Minerando Padrões de Interação de Programadores com Repositórios na Plataforma GitHub

Victor Costa[1], Lesandro Ponciano[1]

[1]PUC Minas em Contagem
Bacharelado em Sistemas de Informação

vitortheles@yahoo.com.br, lesandrop@pucminas.br

Resumo. *A ascensão da comunidade desenvolvedora de códigos abertos vem propiciando também à expansão da área de estudo definida como Mineração de Repositórios de Software. As informações obtidas a partir dos comportamentos dos programadores nos repositórios são de grande valia, uma vez que, após analisados, contribuem no reconhecimento de tendências e padrões de projetos de software. Nesse contexto, propõe-se investigar o comportamento dos programadores ao usarem repositórios mantidos publicamente na plataforma GitHub, bem como a influência desses comportamentos no sucesso de tais repositórios. A partir de métricas quantitativas, busca-se identificar, compreender e interpretar os padrões de engajamento dos programadores empregando técnicas de mineração nos dados dos repositórios mantidos publicamente na plataforma GitHub. O estudo é baseado em dados coletados do repositório Keystone. O resultados apontam a existência de seis agrupamentos distintos quanto à interação dos programadores na plataforma GitHub, denominados de "Programador Arquivador", "Programador Básico", "Programador Codificador", "Programador Inerte", "Programador Produtivo" e "Programador Socializador". A análise desses perfis permite uma maior compreensão sobre a influência dos comportamentos dos programadores com os níveis de sucesso de seus repositórios.*

1. Introdução

Nos últimos anos diversas ferramentas, métodos e técnicas têm sido desenvolvidas para serem utilizadas ao se aplicar Engenharia de Software na construção de seus artefatos [Carvalho et al. 2001]. Exemplos dessas ferramentas são aplicações para automatização de testes, editores de diagramas, utilitários ágeis para prototipagem e repositórios de códigos fonte. Os repositórios podem ser definidos como locais onde são armazenados componentes produzidos durante a implementação e evolução de um software [Santos et al. 2015]. Essa evolução, por sua vez, fica a cargo dos programadores aliados ao trabalho que, a partir desses repositórios podem, basicamente, obter e publicar novas versões de códigos, além de registrar suas documentações.

Aos poucos, têm-se destacado os potenciais de se utilizar informações armazenadas nos repositórios para se entender padrões de desenvolvimento, comportamento de programadores, bem como a dinâmica das equipes que colaboram

produzindo código em um mesmo repositório [Stamelos et al. 2015]. Para atender tal demanda surgiu na Engenharia de Software a área de estudo definida como Mineração de Repositórios de Software (MSR, *Mining Software Repositories*). Este campo de pesquisa, por sua vez, tem se mostrado promissor para Engenharia de Software em decorrência das significativas inferências que podem ser realizadas a partir dos dados armazenados em repositórios [Hassan et al. 2008].

Considerando o contexto apresentado, o estudo do documento em questão propõe-se investigar o seguinte problema: **Como os programadores se comportam ao usarem os repositórios de software e como esse comportamento influencia no sucesso dos repositórios nos quais eles atuam**? Questionamento que pode ser justificado pela possibilidade da descoberta de tendências, seguidas de recomendações aos programadores que buscam qualidade em seus repositórios. Tais tendências podem ser obtidas pela mineração dos dados dos repositórios. Dados úteis, inicialmente para fins específicos (como, por exemplo, para simples conferências), mas quando devidamente analisados contribuem na identificação de aspectos comportamentais (práticos) dos programadores, fornecendo fundamentos de forma a justificar futuras ações para mitigação de erros técnicos e gerenciais no desenvolvimento de softwares.

Como objetivo principal do estudo, pretende-se **caracterizar o comportamento dos programadores ao interagirem com repositórios de software e relacionar tal comportamento com métricas de sucesso dos repositórios.** Tanto os critérios de caracterização de comportamentos quanto às métricas de sucesso dos repositórios serão definidos nesta pesquisa. Para alcançar o objetivo principal, almeja-se atingir os seguintes objetivos específicos: i) definir as métricas de sucesso de repositórios; ii) coletar, na plataforma GitHub[1], dados de uma amostra de repositórios; iii) obter dados de interação dos programadores para, posteriormente, identificar perfis de engajamento; iv) relacionar os perfis estabelecidos com os dados das amostras de repositórios coletados.

2. Referencial Teórico

Nesta seção são apresentados os principais conceitos e classificações relacionadas aos repositórios de software. Diante de tais definições, a plataforma GitHub (exemplo de recurso para armazenamento de repositórios de software) e as principais ações realizadas pelos programadores são devidamente caracterizadas de forma a possibilitar a compreensão das conclusões e atividades realizadas no decorrer deste estudo. Do mesmo modo, ao fim da seção, discute-se as principais ideias propostas pela área de estudo MRS, ao qual se fundamentam às atividades desta pesquisa.

[1] Disponível em: <https://github.com/> Acesso em: 02 mar, 2018.

2.1. Repositórios de Software

Os repositórios de software consistem basicamente em locais para controle de versão, rastreamento de defeitos e comunicação entre programadores aliados no desenvolvimento de uma solução computacional [D'Ambros et al. 2008]. Seu objetivo é promover o uso colaborativo, oferecendo acesso remoto a módulos e/ou pacotes de códigos com o intuito de corrigir, incrementar funcionalidades ou pesquisar soluções bem-sucedidas para replicação.

2.1.1. Sobre a Plataforma GitHub

Conforme apresentado, o estudo em questão utiliza a plataforma GitHub como recurso de armazenamento de repositórios de software a ser explorado para obtenção das conclusões e previsões derivadas dos dados coletados. Lançada em Fevereiro de 2008, a plataforma GitHub consiste em uma aplicação de código aberto baseada no sistema de controle de versão distribuído Git, que já hospeda mais de 10 milhões de repositórios [Kalliamvakou et al. 2014]. Os arquivos são hospedados em redes de servidores remotos disponíveis para acesso online, de qualquer dispositivo com conexão à Internet.

