Análise da Formação e Evolução de Times de Desenvolvimento no Hibernate-ORM

Talita S. Orfanó, Michele A. Brandão, Larissa E. Maia, Mirella M. Moro

Universidade Federal de Minas Gerais - Belo Horizonte - MG

talitaorfano@ufmg.br, {micheleabrandao,larissaemanuelle,mirella}@dcc.ufmg.br

Abstract. *Various factors influence the formation and evolution of relationships in social networks. Here, we have developed a platform called AuthView to characterize the behavior of developers in Hibernate-ORM and how their relationships evolve. The results show that (1) developers tend to start contributing to a project from specific files; (2) small developers tend to share knowledge of a project module always with other major developers; and (3) there is no formation of small internal development teams.*

Resumo. *Diversos fatores influenciam a formação e evolução das relações em redes sociais. Aqui, desenvolvemos uma ferramenta chamada AuthView para caracterizar o comportamento de desenvolvedores no Hibernate-ORM e como suas relações evoluem. Os resultados mostram que (1) desenvolvedores tendem a iniciar contribuição no projeto a partir de arquivos pontuais; (2) pequenos desenvolvedores tendem a compartilhar o conhecimento de um módulo do projeto sempre com outros grandes desenvolvedores; e (3) não há a formação de pequenos times internos de desenvolvimento.*

1. Introdução

O termo *social coding* refere-se ao desenvolvimento de software baseado no compartilhamento de conhecimento entre desenvolvedores. Com a disseminação dessa prática e a popularização dos projetos *Open Source*, as redes sociais de colaboração, como o GitHub[1], representam produtividade, praticidade e aproximam desenvolvedores de diferentes localidades [Tsay et al. 2014]. O GitHub é considerado um dos maiores repositórios de projetos comerciais e pessoais para controle de versão e colaboração.

Nesse contexto, este trabalho propõe um estudo de um repositório do GitHub para compreender as características dos relacionamentos entre os desenvolvedores e a formação dos times. Tal objetivo é dividido em três questões de pesquisa: *(Q1)* Como os desenvolvedores começam a contribuir no projeto? *(Q2)* Como os membros compartilham entre si o conhecimento do sistema no decorrer do tempo? *(Q3)* Há a formação de times menores no projeto? Para responder a essas questões, os arquivos (.java) e *commits* dos desenvolvedores responsáveis pelo repositório do Hibernate-ORM [2] foram analisados. Buscou-se compreender a estrutura desses times a nível de classes, analisando desde o surgimento, a formação e evolução dos relacionamentos entre os desenvolvedores. Para realizar tais análises, desenvolvemos uma ferramenta chamada *AuthView* (*Auth*

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

[2]Hibernate-ORM: https://github.com/hibernate/hibernate-orm

de *Authorship* por considerar autoria de software feita pelos desenvolvedores e *View* por possibilitar a visualização da rede de colaboração). Tal ferramenta permite a análise das contribuições de cada desenvolvedor por meio da coleta dos arquivos de cada *commit* das versões examinadas do repositório.

Entender a formação e evolução de times de desenvolvimento é importante para detectar como as características dos desenvolvedores estão relacionadas aos softwares e seus respectivos componentes. Trabalhos recentes apontam uma significativa relação entre os fatores humanos e a qualidade do software: a falta de responsabilidade e domínio sobre o código podem reduzir sua qualidade [Greiler et al. 2015].

O trabalho está organizado da seguinte forma: a Seção 2 descreve os trabalhos relacionados. Em seguida, a Seção 3 detalha a metodologia e a Seção 4 apresenta a ferramenta *AuthView* e suas principais funcionalidades. Os resultados são mostrados na Seção 5. Finalmente, a Seção 6 apresenta as conclusões e os trabalhos futuros.

