Universidade de Brasília
Instituto de Ciências Exatas
Departamento de Ciência da Computação

Qual a relevância da literatura open-source sob a perspectiva de profissionais e estudantes de graduação

Michelangelo da Rocha Machado

Monografia apresentada como requisito parcial
para conclusão do Bacharelado em Ciência da Computação

Orientador
Prof.a Dr.a Edna Dias Canedo

Brasília
2019

Universidade de Brasília
Instituto de Ciências Exatas
Departamento de Ciência da Computação

Qual a relevância da literatura open-source sob a perspectiva de profissionais e estudantes de graduação

Michelangelo da Rocha Machado

Monografia apresentada como requisito parcial
para conclusão do Bacharelado em Ciência da Computação

Prof.a Dr.a Edna Dias Canedo (Orientador)
CIC/UnB

Prof. Dr. Jan Mendonça Correa
CIC/UnB

Mestre Walter Lucas Monteiro de Mendonça
CIC/UnB

Prof. Dr. Edison Ishikawa
Coordenador do Bacharelado em Ciência da Computação

Brasília, 13 de dezembro de 2019

Dedicatória

Dedico esse trabalho primeiramente à minha amada Lara, essa mulher incrível que me aconselhou a ingressar na Universidade de Brasília, a única pessoa que esteve ao meu lado desde o primeiro dia em que entrei nessa universidade e assim permaneceu até esse tão sonhado momento, passando comigo pelos momentos mais difíceis que já enfrentei em toda minha vida. Só ela sabe como eu abdiquei de absolutamente tudo em nome dessa graduação e só ela viu de perto o quanto alguém é capaz de se superar mesmo quando isso não parece ser mais possível. Além disso, dedico esse trabalho ao meu pai, Luciano, que me ajudou de todas as maneiras que estavam ao seu alcance e que sempre acreditou que eu fosse capaz de conquistar qualquer coisa. Também dedico à minha mãe, Sandra, que sempre me apoiou em qualquer decisão que eu viesse a tomar, sabendo que eu seria bem sucedido em qualquer uma delas. Não menos importante, dedico essa conquista ao meu tio, Antônio Carlos, o qual é um segundo pai para mim, alguém com quem eu sempre soube que poderia contar, e também à todos da minha família que me acolheram em algum momento.

Agradecimentos

Meu principal agradecimento vai para a Prof.a Dr.a Edna Dias Canedo, essa ótima pesquisadora que me introduziu ao meio científico, sendo a minha mentora desde a primeira disciplina que fiz com ela até o momento da minha graduação. Sem você esse trabalho jamais existiria. Também gostaria de agradecer à todos os professores do departamento de Ciência da Computação da Universidade de Brasília, vocês são incríveis e não tenho como descrever a honra que sinto por ter adquirido meu conhecimento através de tantas mentes incríveis, em particular alguns nomes: Cláudia Nalon, Genaína Rodrigues, Rodrigo Bonifácio, Flávio Moura, Fernando Albuquerque e Guilherme N. Ramos. Mesmo que talvez vocês não saibam, vocês me ensinaram coisas que vão muito além do que suas ementas diziam. Também gostaria de agradecer à todos que participaram dessa pesquisa, sem essa contribuição não seria possível acessar esses resultados.

Resumo

Com um aumento significativo de trabalhos de pesquisa em Engenharia de Software nos últimos anos, especialmente daqueles focados no modelo Open-Source, devido à sua ascendência relativamente recente, uma questão que naturalmente surge diz respeito à sua relevância. Diante desse questionamento, esse trabalho busca investigar se a pesquisa em Engenharia de Software, focada particularmente no modelo Open-Source, produz resultados considerados relevantes na percepção dos usuários. Para investigar a relevância percebida da literatura disponível nós conduzimos dois Surveys: um na Universidade de Brasília (UnB), em que nós convidamos os estudantes a avaliar a relevância de ideias e resultados contidos em sumários construídos a partir de trabalhos de pesquisa publicados em um período de dez anos, e outro com profissionais e pesquisadores que contribuem de alguma forma com as comunidades Open-Source, onde a relevância de artigos publicados em um período de cinco anos foi avaliada com base na leitura do título e resumo originais dos trabalhos. Dessa forma, é possível apresentar um *feedback* dos estudantes, profissionais e pesquisadores, possibilitando o discernimento de questões de pesquisa que são consideradas relevantes e consequentemente passíveis de serem disseminadas dentro da comunidade Open-Source e acadêmica. Durante a investigação da relevância dos trabalhos selecionados, a abordagem proposta considerou duas questões: Uma sobre o escopo dos trabalhos identificados e outra sobre a relevância percebida desses trabalhos. Para a primeira questão, foram conduzidos dois mapeamentos sistemáticos da literatura em bases distintas, os quais revelaram um conjunto de trabalhos compostos por uma grande diversidade de resultados. Utilizando sumários elaborados a partir desses trabalhos para o primeiro Survey e os próprios resumos para o segundo, foram então aplicados os Surveys aos estudantes, profissionais e pesquisadores. Nossos achados representam um cenário muito favorável para a pesquisa voltada ao modelo open source, indicando que 77.01% dos estudantes consideram os trabalhos relevantes e que 80.56% dos pesquisadores e desenvolvedores também consideraram os trabalhos como relevantes.

Palavras-chave: Código-Aberto, Relevância de pesquisa, Estudantes de graduação, Desenvolvedores, Pesquisadores, Mapeamento sistemático, Survey

Abstract

The number of Software Engineering research papers has grown significantly over the last few years, especially those related to the open source model. Naturally, this fact raises the question of whether the research on these areas are considered to be relevant or not. This paper aims at accessing the perspective of the open source community as well as the perspective of undergraduate students regarding the relevance of the open source research. To answer about the relevance of available work, two questions were addressed: one about the scope of the studies and another about the perceived quality of these works. For the first one, two *Systematic Literature Mappings* were performed, each for a different survey to be conducted, revealing two set of works composed by a great diversity of results. Using these identified works, two different *surveys* were conducted, one with developers and researchers from several open source communities around the world and another at University of Brasília (UnB) where undergraduate students of Computer Science and related courses were invited to rate the relevance of the selected research papers. Both surveys revealed a very positive outlook on the relevance of this research area, where 77.01% of the students and 80.56% of the the open source practitioners rated the works as relevant. With these results, in addition to providing an overview of the current open source research scenario, it is also possible to give feedback from the open source community and students, providing a way to produce useful and, consequently, more disseminated works among open source practitioners.

Keywords: Open-Source Software , Research Relevance, Undergraduate Students, Practitioners, Systematic Mapping, Survey

Sumário

Lista de Figuras

Lista de Tabelas

Capítulo 1

Introdução

Com o modelo *open-source* emergindo como um fenômeno inovador em aspectos sociais e econômicos nos últimos anos [9][68][30], onde podemos ver plataformas de desenvolvimento *open-source* como o GitHub hospedando mais de 96 milhões de repositórios e mais de 200 milhões de *pull requests* até 2018 e um grande número de softwares *open-source* sendo lançados [24]. A necessidade pela pesquisa acadêmica na área de Engenharia de Software foi naturalmente aumentando.

Nesse contexto, este trabalho está interessado em, primeiramente, reunir trabalhos de pesquisa de Engenharia de Software focados no modelo *open-source* e em sua comunidade, e após identificar tais trabalhos, utilizando grupos distintos compostos por estudantes de graduação e por desenvolvedores e pesquisadores como avaliadores, medir a relevância percebida desses trabalhos para a comunidade *open-source*. Com essa abordagem, além de obter um *feedback* desses grupos sobre a relevância da literatura disponível, abrindo possibilidades para identificar a relevância e o consequente impacto de trabalhos futuros, pode-se também apresentar uma visão geral do atual cenário da pesquisa em Engenharia de software focada no modelo *open-source*.

1.1 Justificativa

A investigação sobre a percepção da relevância de uma área de pesquisa se justifica na possibilidade de fornecer um *feedback* aos pesquisadores e demais interessados a respeito da qualidade percebida pelos respectivos grupos considerados além de apresentar uma visão geral sobre a diversidade de resultados disponíveis e as possíveis lacunas existentes na literatura. Assim, pode-se esperar que tais investigações sejam recorrentes e não parece haver motivos para se acreditar que com a Engenharia de Software isso seria diferente. Existem de fato alguns trabalhos disponíveis na literatura que buscam estabelecer a relevância da Pesquisa em Engenharia de Software de forma geral[22][36][60], porém,

além dessa relevância percebida possuir um caráter mutável ao longo do tempo e entre os grupos considerados, os trabalhos identificados não poderiam ter seus resultados inferidos para a pesquisa focada em *open-source*, por exemplo, devido às especificidades do modelo *open-source*.

1.2 Objetivos

1.2.1 Objetivos Gerais

O objetivo deste trabalho é investigar qual a percepção da relevância da pesquisa em Engenharia de Software na perspectiva *open-source*, pelos profissionais atuantes nessas comunidades e por estudantes de graduação dos cursos de ciência da computação ou áreas afins.

1.2.2 Objetivos Específicos

Para atingir o objetivo geral proposto, foi preciso definir alguns objetivos mais específicos e delimitados, a saber:

1. Realização de dois mapeamentos sistemáticos da literatura em Engenharia de Software, ambos com o objetivo de identificar os trabalhos focados no modelo *open-source* capazes de representar a diversidade de resultados existentes. Cada mapeamento a ser realizado tem por objetivo a sua utilização na condução de um *survey* diferente.

2. Análise dos mapeamentos sistemáticos da literatura realizados, considerando as possíveis categorias identificadas e a abrangência e diversidade dos trabalhos resultantes.

3. Planejamento e execução de dois surveys com o objetivo de coletar a opinião de um grupo composto por alunos de graduação e de outro grupo composto por desenvolvedores e pesquisadores com relação às suas percepções sobre a relevância dos trabalhos identificados em cada mapeamento sistemático.

4. Análise dos resultados obtidos após a condução dos surveys. Uma análise para cada *survey* executado, considerando os aspectos específicos de cada um dos grupos de participantes.

1.3 Metodologia de Pesquisa

A metodologia de pesquisa adotada nesse trabalho foi constituída por dez etapas:

1. Estudo de conceitos relacionados à Engenharia de Software.
2. Estudo de conceitos relacionados ao modelo *open-source*.
3. Levantamento e discussão de trabalhos similares e relacionados.
4. Mapeamentos sistemáticos da literatura de forma a identificar trabalhos de pesquisa em Engenharia de Software focados no modelo ou nas comunidades *open-source*.
5. Análise, classificação e sintetização dentro dos conjuntos de trabalhos identificados.
6. Sumarização dos artigos encontrados no mapeamento sistemático destinado ao *survey* com alunos de graduação.
7. Elaboração de dois questionários utilizando os sumários produzidos para um *survey* com alunos de graduação e os próprios resumos dos artigos para o outro *survey* com profissionais e pesquisadores da comunidade *open-source*.
8. Condução de dois surveys, um com alunos de graduação da universidade de Brasília e outro com desenvolvedores e pesquisadores de comunidades *open-source*.
9. Análise de resultados para cada um dos surveys realizados.
10. Análise geral dos resultados, discussões e conclusões.

1.4 Organização do Trabalho

O restante desse trabalho está organizado da seguinte forma:

No Capítulo 2 é apresentado o referencial teórico necessário para entender a contextualização desse trabalho, sendo este constituído por conceitos da Engenharia de Software e do modelo *open-source*. Nesse capítulo também são apresentados e discutidos os trabalhos correlatos.