A plataforma GitHub, por sua vez, além dos serviços comuns do Git, oferece recursos populares à redes sociais de forma a possibilitar e incentivar a comunicação entre os programadores. Com um viés colaborativo, a plataforma GitHub permite que o programador compartilhe blocos de códigos, comente e até mesmo altere arquivos em repositórios de outros programadores, seja para corrigir ou mesmo incluir novas funcionalidades.

2.1.2. Comportamento dos Programadores na Plataforma GitHub

Na utilização da plataforma GitHub, os programadores executam frequentemente instruções mestres para o gerenciamento dos arquivos, de forma a manter a consistência dos códigos alterados e/ou incluídos junto a versão principal do repositório. Há diversas instruções. Elas diferem entre si quanto à utilidade provida ao programador. As instruções frequentemente usadas pelos programadores revelam o comportamento de interação típico deles com os repositórios. Nos próximos parágrafos são discutidas algumas das principais instruções.

Inicialmente, se faz necessária a execução da instrução *Clone*, responsável por transferir os arquivos do repositório publicados nos servidores da plataforma para a máquina no programador. Nos casos em que já houver alguma versão do repositório na máquina, utiliza-se a execução da instrução *Pull*, garantindo que a versão dos códigos seja a mesma nos repositórios locais e nos servidores *online*. A medida em que os arquivos são alterados, deve-se executar a instrução *Commit*, onde todos os arquivos incluídos no caminho prévio de publicação (chamado de *Index, Stage* ou *Staging Area)* são submetidos aos servidores da plataforma. O *Commit*, por sua vez, cria uma revisão numerada e um comentário com a descrição da alteração submetida. Por fim, a instrução *Push* deve ser executada de forma a publicar os arquivos de todos os *commits* realizados, unindo à versão principal do conjunto de códigos.

Além das instruções básicas descritas acima, eventualmente é utilizada a instrução conceituada como *Branch,* que ramifica o repositório em diferentes visões, possibilitando ao programador atuar em arquivos de forma independente sem ter que alterar a versão principal antes de sua publicação oficial. Em seguida, a plataforma promove a união dos arquivos nos diretórios das diferentes visões. Para os casos em que um mesmo arquivo é editado de diferentes formas em *branches* distintas, se faz necessário o emprego da instrução *Merge,* que une dos arquivos e soluciona (a partir da inclusão de indicadores dentro do código) os eventuais conflitos de alterações nas mesmas linhas.

2.2. Mineração de Repositórios de Software

O crescente desenvolvimento das comunidades software livre tem proporcionado a evolução da área de estudo definida como MRS. Esta área inclui atividades de análise, mineração e recuperação de dados, seguida de uma investigação estatística dos repositórios de software armazenados em ambientes distribuídos. Comumente, são utilizadas técnicas estatísticas como correlação e regressão, de forma a identificar relações entre as variantes dos repositórios que tiveram seus dados coletados [Gokul Dhakal, 2016].

Pelos repositórios oferecerem dados históricos de evolução de códigos, as contribuições dos programadores (como linhas de código-fonte, frequência de *commits*, número de casos de teste executados, correlação entre casos de teste gerados, entre outros) e marcos no desenvolvimento do software (como número de submissões realizadas durante as fases de revisão, iteração, desenvolvimento e testes), permitem a investigação e posterior extração de informações com benefícios potenciais para reconhecimento de tendências e padrões.

Para este estudo, foram coletados dados de comportamentos e contribuições dos programadores nos repositórios a modo de definir perfis de interação. Em seguida, esses foram relacionados aos repositórios classificados com seus níveis de sucesso, possibilitando a descoberta e registro das conclusões.

3. Trabalhos Relacionados

Nesta seção são apresentados os trabalhos relacionados à mineração de repositórios de software. Apesar de não tratarem especificamente da análise de perfil dos programadores, eles apresentam informações relevantes para nortear as abordagens do estudo reportado neste artigo.

Hassan (2008) considera como preceito o fato de que programadores se apoiam basicamente em experiências de trabalhos anteriores para apoio e resolução de erros, definição de complexidade de códigos, realização de testes, entre outros. Esta situação revela a grande vantagem gerada pela análise dos dados dispostos nos repositórios, tendo em vista que se permite minimizar a dependência sobre as experiências e intuições dos profissionais, contribuindo também na difusão do conhecimento. Baseado nesses benefícios, Hassan (2008) apresenta as informações históricas relevantes da área

de estudo MSR, mas enfatiza a discussão de suas oportunidades e desafios futuros. Dentre as oportunidades do MSR, têm-se a criação de técnicas para automatizar e assim aprimorar a extração de informações dos repositórios, bem como suas validações aplicadas. O desafio mais pertinente identificado compreende a limitação dos dados disponibilizados pelos repositórios. Esses, por si só, demonstraram não serem suficientes para determinação da causalidade de suas correlações, uma vez que o fator (inicialmente intangível) "contexto do repositório" (e de seus envolvidos, programadores) deve ser devidamente ponderado.

No estudo de Chatziasimidis et al. (2015), a proposta foi mensurar as características influentes dos programadores para um repositório de sucesso na plataforma GitHub. Apesar da adequada coleta e a análise dos dados, as diretrizes e padrões descobertos são fortemente ameaçados pelo fato de se ter assumido como pressuposto que somente os repositórios com mais de 1000 (mil) *downloads* eram considerados de sucesso. O fato de somente uma variável (quantidade de *downloads*) determinar o sucesso ou não do repositório desfavorece a confiabilidade das regras identificadas, bem como sua propagação à comunidade de programadores.

De acordo com Halloran et al. (2002), a qualidade de um software de código aberto pode ser equiparar aos softwares comerciais e governamentais (considerados, em sua maioria, de excelência no aspecto de qualidade). O êxito dos softwares de código aberto se dá, também, por estarem armazenados em repositórios "populares", onde os programadores colaborativos já estejam familiarizados e dispostos a contribuir e agregar valor. Diante disso, o estudo visou investigar as variáveis determinísticas que torna um software de código aberto bem-sucedido. Dentre as variáveis observadas têm-se: a presença de testes de regressão no processo de compilação e a implementação de ferramenta para rastreamento de erro.