2. Trabalhos Relacionados

Características sociais revelam a complexidade dos relacionamentos entre desenvolvedores. Nesse contexto, Tsay et al. [2014] investigam características sociais que influenciam contribuições em projetos *open source* no GitHub. Além disso, Casalnuovo et al. [2015] estudam como conexões sociais anteriores afetam a produtividade e o surgimento de times. Dabbish et al. [2012] mostra que usuários utilizam informações no GitHub para inferir sobre as habilidades e o conhecimento de desenvolvedores, bem como decidir se os mesmos podem iniciar colaboração no projeto.

Ademais, compreender o quanto desenvolvedores conhecem de um software facilita para os gestores manter as pessoas responsáveis por tarefas com as quais possuem conhecimento e encontrar especialistas para tarefas específicas [Rahman & Devanbu 2011]. Além disso, analisar repositórios no GitHub revela a existência de perfis de comportamentos de desenvolvedores em diferentes linguagens de programação [Rocha et al. 2016].

Em uma abordagem próxima a adotada neste trabalho, Alves et al. [2016] e Batista et al. [2017] analisam a força dos relacionamentos no GitHub a nível de repositório, direcionando o estudo em repositórios desenvolvidos majoritariamente em JavaScript e Ruby. Tais estudos mostram que a maioria dos desenvolvedores são ativos em poucos repositórios. Isso revela a necessidade de uma pesquisa que analise níveis mais profundos de abstração em um determinado repositório.

Assim, este trabalho visa suprir a necessidade levantada por [Alves et al. 2016; Batista et al. 2017]. Este estudo inicia uma análise na formação de times de desenvolvimento, buscando compreender em uma granularidade de arquivos Java. Ademais, analisamos como os desenvolvedores interagem entre si, considerando suas contribuições no decorrer do tempo de vida do projeto.

3. Metodologia

Este artigo analisa a formação e evolução de times de desenvolvimento no GitHub por meio do estudo do Hibernate-ORM, que é um *framework open source* desenvolvido em Java. Escolhemos tal projeto devido a quantidade de desenvolvedores que contribuíram, a quantidade de arquivos fonte existentes que compõem o sistema, a estabilidade no ciclo

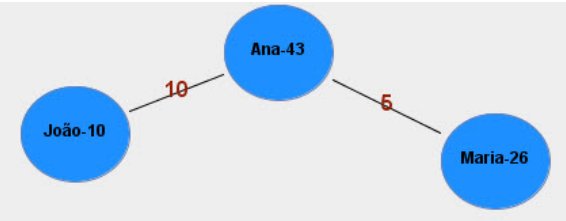

Figura 1: Representação da interação entre os desenvolvedores e suas contribuições.

evolutivo do projeto, a presença de diversas versões do projeto no repositório do GitHub e a sua relevância no cenário de tecnologia da informação.

Para compreender a distribuição dos membros de um time, oito versões (*tags*) do Hibernate-ORM foram analisadas, totalizando 2.195 *commits*, 29 desenvolvedores e mais de 8 mil arquivos. Investigou-se como os desenvolvedores interagem entre si em cada versão. A interação entre os membros acontece quando dois desenvolvedores alteram um mesmo arquivo em uma mesma versão. Por exemplo, dois desenvolvedores A e B que realizaram *commits* em cinco arquivos em comum possuem uma relação mais forte do que C e D que fizeram *commits* em apenas dois arquivos em comum em uma versão.

4. *AuthView* - uma Ferramenta para Analisar Arquivos de Commits

A ferramenta *AuthView* possui quatro funcionalidades principais: a coleta de dados, o refinamento e armazenamento dos dados, a visualização da distribuição dos desenvolvedores em cada versão e a análise comparativa entre versões do repositório. A ferramenta foi criada para permitir futuras análises. Por isso, ela possibilita a análise de qualquer repositório do GitHub, sem restrições de linguagem de programação ou tamanho.