No Capítulo 4 são descritas em detalhes as abordagens utilizadas para alcançar cada um dos objetivos definidos. Ou seja, são detalhadas as etapas envolvidas na busca por trabalhos, na sumarização de resultados e no projeto e execução dos surveys. Também são apresentadas todas as análises de cada fase.

O Capítulo 5 apresenta uma breve discussão dos resultados obtidos após a execução de todas as etapas propostas junto a uma análise do que esses resultados representam. Além disso, também são apresentadas as ameaças para a validação deste trabalho.

Por fim, no Capítulo 6 temos a conclusão, composta por uma síntese da contribuição fornecida por esse trabalho bem como uma discussão sobre a implicação de tais resultados frente à proposta feita.

Capítulo 2

Referêncial Teórico

2.1 Engenharia de Software

De acordo com Sommerville[83], Engenharia de Software é uma disciplina preocupada com todos os aspectos da produção de software, desde os estágios iniciais de especificação de sistemas até a manutenção, após sua implantação. Para Pressman[75], a Engenharia de Software engloba processos, métodos e ferramentas que permitem aos profissionais construir softwares bem desenvolvidos. Sob a mesma perspectiva, de acordo com Wazlawick[97], a Engenharia de Software é considerada como o processo de estudo, criação e otimização de processos para o desenvolvimento de software, sendo as atividades fundamentais de um engenheiro de software a especificação do software, seu desenvolvimento, validação e sua evolução.

Uma das razões pelas quais a Engenharia de Software é tão importante, é a dependência, cada vez maior, da sociedade e dos indivíduos com relação aos sistemas de software, onde tais sistemas precisam ser bem projetados para se obter performance e segurança. De uma perspectiva econômica, uma outra razão é que, à longo prazo, é mais barato utilizar técnicas e métodos de Engenharia de Software do que programar sistemas sem a padronização demandada por tais técnicas e métodos, já que, para a maioria dos tipos de sistemas de software, os maiores custos estão associados às mudanças realizadas no software após sua implantação, quando este já se encontra em uso [83].

Em resumo, a Engenharia de Software é uma disciplina que envolve diversas áreas, à exemplo: processos de software, desenvolvimento ágil, engenharia de requisitos, testes de software, evolução de software e segurança. Assim, dada a importância crucial dos sistemas de software para todos os aspectos da nossa sociedade atual, a pesquisa científica relacionada a todas essas áreas tem sua importância devidamente justificada.

2.1.1 Pesquisa em Engenharia de Software

A pesquisa em Engenharia de Software sofreu, ao longo de sua existência, diversas críticas relacionadas ao rigor de seus experimentos e à validade ou uso prático de seus resultados [92], nas quais se argumentava, por exemplo, contra o tamanho dos experimentos realizados e a possibilidade de extrapolar tais resultados para o mundo real [29]. Parte do problema vem do domínio tratado, no qual experimentos em larga escala aplicados à instâncias reais de problemas são, em geral, inviáveis. Contudo, nos últimos anos, devido tanto a uma maior aderência por parte dos pesquisadores às técnicas da pesquisa empírica em Engenharia de Software, quanto ao avanço das técnicas utilizadas para a pesquisa, a pesquisa em Engenharia de Software se tornou robusta ao ponto de consolidar um corpo de conhecimento confiável, capaz de apoiar o desenvolvimento de diversos métodos e ferramentas na área de Engenharia de Software [94]. Outro fator que contribuiu para o avanço no tratamento dos problemas considerados foi uma maior abundância de dados disponíveis e a evolução das técnicas para tratar tais quantidades de dados, o que alavancou, por exemplo, as mais variadas pesquisas que lidam com mineração de repositórios de software [44].

Além dessas questões relacionadas à metodologia, existem também alegações de que supostamente grande parte dos experimentos realizados seriam focados em temas de pouca preocupação para os profissionais da área, ou mesmo de que os pesquisadores seriam muitas vezes incapazes de compreender completamente os resultados das aplicações de suas soluções em ambientes reais [29]. Os cenários descritos seriam instâncias causadas pela problemática lacuna entre pesquisa e prática, em que grande parte dos resultados de pesquisas em Engenharia de Software não alcançam os profissionais da indústria.

2.2 Open Source

O termo Open Source Software (OSS) foi introduzido por Stallman em 1998 e a ideia por trás do conceito é bem simples, o OSS é desenvolvido por indivíduos ou companhias que lançam produtos sob licenças de código aberto. O código do software lançado sob essa licença pode ser considerado como publicamente disponível, de modo que se pode usar livremente o código, reutilizá-lo através de incorporações em módulos e até mesmo modificá-lo.

2.2.1 Características

De acordo com a Iniciativa Open Source, o *open-source* é definido como um modelo de desenvolvimento que permite a livre distribuição e o completo acesso ao código fonte,

no qual o software pode: **a)** ser executado para qualquer finalidade; **b)** ser estudado de acordo com seu funcionamento, o qual pode ser modificado conforme a necessidade; **c)** ter cópias distribuídas sem custos; e **d)** passar por melhorias, as quais também podem ser lançadas sem a necessidade de taxas ou *royalties*.

Além disso, os termos de distribuição de um OSS devem estar de acordo com os seguintes critérios [2]:

1. Livre distribuição, sendo permitida a venda, sem taxas, do software como componente de um outro produto
2. Disponibilidade do código fonte e de sua forma compilada
3. Permissão para modificações e derivação de novos trabalhos, os quais podem ser distribuídos sob a mesma licença que o original
4. Integridade do código fonte do autor, aonde a licença pode exigir que as modificações feitas sejam distribuídas somente em forma de *patch files*
5. Nenhuma discriminação contra pessoas ou grupos
6. Nenhuma discriminação contra campos de atuação
7. Distribuição da licença, sem a necessidade de licenças adicionais para redistribuições do programa
8. A licença não deve ser específica a um produto
9. A licença não deve restringir outros softwares
10. A licença deve ser tecnologicamente neutra

2.2.2 Considerações

O acesso irrestrito ao código desses softwares permite a inspeção e verificação do software à nível de código fonte, o que intuitivamente, somado ao livre acesso por qualquer um, é um fator que alavanca muito a confiabilidade do software [69]. Como destacado por Pandey and Tiwari[69], Eric Raymond - antigo presidente do grupo de interesse da Iniciativa Open Source - é um dos defensores desse ponto de vista, e afirmou em seu livro que, graças ao mecanismo de revisão por pares implementado no modelo, um software de código aberto maduro é o mais confiável que um software pode se tornar.

Um outro aspecto que surge como consequência do modelo *open-source* é a ocorrência de uma colaboração geograficamente distribuída, a qual se dá através de tecnologias baseadas na web, como o GitHub. Um ponto negativo dessa distribuição geográfica são os

Características	Desenvolvimento Open Source	Desenvolvimento Tradicional
Disponibilidade do código fonte	Livremente disponibilizado	Não disponibilizado
Modificações	Alterações no código fonte	Apenas ajustes e customizações
Debugging	Feito pela comunidade inteira	Recursos limitados para tarefas de debugging
Estrutura das equipes	Hierárquica, porém menos restrita	Ciclo de desenvolvimento padrão
Tamanho das equipes	Grupos grandes de participantes livres; Alta participação dos usuários	Pequenos times de desenvolvimento; Desenvolvedor pago pelo fornecedor
Fornecedor	Uma comunidade	Uma companhia

Tabela 2.1: Diferenças entre modelo *open-source* e proprietário

desafios para se estabelecer uma comunicação eficiente [53]. Além disso, devido ao fato de a colaboração se dar muitas vezes de forma voluntária, os projetos de OSS acabam sofrendo do risco de imprevisibilidade, já que muitos desses colaboradores podem contribuir de forma inconsistente ou mesmo abandonar os projetos repentinamente [50] [12]. Por outro lado, o modelo pode acabar por atrair desenvolvedores motivados e altamente capacitados, os quais não fazem parte da cultura corporativa e os quais podem colaborar com múltiplos projetos distintos ao mesmo tempo [76]. Outro ponto favorável é a ausência de restrição quanto ao número máximo de desenvolvedores em um projeto, algo que, devido aos recursos serem limitados, não se vê no desenvolvimento proprietário tradicional.

2.2.3 Código proprietário

Como visto, o modelo *open-source* contrasta fortemente com o desenvolvimento tradicional de código proprietário, no qual a intenção é justamente prevenir que terceiros acessem o código fonte dos projetos desenvolvidos. O código proprietário é tipicamente entregue em forma de executável apenas, no qual modificações ao produto disponibilizado se restringem à customização de configurações do próprio executável. Outro aspecto a ser considerado é que o número de desenvolvedores voltados ao processo de *debugging* nesses tipos de projetos é limitado e em geral a um pequeno número de pessoas, diferente dos softwares abertos, aonde a comunidade inteira pode participar ativamente no processo [69]. Com relação à estrutura de organização em modelos de desenvolvimento Open Source, ambas são similarmente estruturadas no sentido hierárquico, com o diferencial de que nos projetos de OSS essas hierarquias são menos rígidas [25]. No caso do OSS, geralmente ou as decisões são tomadas com base na centralização de uma única pessoa (como no caso do Kernel Linux), ou em diversos desenvolvedores, os quais formam algo como um time núcleo [47]. A tabela 2.1 resume algumas diferenças entre o modelo de desenvolvimento de código aberto com relação aos modelos tradicionais prorietários.

2.2.4 Licenças populares

Com relação as licenças, citamos algumas licenças populares, aprovadas pela Iniciativa Open Source [1]:

- Apache License 2.0
- BSD 3-Clause "New" or "Revised" license
- BSD 2-Clause "Simplified" or "FreeBSD" license
- GNU General Public License (GPL)
- GNU Library or "Lesser" General Public License (LGPL)
- MIT license
- Mozilla Public License 2.0
- Common Development and Distribution License
- Eclipse Public License version 2.0

2.2.5 Comunidades Open Source

A comunidade *open-source* compreende todos os desenvolvedores envolvidos com o projeto. Nesse sentido, as comunidades desempenham um papel crucial para o sucesso de um projeto Open Source.

A saúde do ecossistema de software refere-se ao funcionamento normal de seus projetos constituintes sendo que os problemas relacionados à saúde tendem a se propagar por todo o ecossistema por meio de dependências técnicas ou sociais [63]. Marsan et al.[63] argumentam que a baixa qualidade do código é o problema técnico mais importante em projetos Open Source, enquanto que a perda de colaboradores é o problema social mais importante.

Steinmacher et al.[86] relataram em sua pesquisa que um fluxo contínuo de recém-chegados é fundamental para a sobrevivência de projetos Open Source. No entanto, esses recém-chegados são suscetíveis a várias barreiras sociais, como problemas de comunicação e recepção. Além disso, de acordo com Carillo et al.[20] uma comunidade que atrai um grande número de novos colaboradores dos quais a maioria não realiza com êxito ou satisfatoriamente as tarefas do projeto pode não prosperar a longo prazo e pode até mesmo desaparecer. Assim, o bem-estar de um projeto *open-source* depende também da transformação de novatos em bons colaboradores que contribuem de maneira substancial para o projeto. Os resultados demonstram que a identidade e integração social são fatores de forte influência na previsão do desempenho de colaboradores [20].