As discussões de Chatziasimidis et al. (2015) e Halloran et al. (2002) se convergem ao tema proposto para o estudo deste documento na perspectiva de que suas metas eram identificar variáveis e regras influentes ao êxito de um repositório, a partir das práticas de seus programadores. Apesar disso, nessas discussões, as características dos programadores foram obtidas e analisadas sem uma investigação e seleção prévia de seus contextos - variável relevante e diretamente influente nos resultados dos estudos, segundo Hassan (2008). Diante disso, a pesquisa deste trabalho também propõe suprir tais lacunas, atribuindo à devida importância a seleção dos programadores e repositórios que terão seus dados coletados, de modo a obter conclusões consistentes e reais na determinação dos perfis de interação e, consequentemente, sua associação com os diferentes níveis de sucesso dos repositórios.

4. Metodologia

O estudo apresentado neste trabalho é classificado como pesquisa quantitativa. A partir de métricas quantitativas, busca-se identificar, compreender e interpretar padrões de engajamento dos programadores empregando técnicas de mineração de dados em uma amostra de repositórios da plataforma GitHub. Salienta-se que a natureza exploratória desta pesquisa não se motiva aos dados genuínos obtidos nas coletas, mas sim, às informações que puderam ser inferidas a partir de tais dados. Nesta seção serão apresentadas as metodologias utilizadas no desenvolvimento do estudo para alcance de seus objetivos.

4.1. Definição das Métricas

As métricas utilizadas para caracterização do comportamento dos programadores e do sucesso dos repositórios basearam-se nas informações disponíveis à recuperação, a partir da Interface de Programação de Aplicações (API, do inglês *Application Programming Interface*) da plataforma GitHub. Foram utilizadas as métricas:

- **Quantidade de seguidores do programador**. Esta métrica representa a quantidade de usuários (também programadores) da plataforma GitHub que registraram o interesse em receber as atualizações e interações de determinado programador na plataforma GitHub. Quanto maior a quantidade de seguidores, maior será a abrangência de visualização e, eventualmente, de colaboração em seus repositórios;
- **Quantidade de usuários que o programador segue**. Esta métrica representa a quantidade de programadores que um determinado usuário (também programador) opta por receber as atualizações e interações na plataforma GitHub;
- **Quantidade de repositórios públicos**. Esta métrica representa a quantidade de repositórios públicos ativos de um programador na plataforma GitHub. São com esses repositórios que os programadores realizam suas contribuições;
- **Quantidade total de *pull requests***. O *pull request* representa uma submissão de código para integração ao ramo principal de determinado repositório. Esta métrica, por sua vez, consiste no somatório dos *pull requests* realizados pelos programadores em todos os repositórios de sua contribuição;
- **Quantidade total de *commits***. O *commit* representa a publicação e gravação de alteração em um ou mais arquivos, com o objetivo de registrar e possibilitar a rastreabilidade dos códigos dos repositórios. Esta métrica, por sua vez, consiste no somatório dos *commits* realizados pelos programadores em todos os repositórios de sua contribuição;
- **Quantidade total de *issues.*** O *issue* representa a ação de o programador reportar um problema de determinado repositório. Esta métrica, por sua vez,

consiste no somatório dos *issues* reportados pelos programadores em repositórios da plataforma GitHub (sendo de sua contribuição ou não).

A métrica *nível de sucesso dos repositórios* é um valor único, derivado do somatório das métricas *quantidade de estrelas* com a *quantidade de forks*, dividido pela *quantidade de dias de criação do repositório*. Cálculo necessário para que a informação do *nível de sucesso dos repositórios* coletados seja consistente e independente da variável de tempo de existência do repositório. O resultado dessa métrica é enquadrado na regra a seguir de forma a determinar a classificação do nível de sucesso dos repositórios:

- **Excepcional.** O repositório é classificado como excepcional se o resultado do nível de sucesso dos repositórios for maior ou igual a 95% percentil do conjunto total de resultados;
- **Muito Sucesso.** O repositório é classificado como de muito sucesso se o resultado do nível de sucesso dos repositórios estiver entre 90% e 95% percentil do conjunto total de resultados;
- **Sucesso.** O repositório é classificado como de sucesso se o resultado do nível de sucesso dos repositórios estiver entre 85% e 90% percentil do conjunto total de resultados;
- **Pouco Sucesso.** O repositório é classificado como de sucesso se o resultado do nível de sucesso dos repositórios estiver abaixo de 85% percentil do conjunto total de resultados.

4.2. Conjunto de Dados

O conjunto de dados foi coletado a partir de uma aplicação desenvolvida exclusivamente para este estudo. Esta aplicação, por sua vez, foi responsável pela chamada dos serviços da API da plataforma GitHub, para obtenção dos dados dos programadores e repositórios. Com base no "repositório semente" Keystone (criado pelo usuário denominado Openstack), obteve-se uma lista de 325 programadores que se relacionam com, no total, cerca de 11.197 repositórios - dados extraídos em Setembro/2018. Foram coletados os dados de tais programadores e dos repositórios a eles associados.

4.3. Estratégia de Normalização e Agrupamento

A análise de agrupamento é aplicada, pois se trata de um relevante procedimento para o desenvolvimento e proposição de hipóteses a respeito da natureza de um conjunto de dados. De forma a possibilitar a realização do agrupamento, se fez necessária a normalização das métricas de comportamento dos programadores. O objetivo foi reduzir a dispersão heterogênea entre as variáveis de cada métrica. Para tal, foi utilizada a técnica da normalização segundo a amplitude (*min-max*), que consiste em subtrair o

valor absoluto da variável pelo menor valor variável da métrica (valor mínimo), dividido pela também subtração do valor máximo pelo valor mínimo (dentre as variáveis da métrica). Ao final desta normalização, as variáveis de todas as métricas se encontraram, consistentemente, entre [0,1].