Coleta de dados. Durante essa etapa são extraídas informações sobre todos os arquivos e os correspondentes desenvolvedores. Dessa forma, foram coletados dados de todos os *commits* de cada versão (*tag*) selecionada. Alguns critérios foram adotados e padronizados durante todas as coletas. Foram excluídos da análise os *commits* relativos à exclusão de arquivos, visto que os desenvolvedores não modificaram efetivamente o arquivo deletado. Por isso, foram considerados apenas *commits* relativos à criação e alteração de arquivos. Todos os arquivos recuperados nessa coleta possuem extensão .java. Para melhor analisar a evolução das relações entre desenvolvedores entre versões, consideramos o *commit* realizado pela primeira vez, ou seja, os *commits* não são cumulativos entre as *tags*. Assim, um *commit* realizado na *tag* 1.1 é desconsiderado ao analisar a *tag* 1.2. Esse refinamento possibilita uma visão da evolução da interação entre os desenvolvedores no decorrer da evolução do projeto.

Armazenamento dos dados. Após coletar e refinar os dados provenientes do GitHub, fez-se necessário seu armazenamento para análises posteriores. Dentre as informações armazenadas estão o nome e a URL do repositório, o nome e o caminho dos arquivos, as versões analisadas, os *commits* de cada versão, os desenvolvedores responsáveis pelos *commits* e os arquivos modificados ou criados em cada versão.

Visualização da distribuição de desenvolvedores. Nesta etapa, a ferramenta possibilita a análise de uma determinada versão do repositório. As informações para análise são o

Tabela 1: Top 10 novos desenvolvedores no Hibernate-ORM. Para garantir a privacidade dos desenvolvedores, não apresentamos o nome real.

Autor	# interações(desenvolvedores)	# contribuições(arquivos)
Desenvolvedor1	3	10
Desenvolvedor2	3	4
Desenvolvedor3	-	16
Desenvolvedor4	1	17
Desenvolvedor5	-	26
Desenvolvedor6	-	1
Desenvolvedor7	1	5
Desenvolvedor8	1	6
Desenvolvedor9	2	3
Desenvolvedor10	2	4

número total de *commits* realizados na versão (*tag*), o total de arquivos (.java) criados ou atualizados e o número de desenvolvedores que realizaram contribuições em arquivos na versão em questão. Para compreender a relação entre os desenvolvedores, desenvolvemos uma funcionalidade que propicia a visualização da relação entre os diferentes desenvolvedores que contribuíram nos arquivos atualizados na versão em análise. Na visualização apresentada na Figura 1, os nós são os desenvolvedores e as arestas constituem a relação entre dois desenvolvedores. Por meio desse recurso, é possível observar a quantidade de arquivos que cada autor contribuiu e a força de interação entre os desenvolvedores. Em cada vértice tem-se a quantidade de arquivos que os dois desenvolvedores contribuíram em comum nessa versão. A Figura 1 também mostra que a interação entre Ana e João é maior do que entre Ana e Maria, visto que eles contribuíram em 10 arquivos em comum. Por outro lado, João e Maria não tiveram nenhuma interação durante essa versão do projeto, pois não houve nenhuma interseção entre suas contribuições.

Análise comparativa. Esta funcionalidade da *AuthView* assemelha-se com a apresentada na Figura 1. Nesta última etapa, é possível realizar uma análise comparativa entre duas versões de um mesmo repositório. Dentre as informações apresentadas pela ferramenta, tem-se o número de desenvolvedores que entraram e saíram do projeto nas versões comparadas. O número de desenvolvedores que entraram corresponde ao total de desenvolvedores que não contribuíram na versão 1, mas contribuíram na versão 2. O número de desenvolvedores que saíram, representam o número de desenvolvedores que contribuíram na versão 1, mas não realizaram contribuições na segunda versão analisada. Além disso, também é possível comparar a visualização da distribuição dos desenvolvedores nas duas versões e os dados sumarizados de *commits*, arquivos e desenvolvedores em cada versão.

5. Resultados

Nesta seção, apresentamos os resultados para as três questões de pesquisa. Este trabalho considera oito versões do Hibernate-ORM a partir da versão 3.6.0.Beta1, que é a primeira presente no repositório deste projeto no GitHub, até a versão 3.6.1.Final.