Capítulo 3

Trabalhos correlatos

Interessados em investigar a relevância da pesquisa em Engenharia de Requisitos para os profissionais da indústria, Franch et al.[37] produziram uma replicação do trabalho de Lo et al.[61]. O estudo se deu através de uma colaboração internacional e resultou em um projeto de pesquisa empírico chamado 'RE-Pract', o qual contou com o planejamento de um *survey* a ser enviado para centenas de profissionais da indústria de várias companhias ao redor do mundo e que ainda estava em andamento no ato de publicação. O *survey* tem por objetivo questionar os profissionais sobre a relevância percebida a partir de um conjunto de 418 artigos de pesquisa em Engenharia de Requisitos publicados em diversas conferências entre o período de 2010 a 2015. Além da percepção da relevância do conjunto de trabalhos para a indústria, os autores se preocuparam com quais ideias de pesquisa seriam melhor avaliadas, com as questões de pesquisa que causariam maior interesse por parte dos profissionais da indústria, com a relação entre a relevância atribuída para o estudo e a sua associação com a indústria e também com a diferença na percepção da relevância com base nos cargos dos participantes. Até o momento de publicação, os pesquisadores haviam acabado a etapa de seleção e sumarização de artigos, aonde reportaram que a maioria dos trabalhos (52%) foram extraídos da conferência ESEM (Empirical Software Engineering and Measurement), provavelmente devido ao tópico da conferência ser justamente o de interesse da pesquisa.

Diante da vaga definição comumente associada à proficiência de desenvolvimento de código (*Coding Proficiency*) em projetos de software, Xia et al.[99] se dedicaram a investigar a seguinte questão de pesquisa: Quais são as habilidades importantes para a melhoria na proficiência de codificação de um desenvolvedor? Em um esforço para responder a essa questão, os autores realizaram um estudo empírico composto por um *survey* que foi conduzido com 340 desenvolvedores de software oriundos de 33 países diferentes. Para a elaboração do survey, os autores identificaram um conjunto composto por 38 proficiências de desenvolvimento de código levantadas em uma série de entrevistas com 15

desenvolvedores de 3 companhias diferentes, as quais foram agrupadas em 9 categorias de habilidades. Com exceção de uma (Domínio de linguagens de programação para sistemas legados como Cobol), todas as habilidades receberam uma avaliação média acima de neutra, sendo que a capacidade de subdividir tarefas e a familiaridade com sistemas de controle de versão (como o Git) foram as melhor avaliadas. Após a condução do survey, os pesquisadores destacaram 21 habilidades avaliadas como importantes pelos participantes junto de justificativas fornecidas com essas avaliações.

Em uma proposta semelhante ao trabalho anterior, Meyer et al.[65] procuraram determinar quais as percepções dos desenvolvedores a respeito dos fatores envolvidos na produtividade do desenvolvimento de software. Para tal, os autores seguiram uma abordagem composta de duas fases. Na primeira fase, conduziram um *survey* que contou com 379 profissionais de desenvolvimento de software, o qual se justificou pelo interesse em determinar o significado de produtividade para tais profissionais. A segunda fase foi composta por um estudo observacional feito com 11 desenvolvedores, que investigou em mais detalhes os aspectos relacionados à produtividade definidos na primeira fase. Os resultados da primeira fase mostraram que, enquanto os desenvolvedores consideram a programação como altamente produtiva, a maioria enfrentou dificuldades em julgar a produtividade de diversas atividades como reuniões ou o gerenciamento de tarefas e e-mails. Outro aspecto levantado pelos participantes foi o suposto impacto negativo causado pelas mudanças de contexto na realização das atividades profissionais. Curiosamente, a segunda fase evidenciou que os desenvolvedores experienciam uma grande quantidade de mudanças de contexto, alternando entre tarefas a cada 6.2 minutos e entre atividades a cada 1.6 minutos. Com os resultados das duas fases, os desenvolvedores propuseram várias maneiras de melhorar a produtividade dos desenvolvedores através de ferramentas e boas práticas.

Interessados na percepção de fatores que influenciam na produtividade de desenvolvimento a partir da perspectiva de gerentes de projeto de software, Oliveira et al.[27] conduziram um estudo qualitativo com base em dados coletados a partir de entrevistas semi-estruturadas. O estudo buscou responder como os gerentes de projeto definem a produtividade dos desenvolvedores e quais os mecanismos que eles usam para medir tal produtividade. As entrevistas foram feitas com 12 gerentes de projeto oriundos de 3 organizações de desenvolvimento de software diferentes do Brasil e revelou quatro fatores que parecem influenciar na percepção dos gerentes sobre produtividade dos desenvolvedores: 1. Entregas pontuais; 2. Construção de artefatos que não exigem retrabalho; 3. Concretização das expectativas dos *Stakeholders*; e 4. Características pessoais como por exemplo proatividade. Os autores concluem apontando para a diferença entre a percepção de produtividade dos gerentes de projeto com relação a percepção dos desenvolvedores

levantada em outros trabalhos da literatura.

Devanbu et al.[31] procuraram descobrir quais são as crenças preconcebidas que os desenvolvedores possuem sobre alguns tópicos da Engenharia de Software e a relação dessas crenças com dados empíricos coletados em projetos nos quais esses desenvolvedores trabalharam. Na busca por essas respostas, foi desenvolvido um questionário usando a escala *likert* no qual a origem das crenças foi solicitada quando a resposta dada era significativa (*Concordo fortemente* ou *Discordo fortemente*). Ao final, os resultados mostraram que de fato os desenvolvedores possuem algumas fortes crenças preconcebidas. E na análise dos dados coletados pelos questionários, também foi observado que: **1.** As opiniões mais fortes dos desenvolvedores são baseadas principalmente em sua própria experiência; **2.** Os jornais comerciais tiveram mais influência nas opiniões dos desenvolvedores do que as pesquisas científicas.

Em uma tentativa de melhorar o trabalho de Lo et al.[61], Carver et al.[23] investigaram a relevância de alguns trabalhos publicados entre 2011 e 2015 sobre Engenharia de Software Empírica a partir da perspectiva dos desenvolvedores. A abordagem adotada foi a realização de um *survey* que foi enviado a centenas de desenvolvedores em várias empresas ao redor do mundo. Os autores concluíram que os desenvolvedores consideram os estudos relevantes (67%).

Em um esforço para descobrir quais perguntas os desenvolvedores de software gostariam que os pesquisadores investigassem, Begel and Zimmermann[14] realizaram uma pesquisa em duas etapas. Eles primeiro questionaram um grupo de desenvolvedores de uma empresa sobre os tópicos que eles gostariam que fossem investigados pela comunidade científica. Depois, eles questionaram outro grupo da mesma empresa sobre a relevância dessas questões levantadas. Os resultados deste trabalho levantaram várias questões que os desenvolvedores consideram relevantes, as quais foram classificadas em diversas categorias; as categorias mais abrangentes foram: **a)** Práticas de desenvolvimento; **b)** práticas de teste; e **c)** Equipes e colaboração.

Capítulo 4

Mapeamento sistemático

Para determinar a relevância percebida dos trabalhos de pesquisa em Engenharia de Software focados no modelo open-source, primeiro é necessário obter um conjunto de trabalhos que servirá de base para a avaliação pelo grupo alvo, devendo esse conjunto de trabalhos ser capaz de expressar a variedade de resultados disponíveis. Conforme elucidado por Grant and Booth[43], o mapeamento sistemático é uma forma de revisão da literatura realizada com a finalidade de mapear e classificar a literatura existente em um determinado tópico. Assim, um dos principais objetivos desse tipo de pesquisa é obter uma visão geral de uma área de pesquisa com relação aos tipos de resultados disponíveis [71]. Por essa razão, adotamos o mapeamento sistemático para realizar a coleta dos trabalhos voltados para a pesquisa do modelo open-source.

Neste trabalho, optamos pela condução de dois mapeamentos sistemáticos diferentes. Aonde cada mapeamento tem por objetivo, além de servir de insumo para um grupo distinto de respondentes do survey, expressar a diversidade dos resultados disponíveis que discutem o modelo open-source. Assim, os mapeamentos em questão foram conduzidos considerando bases de dados e até mesmo períodos de publicação distintos. Essa decisão se justifica em uma tentativa de maximizar a diversidade de resultados identificados nesse trabalho, já que a quantidade de artigos utilizados em cada *survey* deve ser razoável para que os respondentes participem e tenham uma compreensão das pesquisas realizadas nesta área.

Os trabalhos encontrados com a realização dos mapeamentos sistemáticos foram utilizados nos surveys, a fim de se identificar a relevância percebida destes trabalhos pelos estudantes e profissionais. Para o primeiro survey, realizamos uma sumarização dos trabalhos identificados com o objetivo de facilitar a análise por parte dos estudantes. No segundo *survey* em que o público alvo eram pesquisadores e profissionais que atuam nas comunidades open-source, optamos pela utilização dos próprios títulos e resumos dos artigos identificados.

4.1 Primeiro Mapeamento Sistemático da Literatura

A Figura 4.1 apresenta o mapeamento sistemático que foi conduzido neste trabalho, de acordo com os seguintes estágios: Definição da questão de pesquisa; Condução da pesquisa; Triagem de artigos; Extração de palavras chave a partir dos resultados; e extração de resultados.

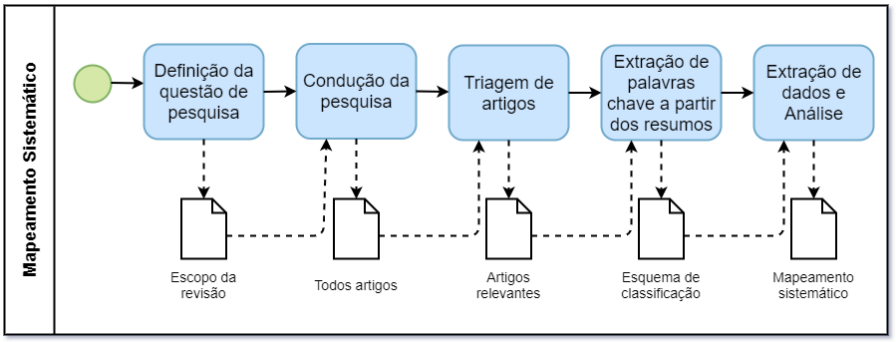

Figura 4.1: Estágios do mapeamento sistemático

A ferramenta Parsifal foi utilizada nas fases de planejamento e condução do mapeamento sistemático de literatura por prover um ambiente de suporte para essas fases, no qual uma das vantagens com relação às demais ferramentas disponíveis é permitir a revisão da literatura de forma colaborativa e na nuvem.

4.1.1 Questão de Pesquisa

O mapeamento foi projetado com o objetivo de identificar trabalhos de pesquisa em Engenharia de Software focados no modelo *open-source* que sejam capazes de representar outros resultados similares presentes na literatura. Dessa forma, espera-se que o conjunto de trabalhos resultantes possa oferecer uma boa perspectiva da diversidade de resultados de pesquisa disponíveis, bem como das possíveis lacunas existentes. Para atingir o objetivo primário desse estudo, foram definidas as seguintes questões de pesquisa:

RQ.1: Existem trabalhos na literatura que podem ser utilizados para representar o cenário atual da pesquisa em Engenharia de Software focada em open-source?

RQ.2: Qual é a relevância da literatura em *open-source* a partir da perspectiva de estudantes de graduação?

RQ.3: Qual é a relevância da literatura em *open-source* a partir da perspectiva dos profissionais que atuam nessas comunidades?

4.1.2 Estratégia de Busca

Para realizar o mapeamento sistemático, foram considerados os trabalhos publicados em um período de dez anos, a fim de maximizar a captura de resultados encontrados na literatura. A estratégia de busca envolveu a busca automática e a busca manual. A busca automática foi realizada em três bases digitais, a saber:

- ACM Digital Library;
- IEEE Xplore Digital library;
- Scopus.