4.4. Estratégia de Agrupamento e Seleção do K

Como estratégia de agrupamento para o estudo em questão, utilizou-se do algoritmo de agrupamento *K-Means*. Este algoritmo visa minimizar a distância dos elementos a um conjunto de "k" centros (também chamados de *clusters*), calculados de forma iterativa. A distância entre um ponto e um conjunto de *clusters* é dada pela distância do ponto ao centro mais próximo dele.

O algoritmo *K-Means* depende do parâmetro *k* (número de *clusters*) definido pelo pesquisador. A determinação do *k* é realizada a partir do *Elbow method* que consiste no cálculo do algoritmo (*K-Means*) para diferentes números de *k*. A partir disso, é analisada a relação de cada *k* com os valores das somas dos quadrados dos *clusters* obtidos e, assim, a quantidade de *k* é determinada a partir do ponto que apresentar uma maior variação, ou seja, que a probabilidade de erro é reduzida de forma significativa.

O software *R Studio*[2] foi utilizado como ferramenta principal para manipulação da base de dados. Tal ferramenta detém de uma diferencial capacidade para processar grandes volumes de dados, além de oferecer funcionalidades acessíveis para utilização do algoritmo *K-Means* e fornecer recursos gráficos para apresentação dos resultados.

5. Resultados Obtidos

Nesta seção são apresentados os resultados obtidos. A princípio, os dados foram extraídos e devidamente normalizados de forma a possibilitar uma efetiva análise e identificação de suas relações e influências.

5.1. Análise Descritiva

As métricas de comportamento dos programadores foram previamente normalizadas, uma vez que apresentaram uma significativa dispersão entre suas variáveis. Na tentativa de possibilitar melhor visualização das variáveis das métricas, recorreu-se aos gráficos de Boxplot para ilustrar a distribuição havida. Os resultados são apresentados nas Figuras 1 e 2. A Figura 1 apresenta os valores não normalizados. A Figura 2 apresenta os valores normalizados, favorecendo a posterior análise e identificação de relações por meio do algoritmo de agrupamento.

[2] Disponível em: <https://www.rstudio.com/> Acesso em: 28 abr, 2018.

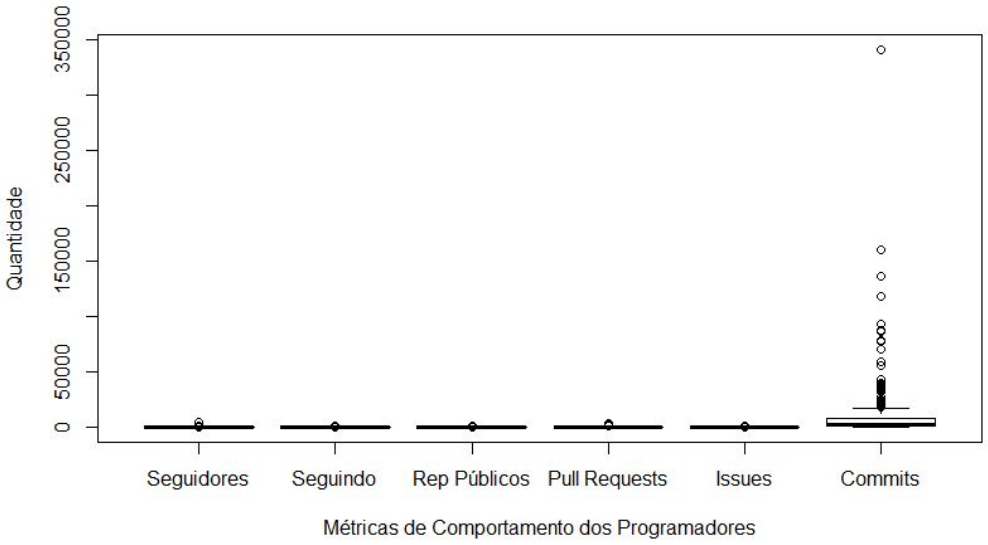

Figura 1. Boxplot das métricas de comportamento dos programadores antes da normalização.

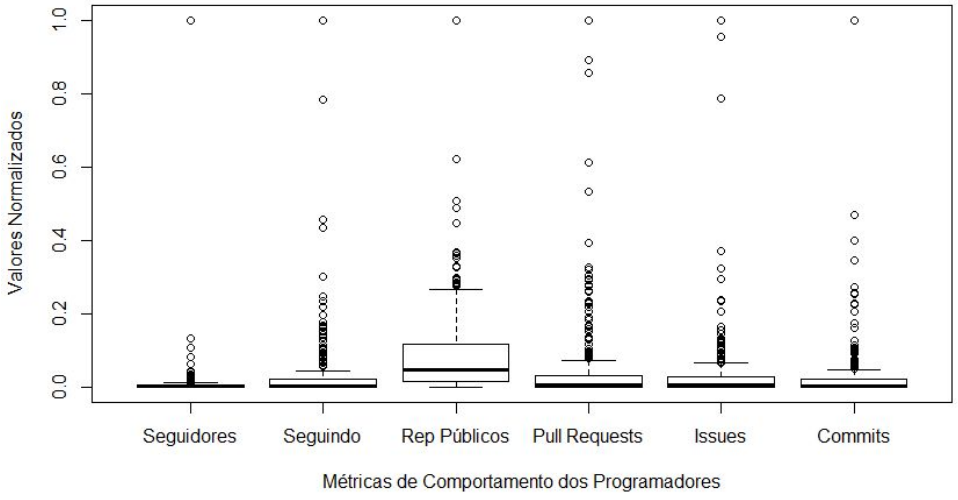

Figura 2. Boxplot das métricas de comportamento dos programadores após a normalização.

Observa-se nas Figuras 1 e 2 que todas as métricas possuem baixas medianas e amplitudes com uma grande quantidade de *outliers*, ou seja, grande quantidade de valores discrepantes. Se comparada as demais métricas, a quantidade de repositórios públicos é a que apresenta a maior variabilidade em seus dados. Destaca-se que todas possuem uma alta concentração em um pequeno grupo de valores, excetuando a métrica de quantidade de repositórios públicos.

