *IEEE Software*, v. 34, n. 2, p. 22–25, Mar 2017. ISSN 0740-7459.

[15] FUKS, H. et al. Do modelo de colaboração 3c à engenharia de groupware. *Simpósio Brasileiro de Sistemas Multimídia e Web–Webmidia*, p. 0–8, 2003.

[16] BASILI, V. R.; WEISS, D. M. A methodology for collecting valid software engineering data. *IEEE Trans. Softw. Eng*, SE-10, no. 6, p. 728–738, 1984.

[17] GOUSIOS, G.; PINZGER, M.; DEURSEN, A. v. An exploratory study of the pull-based software development model. In: ACM. *Proceedings of the 36th International Conference on Software Engineering*. [S.l.], 2014. p. 345–355.

APÊNDICE A – Artigo Mapeamento Sistemático

Colocar aqui o artigo.