A busca manual foi realizada nos anais de conferências e periódicos na área de engenharia de software. A Tabela 4.1 apresenta os veículos consultados.

Nome do Veículo	**Endereço virtual**
IEEE Transactions on Software Engineering	https://computer.org/csdl/journal/ts
Empirical Software Engineering	https://springer.com/journal/10664
ESEM - International Symposium on Empirical Software Engineering and Measurement	http://esem-conferences.org/
ICSE - International Conference on Software Engineering	http://icse-conferences.org/

Tabela 4.1: Conferências e periódicos utilizados na busca manual

String de busca

Para realizar a busca dos trabalhos, definimos a seguinte string de busca:
("Open Source"OR "Free Software"OR "OS"OR "OSS"OR "FOSS") AND ("Communities"OR "Community"OR "Contributors"OR "Project"OR "Software")

4.1.3 Critérios de Inclusão e Exclusão

Para realizar a seleção dos trabalhos identificados durante o mapeamento sistemático, os seguintes critérios de inclusão e exclusão foram definidos:

Critérios de Inclusão

1. Trabalhos que representam melhorias para as técnicas adotadas nos projetos open-source.
2. Trabalhos que analisam aspectos relacionados ao desenvolvimento de projetos open-source.
3. Trabalhos que analisam as vantagens e desvantagens da adoção do modelo open-source.
4. Trabalhos que analisam os aspectos envolvidos no ensino e aprendizagem do modelo *open-source* ou na introdução de novos participantes ao modelo.

Critérios de Exclusão

1. Trabalhos que apresentam resultados similares àqueles já identificados
2. O não cumprimento dos critérios de inclusão
3. Trabalhos que não pertencem ao escopo desse trabalho
4. Trabalhos publicados como *Short Paper*.

Cabe ressaltar que os critérios de inclusão foram cuidadosamente projetados para se alcançar o efeito de classificação de trabalhos similares em categorias distintas, onde a similaridade se restringiu à categoria de classificação definida.

4.1.4 Seleção de Artigos

O processo de avaliação da qualidade dos artigos identificados após a execução da estratégia de busca foi projetado para que apenas os artigos mais representativos chegassem a compor cada uma das categorias identificadas.

O processo completo de busca e seleção, considerando os resultados obtidos após a execução de cada etapa, é apresentado na Figura 4.2, o qual foi realizado da seguinte forma:

1. Aplicação da *string* de busca nas bases digitais selecionadas;
2. Identificação de estudos potencialmente relevantes com base na leitura das palavras-chave, do título e, quando necessário, do resumo, gerando uma lista preliminar de estudos, os quais resultaram em um total de 600 artigos. Nessa etapa também foram descartados os trabalhos duplicados;

3. Leitura da introdução, metodologia e conclusão dos artigos pré-selecionados, aplicando os critérios de inclusão e exclusão. Esse estágio resultou em um total de 93 artigos;

4. Leitura completa dos artigos selecionados no estágio anterior. No total foram classificados 38 artigos para responder a questão de pesquisa.

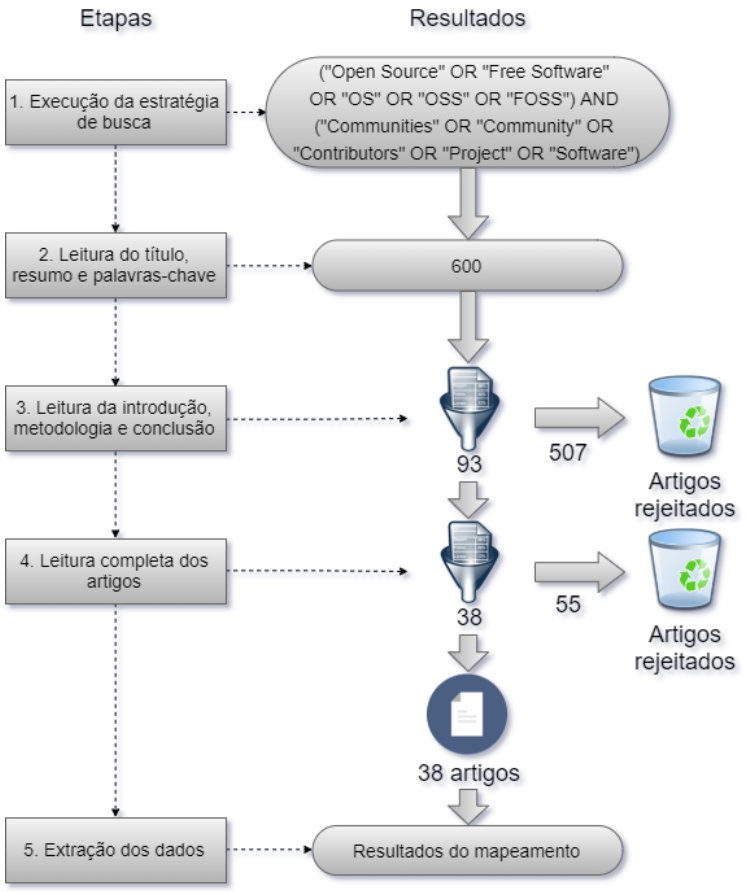

Figura 4.2: Etapas do mapeamento sistemático

Durante a extração e classificação dos resultados, realizamos a leitura completa dos 38 artigos primários selecionados, a partir da qual um processo de revisão em pares foi adotado para a avaliação, sendo que os casos de desentendimento foram resolvidos por um terceiro revisor. Além disso, foi realizada uma extração piloto dos dados, a fim de alinhar o entendimento dos revisores com relação aos critérios definidos para a seleção. A extração piloto dos dados foi realizada com cinco artigos primários aleatórios, com os quais os revisores discutiram sobre os desentendimentos com relação às respostas individuais de cada um.

O procedimento utilizada para lidar com uma quantidade manuseável de trabalhos foi incorporada em um dos critérios de exclusão, a saber: "Trabalhos que apresentam

resultados similares àqueles já identificados". Essa decisão não afetou a qualidade do mapeamento sistemático, já que o objetivo era focar justamente na diversidade de resultados. Além disso, o projeto do *survey* considerou a utilização de sumários, propondo a captura dos resultados em um nível relativamente alto de abstração, no qual, idealmente, teríamos resultados muito similares com sumários praticamente indistinguíveis.

4.1.5 Resultados do Mapeamento

Essa seção apresenta os resultados obtidos com a execução do mapeamento sistemático, junto a uma análise relacionada à questão de pesquisa. Ao final, são discutidos alguns dos resultados identificados.

A primeira iteração no processo de seleção gerou um total de 600 trabalhos. A fim de conduzir as demais fases do mapeamento, os trabalhos foram exportados para a plataforma *parsifal*. Na plataforma, as etapas 3 e 4 da estratégia de seleção foram realizadas, resultando em respectivamente, 93 e 38 artigos. A distribuição dos trabalhos identificados em cada etapa de seleção é apresentada na Figura 4.3.

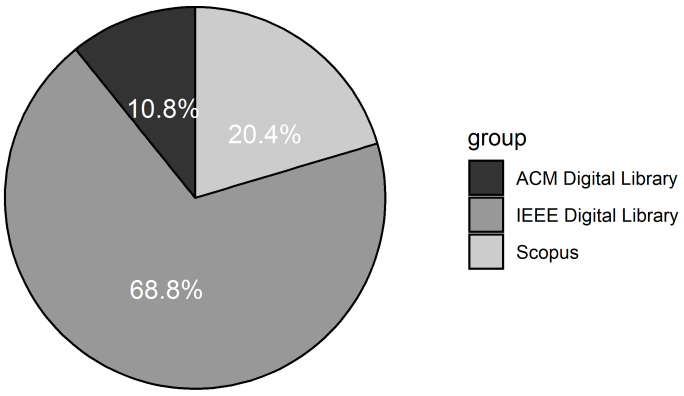

Figura 4.3: Distribuição dos trabalhos por base digital

Os 38 estudos primários foram analisados e classificados de acordo com as categorias definidas através dos critérios de inclusão dos trabalhos selecionados, a saber:

1. *Practical Improvements* – Trabalhos que representam melhorias para as técnicas adotadas nos projetos open-source;

2. *Aspect Analysis* – Trabalhos que analisam aspectos relacionados ao desenvolvimento de projetos open-source;

3. *Advantages/Disadvantages of open-source* – Trabalhos que analisam as vantagens e desvantagens da adoção do modelo open-source;

4. *open-source Teaching* – Trabalhos que analisam os aspectos envolvidos no ensino e aprendizagem do modelo open-source;

A Tabela 4.2 apresenta os estudos primários identificados no mapeamento sistemático, bem como a sua respectiva categorização. As quatro categorias tiveram uma quantidade similar de trabalhos, variando entre 8 e 12 (figura 4.4).

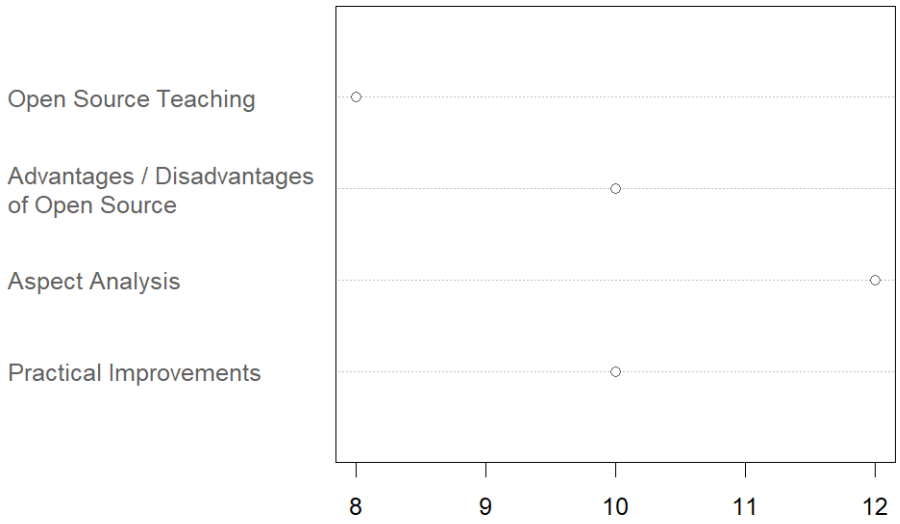

Figura 4.4: Quantidade de artigos por categoria

Aspect Analysis

Alguns estudos primários fornecem formas de medir diferentes aspectos diretamente relacionados ao desenvolvimento do modelo open-source, tais como: reuso, evolução, saúde da comunidade, etc.

Fielding[35] (Id E5) assumiram uma perspectiva focada na arquitetura do produto, apresentando fatores que podem influenciar no sucesso de projetos open-source. O trabalho explora as características de projetos grandes e bem sucedidos de permitir a colaboração distribuída por meio de extensões e, ao mesmo tempo, preservar o controle centralizado.

Resultados igualmente interessantes são apresentados por Wang et al.[95] (Id E1), no qual são discutidos os benefícios do *micro-blogging*, mostrando que nos casos considerados houveram grandes avanços relacionados à produtividade geral dos projetos.