5.2. Análise do Agrupamento

A identificação de perfis foi realizada a partir do agrupamento dos programadores, de acordo com seus comportamentos (quantidade de seguidores, programadores seguindo, repositórios públicos, *pull requests*, *commits* e *issues*) extraídos da plataforma GitHub. Diante de 11.086 registros de interações de programadores com repositórios, analisou-se a quantidade de *clusters*, conforme *Elbow method*. A Figura 3 apresenta os valores das somas dos quadrados, variando de acordo com a quantidade de *clusters* - neste caso, de 2 a 10 *clusters*.

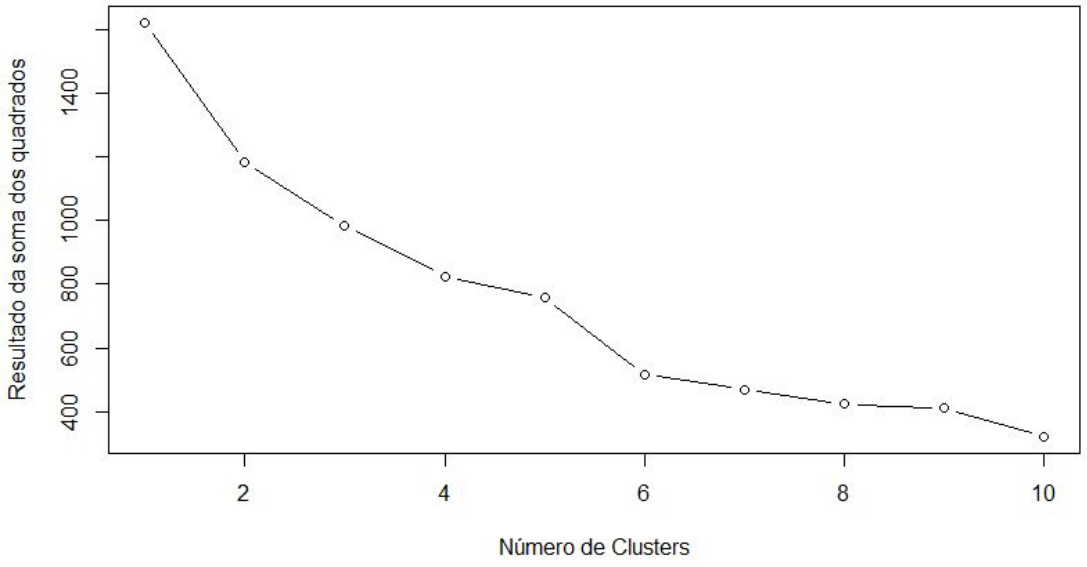

Figura 3. Análise da quantidade de *clusters* a partir do *Elbow method.*

Conforme apresentado nessa figura, tem-se que o número de *clusters* para o estudo em questão é 6. Ou seja, a partir dos dados coletados podem-se extrair 6 diferentes perfis de comportamento dos programadores na plataforma GitHub, de acordo com as métricas quantidade de seguidores, programadores seguindo, repositórios públicos, *pull requests*, *commits* e *issues*. Quantidade de *clusters* justificada pelo fato de ser, no gráfico, o ponto em que o valor das somas dos quadrados é reduzido de forma mais significativa, se comparado aos demais. Após essa quantidade, aumentar o número de *clusters* não implica na redução da probabilidade de erros no agrupamento.

5.3. Perfis dos Programadores

O centróide de cada *cluster*, gerados pelo algoritmo de agrupamento, é analisado e nomeado de forma a promover uma classificação e caracterização dos perfis de interação dos programadores. Em seguida, todos os 325 programadores da coleta foram devidamente enquadrados na classificação, de acordo com a sua proximidade com o centróide do *cluster* em que ele se encontra.

A Figura 4 apresenta a caracterização geral de cada *cluster*, onde o eixo horizontal consiste nos perfis e cada barra representa uma métrica de comportamento dos programadores. O eixo vertical indica o valor normalizado das métricas de comportamento, por perfil.