***(Q1)* Como os desenvolvedores começam a contribuir no projeto?** Para responder esta pergunta, foi realizada uma análise sobre a interação entre 29 desenvolvedores. Dentre

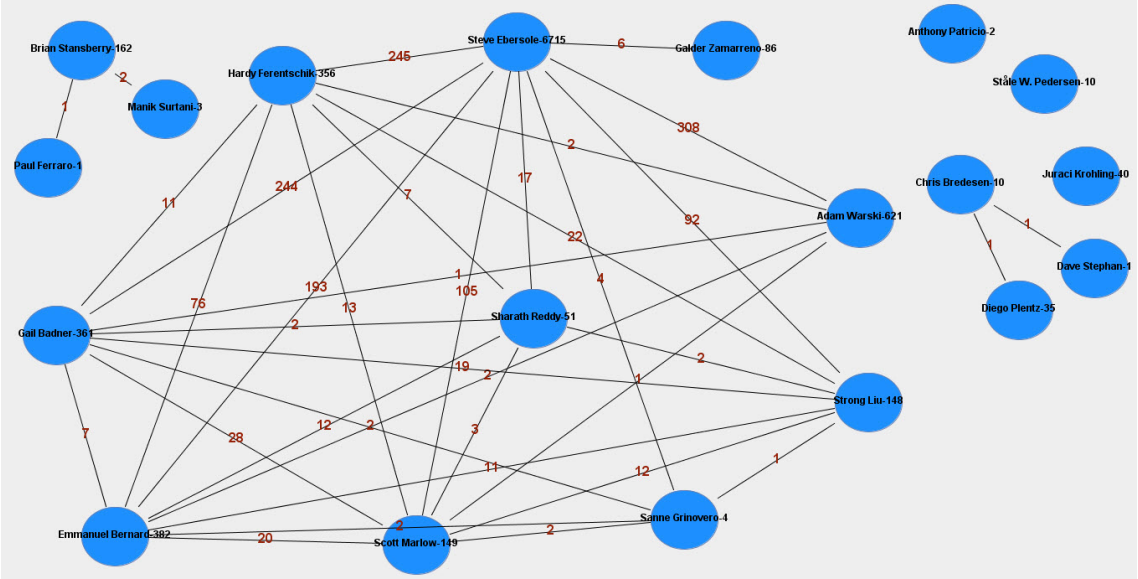

Figura 2: Visualização dos desenvolvedores na versão 3.6.0.Beta1 do Hibernate-ORM.

esse total, apenas 19 desenvolvedores realizam contribuições na primeira versão analisada do projeto. A identificação dos novos autores é feita por meio da análise comparativa entre os autores que contribuíram na versão *i* e aqueles que contribuíram na versão *i+1*. Para responder a essa questão, avaliamos o comportamento dos 10 desenvolvedores que ingressaram no projeto após a versão inicial. A Tabela 1 mostra que os novos desenvolvedores tendem a interagir com poucos desenvolvedores e a realizar poucas contribuições (medida pela quantidade de arquivos modificados e/ou criados) ao ingressar no projeto.

***(Q2)* Como os membros compartilham entre si o conhecimento do sistema no decorrer do tempo?** Para responder esta questão, foi analisado como a interação entre os desenvolvedores mantém-se nas versões analisadas. No entanto, nas oito versões analisadas, apenas três versões revelam interações entre os desenvolvedores. As versões sem interação entre os desenvolvedores são versões pequenas, possivelmente originadas para corrigir problemas pontuais no projeto sem o desenvolvimento de novas funcionalidades. Isso pode ser melhor compreendido pela quantidade reduzida de arquivos em que cada autor contribuiu nesses casos. Nas versões em que foi possível constatar interações entre os desenvolvedores, percebeu-se notoriamente uma distinção entre o comportamento dos pequenos e grandes desenvolvedores (também chamados de *heroes*). Os pequenos desenvolvedores, isto é, aqueles que realizam poucas contribuições no projeto, se relacionam em geral com um ou dois desenvolvedores, enquanto os *heroes*, aqueles que realizam muitas contribuições no projeto, possuem número elevado de interações entre desenvolvedores, que variam entre pequenos desenvolvedores e outros *heroes*.