Com um foco menos tradicional, Fan et al.[34] (Id E6) apresentam uma forma de comparar a qualidade de projetos *open-source* com base no *feedback* de usuários do código reutilizado, e apesar da abordagem orientada à *feedback* ter um atraso com relação à

Tabela 4.2: Artigos selecionados no primeiro mapeamento sistemático

Id	Título	Ref.
	Aspect analysis	
E1	Microblogging in Open Source Software Development: The Case of Drupal and Twitter	[95]
E2	What Makes an Open Source Code Popular on Git Hub?	[98]
E3	We Don't Need Another Hero?: The Impact of "Heroes"on Software Development	[3]
E4	Evaluating the Effects of Architectural Documentation: A Case Study of a Large Scale Open Source Project	[54]
E5	Software architecture in an open source world	[35]
E6	Ranking open source software based on crowd wisdom	[34]
E7	Measuring the Evolution of Open Source Software Systems with Their Communities	[96]
E8	How does contributors involvement influence open source systems	[7]
E9	A Case Study: Open Source Community and the Commercial Enterprise	[40]
E10	A survey on Open Source Software Trustworthiness	[16]
E11	A systematic review of research on open source software in commercial software product development	[49]
E12	Poster: Understanding newcomers success in open source community	[13]
	Practical improvements	
E13	Towards Promoting Design and UML Modeling Practices in the Open Source Community	[6]
E14	Improvement on ABDOM-Qd and Its Application in Open-Source Community Software Defect Discovery Process	[46]
E15	Software process: the key to developing robust, reusable and maintainable open-source software	[80]
E16	Semantic tools for improving software development in open source communities	[58]
E17	Evaluating Quality of Open Source Components for Reuse-Intensive Commercial Solutions	[78]
E18	A reference model for evaluating performance of open source community	[8]
E19	Estimating Development Effort in Free/Open Source Software Projects by Mining Software Repositories: A Case Study of OpenStack	[77]
E20	Reliability Growth of Open Source Software Using Defect Analysis	[89]
	Advantages/Disadvantages of Open Source	
E21	Toward an Understanding of the Motivation Open Source Software Developers	[100]
E22	The organization of open source communities: Towards a framework to analyze the relationship between openness and reliability	[52]
E23	Comparison of big data analyses for reliable open source software	[91]
E24	The QualOSS open source assessment model measuring the performance of open source communities	[84]
E25	On the Impact of Being Open	[81]
E26	Inner Source–Adopting Open Source Development Practices in Organizations: A Tutorial	[88]
E27	Open Source Software: Is It Worth Converting?	[57]
E28	Open Source Drives Innovation	[33]
	Open Source teaching	
E29	Using open source projects in higher education: A two-way certification framework	[70]
E30	What can software engineering students learn from studying open source software?	[21]
E31	Training Software Engineers Using Open-Source Software: The Professors' Perspective	[73]
E32	Integrating Open Source Software into software engineering curriculum: Challenges in selecting projects	[41]
E33	Teaching Evolution of Open-Source Projects in Software Engineering Courses	[18]
E34	Experiences with Open Source Software Engineering Tools	[93]
E35	Enriching software engineering courses with service-learning projects and the open source approach	[59]
E36	Software engineering education in the era of outsourcing, distributed development, and open source software: challenges and opportunities	[45]
E37	Utilizing open source software in teaching practice-based software engineering courses	[32]
E38	How Successful Open Source Projects Work, and How and Why to Introduce Students to the Open Source World	[28]

avaliação de projetos, o método apresentou resultados satisfatórios após a comparação dos resultados com alguns *blogs* profissionais e comunidades de produção de software.

Já o trabalho de Gary et al.[40] (Id E9), analisa as vantagens e desvantagens do modelo no desenvolvimento de um pequeno software. Os autores reportaram diversas vantagens, mas também houveram desvantagens consideráveis (e.g. ciclos de lançamento instáveis). No final, o resultado da matriz de avaliação de fatores externos revelou que a companhia, até o momento do estudo, estava indo bem na indústria de desenvolvimento *open-source* para educação superior.

O trabalho de Bayati[13] (Id E12), por sua vez, busca investigar os fatores que influenciam o sucesso de novos desenvolvedores que acabaram de ingressar em projetos open-source.

O trabalho de Agrawal et al.[3] (Id E3), demonstra que a ocorrência de desenvolvedores heroicos (que produzem a maior parte do código sozinhos), afeta de forma desprezável a taxa com que os problemas e falhas são tratadas ou o tempo necessário para concluir melhorias no software.

Por fim, Bianco et al.[16] (Id E10) conduziram um *survey* a fim de investigar fatores que influenciam na confiança da adoção de produtos e componentes open-source. Os participantes pertenciam a três diferentes cargos: gerente sênior; gerente de projetos; e desenvolvedor. Alguns dos fatores que foram selecionados pelos autores como mais relevantes na hora da decisão foram: confiabilidade; grau com que o produto atende aos requisitos funcionais; disponibilidade de documentação; e manutenção.

Practical Improvements

Alguns artigos forneceram resultados que implicam diretamente em melhorias significativas para as práticas envolvidas no desenvolvimento orientado ao modelo open-source. Esses trabalhos forneceram resultados que variam desde a introdução de métodos que facilitam a detecção de erros em projetos *open-source* [46] (Id E14) até a apresentação de um processo de desenvolvimento de software que facilita o desenvolvimento multiplataforma em projetos *open-source* [80] (Id E15).

Quando se trata de reuso, ocorrem diversas preocupações com relação à decisão de qual componentes será utilizado. Nesse sentido, o trabalho de Rudzki et al.[78] (Id E17) apresenta um mecanismo para avaliar a qualidade de componentes de software open-source. O mecanismo apresenta critérios úteis para decidir sobre o quão apto um código está para ser reutilizado. A lista de critérios é extensiva e algumas das categorias identificadas foram: *Código fonte*, *Documentação e suporte* , *Comunidade* , *Popularidade* e *Maturidade*.

O trabalho de An et al.[8] (Id E18) se baseia na análise dos componentes de uma comunidade *open-source* e suas abordagens para o compartilhamento de conhecimento e

se propõe a criar um modelo que permita a avaliação da performance de uma comunidade. O trabalho também apresenta uma fórmula que mede a taxa de troca de conhecimento em projetos de software open-source.

Além disso, identificamos o trabalho de Robles et al.[77] (Id E19), o qual apresenta uma nova abordagem para estimação de esforço. Essa abordagem se baseia na observação de um trabalho de implementação real e considera dados de repositórios gerenciadores de código. O esforço estimado levou em consideração aqueles indivíduos que não dedicaram seus dias inteiros de trabalho para os projetos e também aqueles desenvolvedores (contratados ou não) que trabalharam por tempo integral nos projetos.

Por fim, Syed Mohamad and McBride[89] (Id E20) examinaram a diferença entre as taxas de ocorrência de erros em dois projetos *open-source* populares que se encontravam ativos na época com relação a alguns projetos proprietários, fornecendo uma forma de investigar a confiabilidade de projetos *open-source* com relação à ocorrência de erros.

Advantages/Disadvantages of Open Source

Com relação à adoção do modelo open-source, encontramos diversos estudos primários com diferentes resultados evidenciando os aspectos positivos e negativos, os resultados variam desde estudos de caso até a apresentação de diretrizes para facilitar uma transição para o modelo.

Assumindo uma perspectiva abrangente, o trabalho de Ebert[33] (Id E28) fornece diretrizes gerais para alavancar os pontos fortes do modelo *open-source* sobre qualquer projeto. Já o trabalho de Joode and Bruijne[52] (Id E22) se preocupa com a confiabilidade dos projetos. Os autores discutem três hipóteses formuladas, as quais relacionam aspectos de arquitetura diretamente com a confiabilidade do software open-source. Os aspectos de arquitetura considerados foram: a porcentagem de desenvolvedores que também são usuários do software; a transparência no fluxo de comunicação; e a popularidade do projeto.

Alguns estudos apresentam uma análise de aspectos relacionados à comunidade open-source, como o de Soto and Ciolkowski[84] (Id E24) o qual analisou as métricas do modelo *QualOSS*, focando particularmente nas métricas que se referem à análise do desempenho de comunidades de software open-source. As principais questões consideradas por essas métricas são a probabilidade de uma comunidade *open-source* sobreviver a longo prazo (sustentabilidade) e a habilidade de produzir software com alta qualidade de forma consistente (maturidade). Também temos o trabalho de Ye and Kishida[100] (Id E21), no qual apresentaram formas de se criar comunidades *open-source* sustentáveis, dando atenção para a importância de uma comunidade dinâmica e auto reprodutiva e o impacto positivo dessa sobre os projetos.

Com um outro foco, ao contrastar diferentes movimentos open-source, Schuwer et al.[81] (Id E25) analisaram os impactos do modelo *open-source* em um contexto altamente amplo do desenvolvimento de software.

Por fim, preocupado com a perspectiva corporativa, Stol and Fitzgerald[88] (Id E26) apresentaram os fatores que devem ser levados em consideração ao se adotar o modelo *open-source* nas organizações que seguem modelos de código proprietário. Seguindo uma similar linha de pesquisa, através de um estudo de caso, Laplante et al.[57] (Id E27) analisaram quais são os cenários em que vale a pena realizar a mudança para o modelo open-source, sugerindo também que tal mudança não precisa ser devastadora.

Open Source Teaching

Um dos grandes problemas relacionados ao modelo *open-source* é a dificuldade encontrada por desenvolvedores quando são confrontados pela primeira vez com o modelo. Como forma de lidar com esse problema, diversos estudos se preocupam com a investigação de medidas e abordagens que introduzam o ensino do modelo *open-source* nas instituições de ensino ou que facilitem a chegada de novos membros às comunidades open-source.

Um trabalho que tenta motivar a implantação do ensino nos cursos universitários é o de Papadopoulos et al.[70] (Id E29), o qual argumenta que o envolvimento de estudantes com projetos *open-source* reais e com suas comunidades é um método efetivo de se ensinar o desenvolvimento de habilidades, já que tal envolvimento resulta nos participantes obtendo familiaridade com as ferramentas e tecnologias usadas no desenvolvimento de software open-source. Uma das propostas de seu trabalho é a emissão de certificados para os estudantes que se envolvessem nesses projetos, em uma tentativa de formalizar os resultados dos participantes. Com interesse similar, Toth[93] (Id E34) apresenta uma forma de melhorar o foco prático dos cursos de Ciência da Computação e Engenharia de Software através da experiência com o desenvolvimento de projetos open-source.

O trabalho de Liu[59] (Id E35) propõe um processo de software educacional que facilita a adoção de projetos reais e complexos nos cursos de Engenharia de Software, contornando os problemas resultantes de um ensino que usa como base apenas projetos simples, cujos desafios são muito distantes daqueles apresentados em projetos reais.

Finalmente, apresentamos dois trabalhos que lidam com a perspectiva dos desenvolvedores não familiarizados com o modelo open-source. Hawthorne and Perry[45] (Id E36) lidam com as dificuldades que os engenheiros de software encontram ao trabalhar com as práticas de desenvolvimento de software distribuídos.

4.2 Survey realizado com estudantes de graduação

Para investigar a relevância da pesquisa em Engenharia de Software focada no modelo *open-source* utilizando estudantes universitários como avaliadores, o conjunto de trabalhos identificados no primeiro mapeamento sistemático, apresentado na Seção 4.1 foi utilizado como base para a realização de um survey, o qual se justifica como uma forma de se obter a opinião direta dos estudantes sobre o cenário investigado. Dessa forma, a questão de pesquisa que guiou o presente *survey* foi a RQ.2.