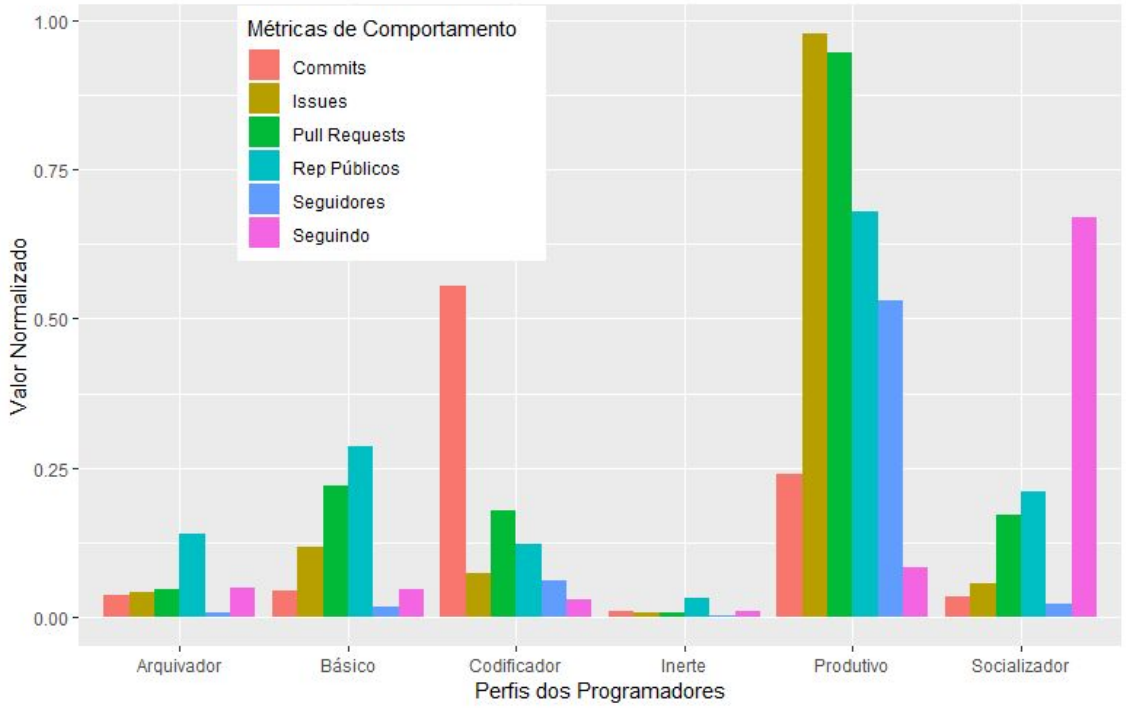

Figura 4. Perfis de interação dos programadores na plataforma GitHub, detalhados por suas métricas de comportamento.

Os parágrafos a seguir apresentam a caracterização detalhada de cada perfil de interação identificado.

Programador Arquivador na plataforma GitHub**.** Não se preocupa em seguir outros programadores e também não é seguido por uma quantidade significativa de programadores. Isto, pois, as métricas quantidade de seguidores e quantidade seguindo são baixas (0,0081 e 0,0496, respectivamente). Também realiza poucas contribuições em seus repositórios públicos (*commits* com 0,0358*, issues* com 0,0419 e *pull requests* com 0,0465), após sua versão inicial. Seu objetivo principal é utilizar a plataforma GitHub como ferramenta de arquivamento de suas aplicações desenvolvidas. Inferência realizada a partir do valor da métrica de quantidade de repositórios públicos, igual a 0,1402. Dentre os programadores da coleta, 8,91% se enquadram neste perfil.

Programador Básico na plataforma GitHub. Tal programador participa moderadamente na realização de contribuições à plataforma. Busca contribuir com frequência na inclusão de *pull requests* (métrica igual a 0,2190) mas seu principal

objetivo é expandir o catálogo de repositórios (métrica quantidade de repositórios públicos igual a 0,2869). Dentre os programadores da coleta, 0,62% se enquadram neste perfil.

Programador Codificador na plataforma GitHub. Destaca-se somente na quantidade de *commits* submetidos (métrica igual a 0,5540). Não se preocupa em ampliar seu catálogo de repositórios (métrica quantidade de repositórios públicos igual a 0,1228), nem em interagir com outros repositórios e programadores (quantidade de seguindo igual a 0,0304). Dentre os programadores da coleta, 63,7% se enquadram neste perfil.

Programador Inerte na plataforma GitHub. Não utiliza a plataforma com frequência e, consequentemente, não se destaca por nenhum comportamento. Quando tem algum repositório público, realiza poucas contribuições. Isto, pois, a média de interação dos programadores com os repositórios (quantidade de *commits*, *issues* e *pull requests* é de 0,0077) . Dentre os programadores da coleta, 24,31% se enquadram neste perfil.

Programador Produtivo na plataforma GitHub. Tal programador não se preocupa em seguir outros usuários (métrica de quantidade seguindo igual a 0,0826). Apesar disso, realiza frequentes alterações em códigos, submissões de *pull requests* (0,1704) e *issues* (0,0562) e, consequentemente, acaba despertando o interesse de outros programadores (elevada quantidade de seguidores, com o valor da métrica igual a 0,5309). Dentre os programadores da coleta, 1,24% se enquadram neste perfil.

Programador Socializador na plataforma GitHub. Realiza poucas contribuições em seus repositórios públicos (*commits, issues* e *pull requests* com média igual a 0,0870) e também não influencia outros programadores (quantidade de seguidores igual a 0,0213). Seu objetivo é seguir outros programadores, supostamente para obter novos conhecimentos. Isto, pois, a quantidade de programadores seguindo é de 0,6685. Dentre os programadores da coleta, 1,24% se enquadram neste perfil.

5.4. Relação de Perfis de Interação com o Sucesso dos Repositórios

Definidos os perfis de interação dos programadores na plataforma GitHub, foram realizados os cálculos de forma a determinar os índices de colaboração dos programadores (segregados por perfis) sobre os diferentes níveis de sucesso dos repositórios, classificados como *Excepcional*, *Muito Sucesso*, *Sucesso* e *Pouco Sucesso*. Esta relação, por sua vez, pode ser notada a partir da Figura 5.