***(Q3)* Há a formação de times menores no projeto?** Para responder essa questão, observou-se o comportamento de todos os desenvolvedores nas oito versões. Nesta etapa, a funcionalidade de visualização da rede de desenvolvedores permite observar a existência de alguns pequenos grupos de aproximadamente três desenvolvedores que possuem interações em comum. Entretanto, tais pequenos grupos não podem ser considerados times de desenvolvimento, pois o número de arquivos que esses desenvolvedores interagiram é

pequeno. Em contrapartida, observou-se uma grande interação entre a maioria dos desenvolvedores, conforme mostra a Figura 2. Isso mostra que não há pequenos times dentro do projeto, mas um único e grande time de desenvolvimento bem conectados.

6. Conclusões

Este trabalho analisou a formação e evolução dos times de desenvolvimento do GitHub, utilizando como caso de estudo o repositório Hibernate-ORM. A *AuthView* foi criada para auxiliar na coleta, armazenamento, análise e comparação dos dados. A análise buscou responder três questões de pesquisa. A primeira questão revela que os desenvolvedores tendem a começar a contribuir no projeto a partir de arquivos pontuais e não se relacionam de forma significativa com o grande time de desenvolvimento do projeto. A segunda questão apresenta como o conhecimento entre os desenvolvedores é distribuído no decorrer do tempo, percebe-se que os grandes desenvolvedores concentram a maior parcela do conhecimento do software, mas distribui esse conhecimento entre si. Por outro lado, um pequeno autor tende a compartilhar o conhecimento de um módulo do projeto sempre com um outro *heroe*. A terceira questão de pesquisa estuda como os desenvolvedores estão organizados no Hibernate-ORM. Percebeu-se que não há a formação de pequenos times internos de desenvolvimento, mas existe um grande time de desenvolvedores responsáveis pelo projeto conectados entre si. Como trabalhos futuros, pretendemos analisar mais versões do Hibernate-ORM, considerando *major* e *minor releases*, avaliar repositórios de diferentes linguagens de programação, bem como analisar as contribuições e o comportamento dos grandes desenvolvedores em outros repositórios do GitHub.

Agradecimentos. Trabalho parcialmente financiado por CAPES, CNPq e FAPEMIG.

Referências

Alves et al., G. B. (2016). The strength of social coding collaboration on github. In *Anais do SBBD*, pages 247–252, Salvador, Brasil.

Batista et al., A. N. (2017). Collaboration strength metrics and analyses on github. In *IEEE/WIC/ACM WI*, pages 1277–1286, Leipzig, Alemanhã.

Casalnuovo et al., C. (2015). Developer onboarding in github: the role of prior social links and language experience. In *ESEC/FSE*, pages 817–828, Bergamo, Itália.

Dabbish, L., Stuart, C., Tsay, J., and Herbsleb, J. (2012). Social coding in github: transparency and collaboration in an open software repository. In *CSCW*, pages 1277–1286, Seattle, USA.

Greiler, M., Herzig, K., and Czerwonka, J. (2015). Code ownership and software quality: a replication study. In *MSR*, pages 2–12, Florença, Itália.

Rahman, F. and Devanbu, P. (2011). Ownership, experience and defects: a fine-grained study of authorship. In *ICSE*, pages 491–500, Honolulu, USA.

Rocha, L. M. A., Silva, T. H. P., and Moro, M. M. (2016). Análise da contribuição para código entre repositórios do github. In *Anais do SBBD*, pages 103–108, Salvador, Brasil.

Tsay, J., Dabbish, L., and Herbsleb, J. (2014). Influence of social and technical factors for evaluating contribution in github. In *ICSE*, pages 356–366, Hyderabad, Índia.