Para o desenvolvimento do survey, foi utilizado o *framework* proposto por Lo et al.[62], o qual foi realizado nas seguintes etapas: *Seleção e resumo dos artigos de interesse*; *Seleção dos participantes*; *Submissão do survey e coleta de dados*;

4.2.1 Seleção e resumo dos artigos de interesse

A seleção dos estudos primários ocorreu durante a execução do primeiro mapeamento sistemático (Seção 4.1), portanto, nessa fase fizemos apenas a sumarização dos estudos identificados. Para o processo de sumarização, realizamos uma síntese dos resultados de cada artigo para que cada participante pudesse se concentrar apenas na ideia principal apresentada em cada trabalho.

Dessa forma, durante a elaboração dos resumos, utilizamos o abstract e, quando a contribuição do estudo não estava clara o suficiente, também realizamos a leitura das seções de introdução e conclusão dos estudos primários selecionados. Procuramos manter o resumo o mais breve possível, destacando as utilidades práticas que o trabalho apresentava.

4.2.2 Seleção dos participantes

A escolha pelos alunos universitários de cursos relacionados à computação foi feita pelo fácil acesso a esses respondentes e pelo fato de que a maior parte desses estudantes, além de já terem tido algum contato na prática com projetos e com a comunidade open-source, também estudou, mesmo que brevemente, a maior parte dos conceitos teóricos relacionados ao modelo, o que os torna aptos, em um nível satisfatório, a avaliar a relevância dos trabalhos.

Convidamos os alunos dos cursos de Ciência da Computação, Engenharia da Computação e Licenciatura em Computação da Universidade de Brasília (UnB) para responder ao survey. Os convites, apesar de estarem abertos para estudantes de qualquer disciplina desses cursos, foram feitos com alunos matriculados em disciplinas que fazem parte dos últimos períodos dos cursos, o que favorece a elegibilidade desses participantes.

4.2.3 Submissão do survey e coleta de dados

Para facilitar a disseminação e o alcance, bem como a análise de resultados, foi utilizado um questionário online, com geração automática através da execução de um *script* da Google criado para a produção de questionários. Os sumários produzidos mantiveram suas classificações originais concebidas durante o processo de mapeamento, a saber:

- *Practical Improvements*
- *Aspect Analysis*
- *Advantages/Disadvantages of Open Source*
- *Open Source Teaching*

Primeiramente o questionário se preocupou com a coleta de informações demográficas para posterior análise dos resultados, as informações demográficas consideradas se restringiram ao tipo de envolvimento com o modelo *open-source* e à quantidade de tempo que o indivíduo possuí com esse tipo de envolvimento. Após essa seção inicial, devido à impossibilidade de incorporar um mecanismo de randomização das questões de forma adequada (a randomização não fazia distinção entre as questões demográficas e aquelas relacionadas à avaliação dos artigos), a implementação de tal mecanismo, que teve por objetivo distribuir as análises igualmente entre os artigos (que não precisavam ser todos avaliados), se deu então através da declaração explicita em uma seção anterior aos sumários, a qual pedia para o participante escolher aleatoriamente um subconjunto de no mínimo três trabalhos para avaliação.

4.2.4 Resultados do survey

Após aplicar o survey, obtivemos 376 artigos avaliados, nos quais observamos uma boa distribuição com relação aos dados demográficos de experiência em projetos open-source. As estatísticas foram geradas para diferentes classes: por experiência, por tipo de contribuição, por categoria de contribuições e por resultados gerais.

Dados demográficos

Pelos resultados, apesar do maior grupo ser composto por estudantes com tempo de experiência inferior a um ano (31.3%), a maioria dos participantes possuem ao menos 1 ano de experiência em projetos *open-source* (68.9%). Além disso, tanto o grupo de participantes com menos de 1 ano de experiência, quanto os grupos de 1 a 2 e de 2 a 4 anos de experiência tiveram uma quantidade similar de participantes (31.4% para 1 a 2 anos e 24.9% para 2 a 4 anos). Por fim, tivemos que 12.1% dos participantes tinham 4 anos

ou mais de experiência com o modelo open-source. Esses resultados são apresentados na Figura 4.5. Com relação ao tipo de contribuição, conforme apresentado na Figura 4.6, é possível identificar que a maioria contribuiu com projetos *open-source* (68.6%), 12.5% dos respondentes realizam algum tipo de pesquisa e 18.9% dos participantes apenas consomem produtos da comunidade open-source.

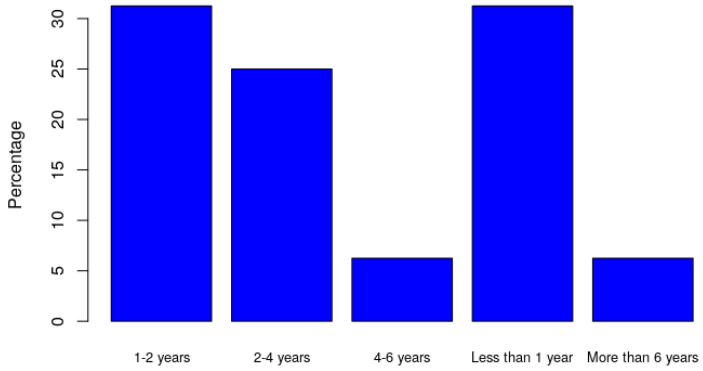

Figura 4.5: *Tempo de experiência com o modelo open-source*

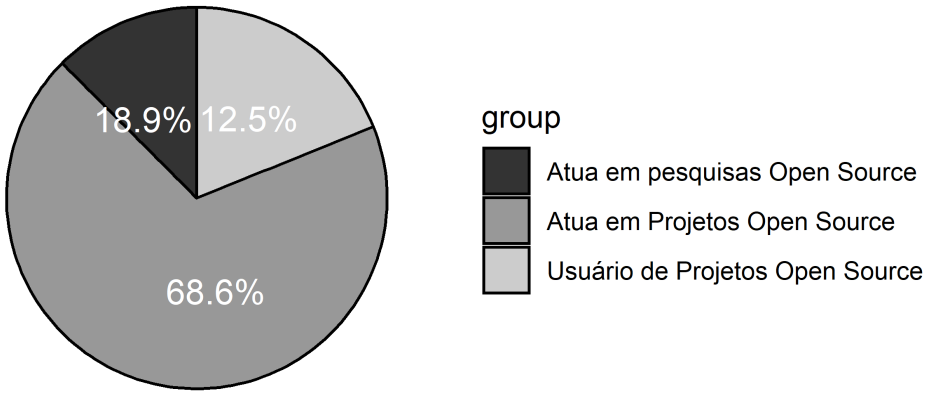

Figura 4.6: *Tipo de contribuição com o modelo open-source*

Percepção da contribuição dos estudos primários analisados

Apesar das possíveis limitações relacionadas à representatividade, diante dos resultados é possível perceber que, predominantemente, os participantes consideram que os trabalhos de pesquisa em Engenharia de Software focados no modelo *open-source* são de fato relevantes, isso porque a maioria das respostas marcadas foram *Concordo* e *Concordo fortemente*. A partir dos resultados também podemos perceber que as análises dentro

de cada uma das categorias formuladas obteve avaliações consideravelmente similares. Para as respostas recebidas, as avaliações *Concordo* e *Concordo plenamente* variaram de 75.90% (Figura 4.8) a 82.71% (Figura 4.7). As avaliações *Discordo* e *Discordo fortemente* variaram de 1.76% (Figura 4.9) a 3.36%, conforme apresentado na Figura 4.8.

Finalmente, o número de avaliações neutras foi significativo e variou entre 14.14% na categoria "Aspect Analysis"(figura 4.7) e 25.43% na categoria "Advantages/Disadvantages of Open-Source"(Figura 4.10). As demais porcentagens para as avaliações neutras podem ser vistas na tabela 4.3

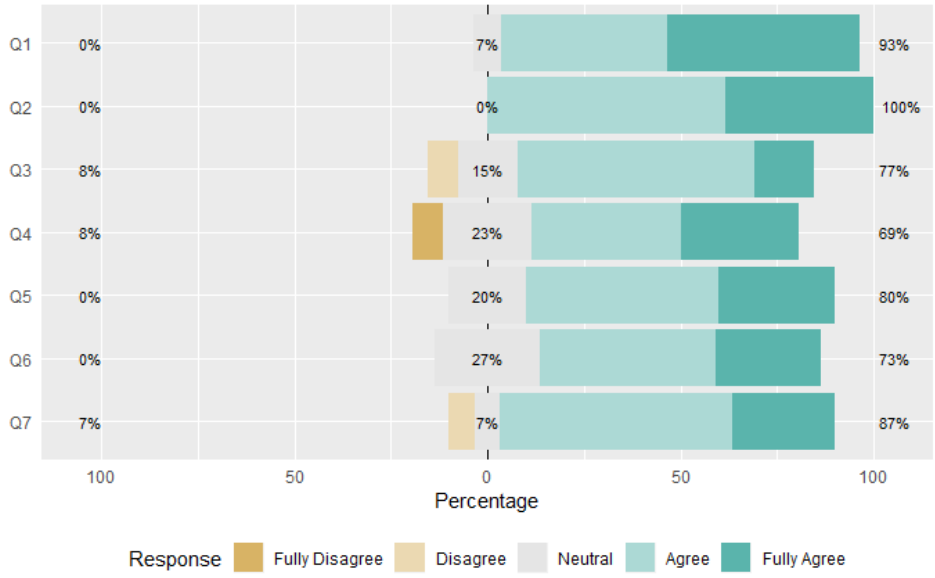

Figura 4.7: *Resultados da categoria Aspect analysis*

Com relação às avaliações negativas, houveram poucas avaliações negativas, indicando que poucos artigos trouxeram resultados que possam ser de fato considerados com pouca relevância. Dessa forma, o número de avaliações *Discordo* e *Discordo fortemente* atingiu um máximo de 3.36% na categoria de resultados *Open Source Teaching*, conforme apresentado na Figura 4.7. Já para as respostas positivas, a Tabela 4.3 apresenta as porcentagens de avaliações positivas para cada categoria de artigos.

Os resultados representam a resposta para a questão de pesquisa, de forma que, analisando os resultados gerais, como obtivemos 77.01% de respostas positivas (avaliações *Concordo* e *Concordo fortemente* combinadas), podemos interpretar os resultados como um forte indicador de que os trabalhos disponíveis na área de Engenharia de Software focados no modelo *open-source* são considerados pelos estudantes de graduação como relevantes para as comunidades open-source.