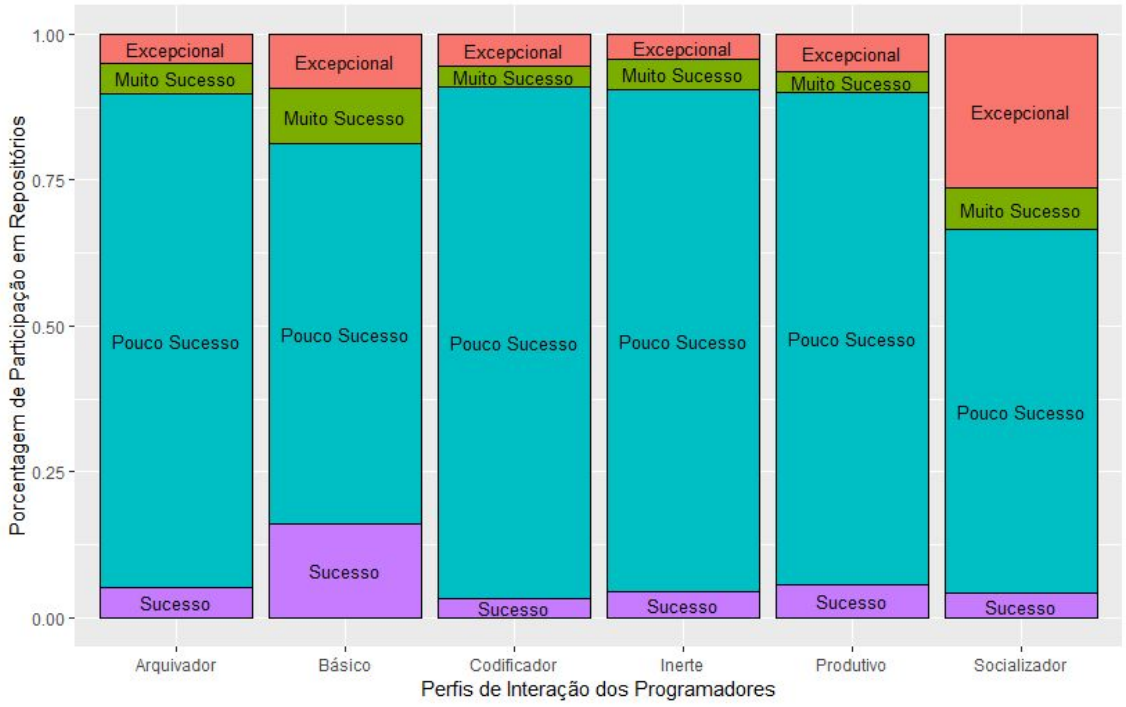

Figura 5. Interação dos programadores (segmentados por perfis) com os repositórios de diferentes níveis de sucesso.

Verifica-se, a partir da Figura 5, que o perfil de programador Socializador é o que mais interage (26,69%) com os repositórios classificados como Excepcionais. Se comparado aos demais perfis também, o Socializador também é o perfil que menos interage com os repositórios classificados como Pouco Sucesso (62,37%). O perfil de programador Básico colabora mais com repositórios de Pouco Sucesso (65,17%) mas se destaca por também contribuir com os repositórios de Sucesso (16,10%). Os demais perfis de programadores (Arquivador, Codificador, Inerte e Produtivo) apresentam níveis semelhantes de colaboração com repositórios, predominando as porcentagens de colaboração com os repositórios de Pouco Sucesso.

A Figura 6 apresenta o inverso da relação apresentada na Figura 5. Nessa, é apresentada a participação dos perfis de programadores por nível de sucesso dos repositórios.

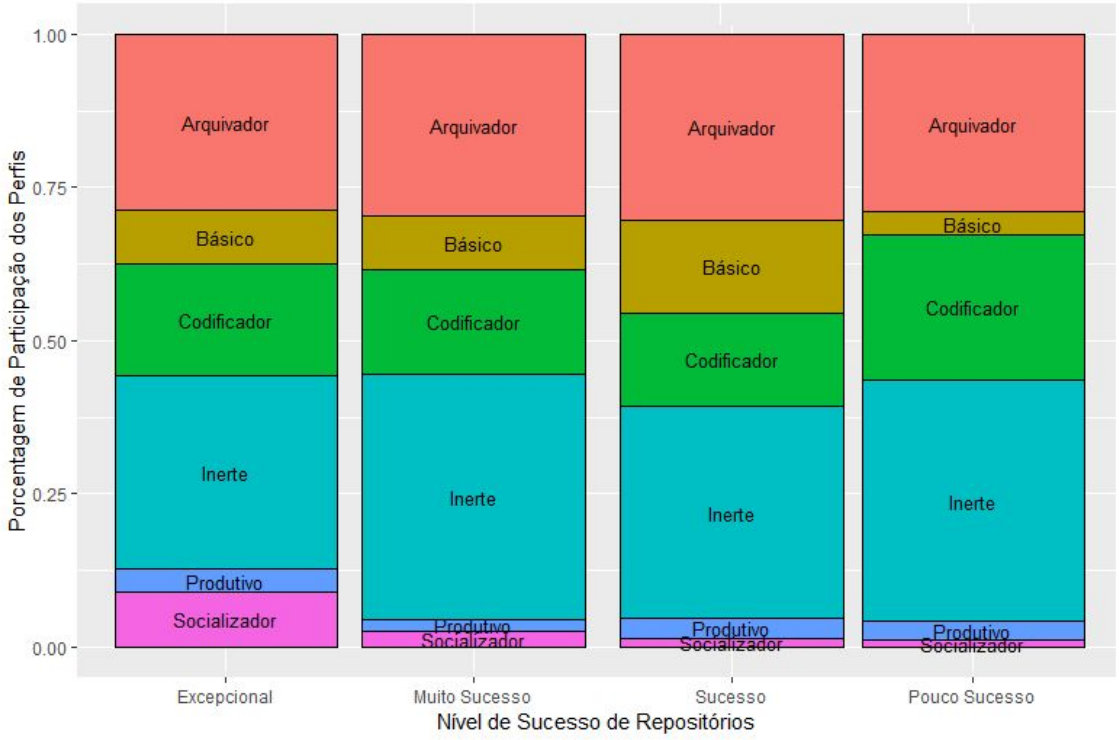

Figura 6. Participação dos programadores (segmentados por perfis) com os repositórios de diferentes níveis de sucesso.

Evidencia-se, a partir da Figura 6, que os repositórios de Pouco Sucesso estão mais relacionados com os programadores do perfil Codificador, do que os demais (23,74%). O perfil Codificador interage menos com os repositórios de Sucesso (15,42%), mas têm porcentagens semelhantes com os repositórios Excepcionais e de Muito Sucesso. O perfil de programador Inerte predomina e o perfil Arquivador demonstra ser igualmente distribuído sobre as quatro classificações de sucesso dos repositórios. Destaca-se a porcentagem de participação do perfil Socializador nos repositórios classificados como Excepcional (9,01%).

5.5. Discussão

O desenvolvimento da pesquisa, realizada a partir dos dados da plataforma GitHub, revelou a existência de 6 perfis de interação dos programadores com repositórios. Os perfis identificados são relevantes e significativamente diferentes entre si. Tais perfis contribuem na compreensão de como os diferentes padrões de engajamento podem resultar em diferentes níveis de sucesso dos repositórios.

Quatro perfis de interação, nomeados como Programador Arquivador, Codificador, Inerte e Produtivo, apresentam porcentagens semelhantes em relação às suas influências nos níveis de sucesso dos repositórios. Apesar disso, manifestam padrões de comportamento distintos, como por exemplo, a diferença polarizada entre os

perfis de Programador Produtivo e Programador Inerte. O Programador Produtivo, se comparado aos demais perfis, contém os maiores índices de contribuição com a plataforma GitHub. Em oposição ao perfil Programador Inerte, que possui valores muito reduzidos em relação a todas as métricas de interação com a plataforma.

Observa-se que o perfil Programador Participante se relaciona pontualmente com os repositórios de sucesso. Relevância determinada pelo fato de interagir moderadamente com a plataforma. Apesar de buscar sempre expandir seu catálogo de repositórios públicos, sua principal contribuição é na inclusão de *pull requests*.

O perfil Programador Socializador se destaca em relação aos demais, uma vez que mostra uma relação direta com os repositórios Excepcionais. Supõe-se que a variável determinante para tal evidência é a quantidade de programadores seguindo, ou seja, os programadores socializadores se destacam por manterem repositórios muito bem sucedidos, justamente, pois se socializam muito com demais programadores da plataforma e estão constantemente buscando novas influências e novos conhecimentos.

Diante desta discussão, nota-se que o sucesso dos repositórios não se relaciona diretamente com a supervalorização de todas as métricas de comportamento, mas sim, com uma interação moderada com a plataforma ou mesmo com a busca constante por novas influências e aprimoramentos técnicos. Os repositórios Excepcionais balancearam melhor os diversos perfis de programadores identificados, o que remete-se a ideia de que as equipes de trabalho multidisciplinares tendem a gerar melhores resultados.

6. Conclusões e Trabalhos Futuros

Este estudo se propôs a responder a seguinte questão: como os programadores se comportam ao usarem os repositórios de software e como esse comportamento influencia no sucesso de tal repositório? Para se obter as respostas deste questionamento, inicialmente, foram realizadas pesquisas para embasamento teórico sobre a área de estudo Mineração de Repositórios de Softwares, de forma a identificar os resultados já descobertos por outros pesquisadores. Posteriormente, foram definidas as métricas que determinam o comportamento dos programadores e também métricas que determinam os níveis de sucesso dos repositórios. A partir dos dados coletados da plataforma GitHub, aplicou-se um algoritmo de mineração de dados para identificar os diferentes padrões de interação de programadores com os repositórios e também sua influência em relação com aos seus níveis de sucesso.

Os resultados obtidos mostram que os diferentes perfis de interação dos programadores com os repositórios do GitHub são bem demarcados. Os perfis identificados foram rotulados como programador arquivador, codificador, socializador, inerte, básico e produtivo e, diferem entre si de acordo com um conjunto de métricas que medem o grau de interação dos programadores com os repositórios.

Observou-se que a maior porcentagem de programadores (63,7%) é classificada como Programadores Codificadores. Apesar disso, grande parte dos programadores codificadores interagem com repositórios definidos sendo como de pouco sucesso. Em contrapartida, o perfil de programadores que mais interage com os repositórios classificados como excepcionais são os programadores de perfil Socializador. Estes, por sua vez, representam cerca de 1,24% da amostra total de programadores coletados e, certamente, se destacam por se socializarem com demais programadores da plataforma, buscando constantemente novas influências, conhecimentos e aprimoramentos técnicos.

Em relação aos trabalhos futuros, partindo dos resultados dos perfis identificados neste estudo, sugere-se realizar uma pesquisa qualitativa entre programadores pré-selecionados de cada perfil a fim de detalhar suas variáveis contextuais, para melhor diferenciá-los. Os dados e códigos utilizados nesta pesquisa foram disponibilizados em <https://drive.google.com/file/d/1mDN70dzB85rQRTni3-JLPN-l60JpduCP/view?usp=sharing> e <https://github.com/vitortheles/extratorGitHub>, respectivamente, para assim serem utilizados em eventuais evoluções deste estudo.

Referências Bibliográficas

CARVALHO, A. M. B. R., CHIOSSI, T. C. S. (2001) "Introdução à Engenharia de Software". Editora Unicamp.

CHATZIASIMIDS, Fragkiskos, STAMELOS, Ioannis. (2015) "Data collection and analysis of GitHub repositories and users". In: 6th International Conference on Information, Intelligence, Systems and Applications (IISA), pp. 1-6. IEEE.

D'AMBROS M., GALL H., LANZA M., PINZGER M. (2008) "Analysing Software Repositories to Understand Software Evolution". In: Software Evolution. Springer, Berlin, Heidelberg, pp. 37-67.

GOUKUL Dhakal. (2016) "Mining Software Repositories", Disponível em: http://gokuldhakal.com.np/wp-content/uploads/2016/12/Mining_software_repositories.pdf, Acesso em 29 Abr. 2018.

HALLORAN, T. J., SCHERLIS, W. L. (2002) "High Quality and Open Source Software Practices". In: Meeting Challenges and Surviving Success: 2nd Workshop on Open Source Software Engineering, ICSE 24, Orlando, USA (pp. 43-68).

HASSAN, Ahmed E. (2008) "The road ahead for Mining Software Repositories". In: Frontiers of Software Maintenance, FoSM, pp. 48-57. IEEE.

KAGDI H. H., COLLARD M. L., MALETIC J. I. (2007) "A survey and taxonomy of approaches for mining software repositories in the context of software evolution". Journal of Software Maintenance, pp. 77–131.

KALLAMVAKOU E., GOUSIOS G., BLINCOE K., SINGER L., GERMAN DM., DAMIAN D. (2014) "The promises and perils of mining github". In: Proceedings of the 11th Working Conference on Mining Software Repositories, pp. 92–101.

LINDEN, R. . Técnicas de Agrupamento. Revista de Sistemas de Informação da FSMA, v. 1, p. 18-36, 2009.

SANTOS, Henrique Machado, FLORES Daniel. (2015) “Repositórios digitais confiáveis para documentos arquivísticos: ponderações sobre a preservação em longo prazo”. In: Perspectivas em Ciência da Informação.pp. 198-218.