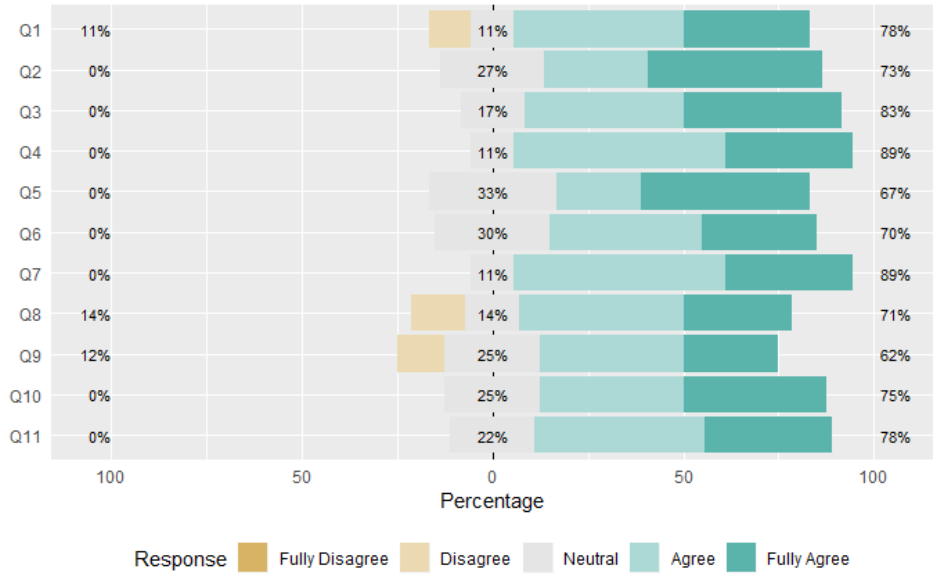

Figura 4.8: *Resultados da categoria open-source teaching*

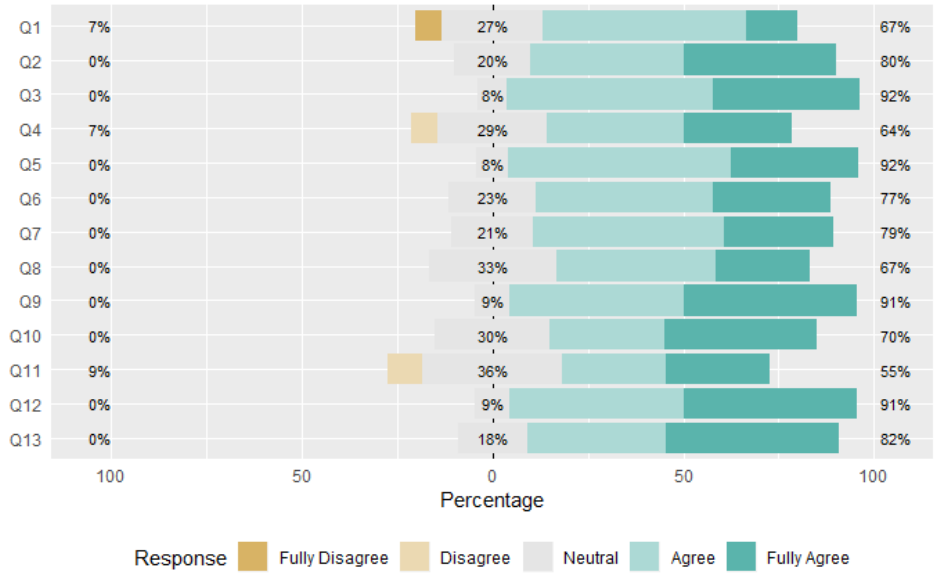

Figura 4.9: *Resultados da categoria Practical Improvements*

4.3 Segundo Mapeamento Sistemático da Literatura

Seguindo a mesma estratégia definida para o primeiro mapeamento apresentado na Seção 4.1, optamos pela divisão do processo de revisão nas seguintes etapas (Figura 4.1): Definição da questão de pesquisa; Condução da pesquisa; Triagem de artigos; Classificação; e extração de resultados. Devido à experiência satisfatória, optamos também por manter o uso da ferramenta *Parsifal* para as fases de planejamento e condução do mapeamento sistemático.

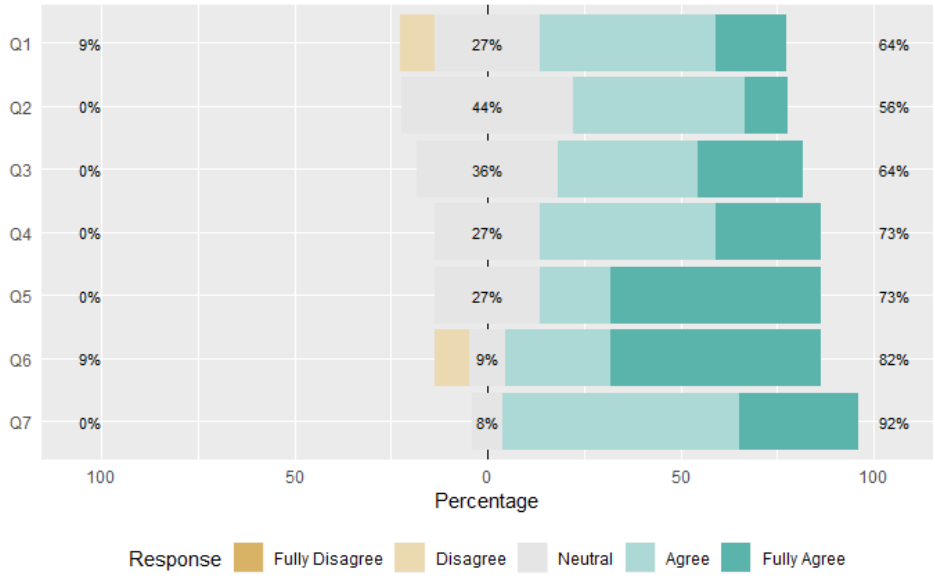

Figura 4.10: *Resultados da categoria Advantages/Disadvantages of Open Source*

Categoria	**Avaliações Positivas**	**Avaliações Neutras**	**Avaliações Negativas**
Aspect analysis	82.71 %	14.14 %	3.29 %
Practical Improvements	77.46 %	20.85 %	1.77 %
Advantages / Disadvantages of Open-Source	72.00 %	25.43 %	2.57 %
Open-Source teaching	75.9 %	20.55 %	3.36 %

Tabela 4.3: Avaliações positivas, neutras e negativas por categoria

4.3.1 Questão de pesquisa do segundo mapeamento

Como nosso interesse continua sendo a busca por um conjunto de trabalhos capazes de expressar a variedade de resultados disponíveis na área de Engenharia de Software que são focados no modelo open-source. O segundo mapeamento teve como objetivo responder a R1.3, definida na Seção 4.1.1.

4.3.2 Estratégia de Busca

Como, idealmente, a condução de um segundo mapeamento sistemático seguindo a estratégia anterior tenderia a resultar em um mesmo conjunto de trabalhos identificados, optamos por tomar algumas medidas a fim de evitar isso. Uma dessas medidas foi con-

siderar bases de dados distintas para a busca. Além disso, outra decisão tomada sob essa justificativa foi a restrição do período de busca para cinco anos em vez de dez anos. Destacamos também que a estratégia de busca considerou apenas a busca automática. Assim, as bases de dados consideradas para esse segundo mapeamento foram:

- Google Scholar;
- DBLP.

A base Google Scholar foi escolhida por ser uma base de pesquisa para a literatura acadêmica de maneira ampla [42]. Já a base DBLP foi escolhida por cobrir exclusivamente publicações da área de Ciência da Computação [26].

4.3.3 String de busca

Como não houve alteração na questão de pesquisa, mantivemos a mesma string de busca da Seção 4.1.2:

4.3.4 Critérios de inclusão e exclusão

Os critérios de inclusão e exclusão foram também mantidos, com exceção de uma pequena alteração com relação à identificação de trabalhos similares, a qual também foi motivada pela intenção de se obter um conjunto distinto de resultados com relação ao primeiro mapeamento da Seção 4.1.

Critérios de inclusão

1. Trabalhos que representam melhorias para as técnicas adotadas com frequência em projetos open-source.
2. Trabalhos que analisam aspectos relacionados ao desenvolvimento de projetos open-source.
3. Trabalhos que analisam as vantagens e desvantagens da adoção do modelo open-source.
4. Trabalhos que analisam os aspectos envolvidos no ensino e aprendizagem do modelo *open-source* ou na introdução de novos participantes ao modelo.

Critérios de exclusão

1. Trabalhos que apresentam resultados similares àqueles já identificados, incluindo aqueles identificados no mapeamento anterior.
2. Trabalhos publicados como *short Paper*.
3. O não cumprimento dos critérios de inclusão.
4. Trabalhos que não pertencem ao escopo da pesquisa.

4.3.5 Seleção de artigos

A seleção de artigos seguiu o mesmo processo adotado na Seção 4.1, conforme apresentado na Figura 4.11. O processo possui as seguintes etapas:

1. Aplicação da *string* de busca nas bases selecionadas.
2. Leitura das palavras chave, do título e, quando insuficiente, do resumo dos artigos identificados na primeira etapa. Também ocorreu o descarte de trabalhos que eram detectados como duplicações. Essa análise preliminar resultou em um total de 176 artigos.
3. Aplicação dos critérios de inclusão e exclusão durante a leitura dos trabalhos identificados na etapa anterior. Ao final dessa etapa, a quantidade remanescentes foi de 64 artigos.
4. Finalmente, seguindo os critérios de pesquisa estabelecidos, a leitura completa dos 64 artigos resultantes da etapa anterior, terminando na seleção de 24 artigos.

4.3.6 Resultados

Ao final da segunda etapa da estratégia adotada, os 176 artigos identificados foram extraídos para a plataforma *Parsifal*, sendo que, conforme a figura 4.12, 45.45% foram extraídos da base do Google Scholar e 54.55% na base DBLP. Na plataforma, as duas etapas seguintes foram conduzidas, resultando em 64 e 24 artigos cada uma.

Através da análise dos artigos resultantes da última etapa da estratégia de seleção, foi possível realizar a classificação dos resultados, a qual manteve as categorias concebidas no mapeamento sistemático realizado na seção 4.1.5, com uma única única alteração na última categoria, a qual, devido à natureza dos resultados identificados, ficou mais adequada após a alteração. Durante a classificação, conflitos a respeito da inclusão de um

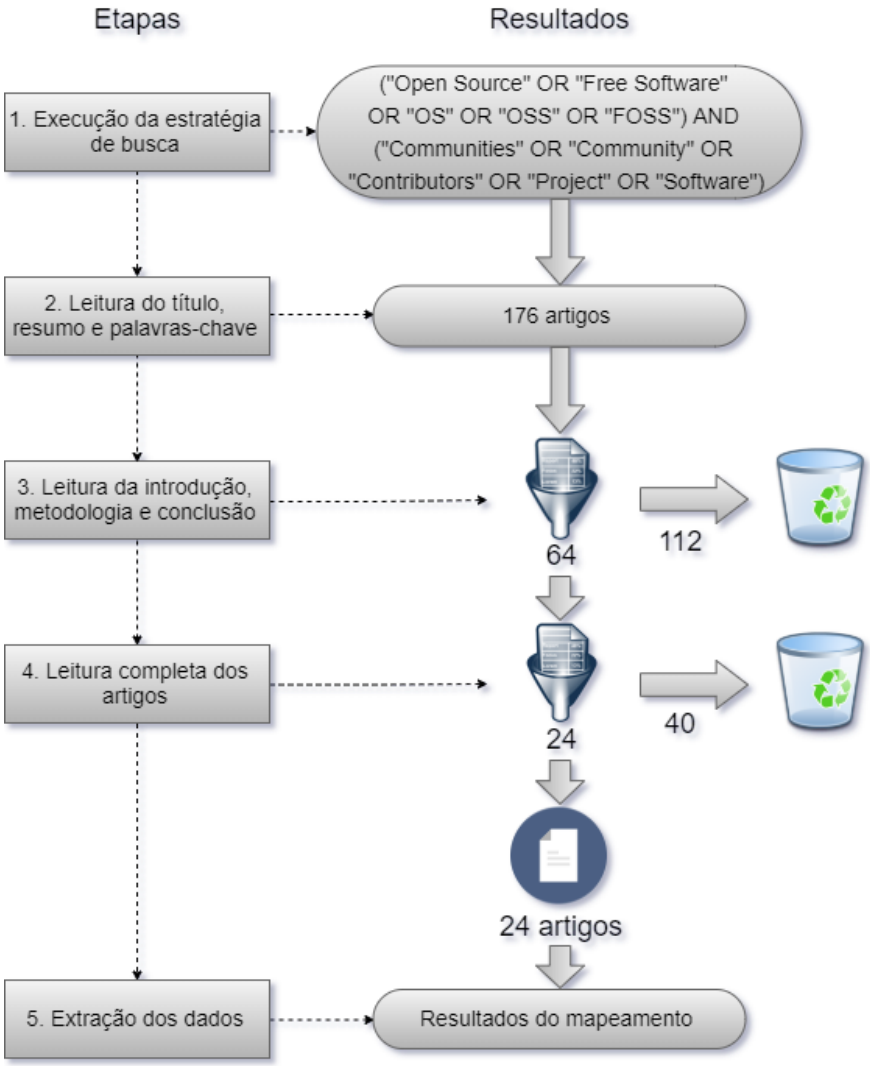

Figura 4.11: Etapas do mapeamento sistemático - Resultados

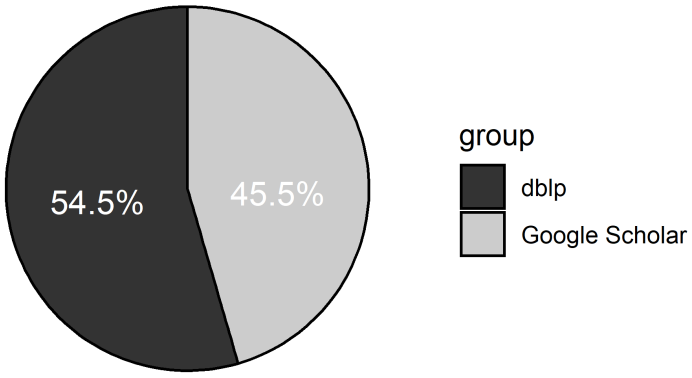

Figura 4.12: *Distribuição dos trabalhos por base digital*

artigo em uma ou mais categorias foram resolvidos de forma a manter uma quantidade homogênea de artigos em cada seção.

Tabela 4.4: Artigos selecionados no segundo mapeamento sistemático

Id	Título	Ref.
	Aspect analysis	
E1	Affiliated Participation in Open Source Communities	[5]
E2	What makes a good contributor? Understanding contributor behavior within large Free / Open Source Software projects – A socialization perspective	[19]
E3	Characterizing the Roles of Contributors in Open-source Scientific Software Projects	[66]
E4	Diversity and Inclusion in Open Source Software (OSS) Projects: Where Do We Stand?	[17]
E5	How Practitioners Perceive Coding Proficiency	[99]
E6	Why Does Code Review Work for Open Source Software Communities?	[4]
	Practical improvements	
E7	REACT - A Process for Improving Open-Source Software Reuse	[56]
E8	Machine Learning Approach for Reliability Assessment of Open Source Software	[15]
E9	Analysis and Detection of Information Types of Open Source Software Issue Discussions	[11]
E10	Team Activities Measurement Method for Open Source Software Development Using the Gini Coefficient	[64]
E11	Modelling Risks in Open Source Software Component Selection	[82]
E12	Characterizing and predicting blocking bugs in open source projects	[39]
	Advantages/Disadvantages of Open Source	
E13	Investigating Open Source Software Benefits in Public Sector	[51]
E14	Critical Barriers to Business Intelligence Open Source Software Adoption	[74]
E15	Beyond free software: An exploration of the business value of strategic open source	[67]
E16	Adoption of Free Libre Open Source Software (FLOSS): A Risk Management Perspective	[55]
E17	Benefits of Open Source Software in Defense Environments	[79]
E18	Economic Benefits of Free and Open Source Software: An Evaluation for Health Sector	[10]
	Open Source Newcomers	
E19	Overcoming Open Source Project Entry Barriers with a Portal for Newcomers	[85]
E20	Training Software Engineers sing Open-Source Software: The Students' Perspective	[72]
E21	What Attract Newcomers to Onboard on OSS Projects? TL;DR: Popularity	[38]
E22	Exploring and Expanding GSE Education with Open Source Software Development	[48]
E23	Let Me In: Guidelines for the Successful Onboarding of Newcomers to Open Source Projects	[87]
E24	Teaching Software Engineering with Free Open Source Software Development:An Experience Report	[90]

Com a alteração mencionada, as categorias ficaram: **1.** *Practical Improvements* - Com trabalhos que representam melhorias para as técnicas adotadas com frequência em projetos open-source; **2.** *Aspect Analysis* - Com trabalhos que analisam aspectos relacionados ao desenvolvimento de projetos open-source; **3.** *Advantages/Disadvantages of Open Source* - Com trabalhos que analisam as vantagens e desvantagens da adoção do modelo open-source; e **4.** *Open Source Newcomers* - Com trabalhos que analisam os aspectos envolvidos na recepção de novos praticantes e aprendizagem do modelo open-source.

A Tabela 4.4, que inclui os resultados identificados, e as categorias sob as quais estes resultados foram classificados representam a resposta para a questão de pesquisa RQ.1 proposta no mapeamento. Tais resultados de fato exibem um alto fator de diversidade, o qual resultou em trabalhos, mesmo dentro de uma mesma categoria, com focos altamente distintos.

4.4 Survey com desenvolvedores e pesquisadores

Com a identificação dos artigos resultantes do mapeamento sistemático da seção 4.3, foi possível a elaboração do questionário a ser utilizado na investigação da relevância da pesquisa de Engenharia de Software focada no modelo *open-source* utilizando desenvolvedores e pesquisadores como avaliadores. Como mencionado, a realização do *survey* nos permite obter de forma direta a percepção dessa população com relação à relevância da literatura disponível. Assim, a questão de pesquisa que guiou este *survey* foi a RQ.3.

Para a condução do survey, utilizamos uma adaptação do *framework* proposto por Lo et al.[62], cujo processo foi dividido nas seguintes fases: *Seleção dos artigos de interesse*; *Seleção dos participantes*; *Submissão do survey e coleta de dados*;

4.4.1 Seleção dos artigos de interesse

Os artigos de interesse foram identificados através da realização do mapeamento sistemático da seção 4.3, os quais, dada a metodologia adotada, acreditamos que sejam capazes de capturar o cenário atual da pesquisa focada em open-source. Para a apresentação desses artigos no survey, foram utilizados os títulos e resumos originais dos artigos, sem sua sumarização, como proposto originalmente pelo *framework* adaptado.

4.4.2 Seleção de participantes

Os profissionais desenvolvedores e pesquisadores das comunidades *open-source* representam o público alvo de maior interesse para a investigação da relevância da pesquisa em *open-source* e por essa razão foram escolhidos como os participantes do survey. A procura por essa população se deu nas próprias comunidades *open-source* através das listas de emails das comunidades e, devido ao fato dos desenvolvedores e pesquisadores estarem distribuídos geograficamente, entende-se que desenvolvedores e pesquisadores de comunidades *open-source* de possivelmente todas as partes do mundo foram convidados a participar no survey. Foram consideradas 40 listas de comunidades *open-source* que variaram em número de integrantes entre 84 e 1224 inscritos cada uma.

4.4.3 Submissão do survey e coleta de dados

O *survey* foi implementado através da execução de um *script* para geração de questionários online produzido nas plataformas Google Script e Google Forms, aonde a primeira seção foi reservada para a coleta de informações demográficas e a seção seguinte para a apresentação dos artigos aos respondentes. As estatísticas coletadas não foram apresentadas aos participantes.

Na seção referente às informações demográficas, foram solicitadas as seguintes informações:

- Local de trabalho
- Grau de formação
- Idade
- Tipo de experiência com o modelo open-source
- Tempo de experiência com o modelo open-source

Os artigos apresentados na segunda seção do questionário foram agrupados nas categorias definidas na seção 4.1.5, incluindo a alteração na última categoria, mencionada na seção 4.3.6. Dessa forma, as categorias utilizadas foram:

- *Aspect Analysis*
- *Practical Improvements*
- *Advantages/Disadvantages of Open Source*
- *Open Source Newcomers*

Para cada artigo apresentado, solicitou-se a avaliação por parte do participante com relação à relevância dos resultados contidos naquele trabalho de pesquisa através da seguinte pergunta: *"how important do you evaluate this paper contribution?"*. Sendo as possíveis respostas apresentadas em uma escala com as seguintes opções: *"Very Important"*, *"Important"*, *"Moderately Important"*, *"Slightly Important"*, *"Not Important"*.

4.4.4 Resultados do survey

Após disponibilizar o *survey* por 1 mês, foram contabilizadas 1789 avaliações de artigos. Com esses resultados, esta seção se propõe a responder a questão de pesquisa *RQ3: Qual é a relevância da literatura em open-source a partir da perspectiva dos profissionais que atuam nessas comunidades?*.

Dados demográficos

A distribuição correspondente ao nível de educação dos participantes pode ser vista na figura 4.13, na qual podemos perceber que a maioria dos participantes possui um grau de formação equivalente ao doutorado (27.3%), seguido por participantes com apenas graduação (20.8%). Uma grande parte dos participantes também possuem título de mestre (19.5%) ou é estudante de mestrado (19.5%).

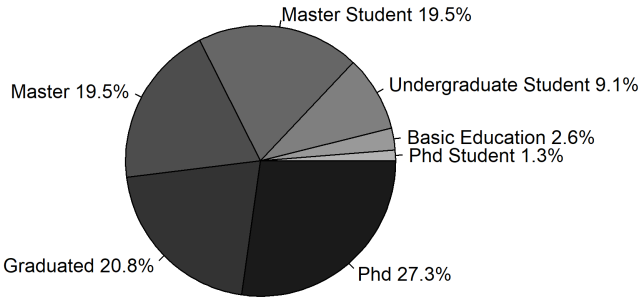

Figura 4.13: Grau de formação dos participantes do survey

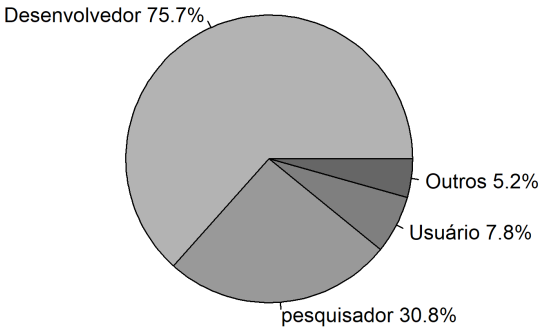

Figura 4.14: Tipo de contribuição dos participantes do survey

Com relação à idade dos participantes, temos a figura 4.15, que apresenta uma boa distribuição de idade entre os participantes. De forma que a maioria dos participantes estão na faixa etária de 41 a 50 anos (26%). Além disso, as faixas etárias de 36 a 40 (18.2%), de 51 a 60 (15.6%) e de 26 a 30 (14.3%) também representaram grande parte dos participantes e ficaram com porcentagens similares. As faixas etárias menos representativas foram as de participantes com mais de 61 anos (4.9%).

Conforme pode ser visto na figura 4.14, o grupo de participantes foi composto majoritariamente por desenvolvedores (75.7%). Já os pesquisadores representaram 30.8% dos participantes. Por fim, usuários e outros participantes da comunidade *open-source* representaram juntos 13% dos respondentes.

Analisando o tempo de experiência com o modelo *open-source* (figura 4.16), percebemos que a maioria dos respondentes (36.4%) possuem mais de 10 anos de experiência com o modelo. Também destacamos que as demais faixas de experiência ficaram com distribuições similares (20.8% a 22.1%), de forma que 58.5% possuem mais de 4 anos de experiência.

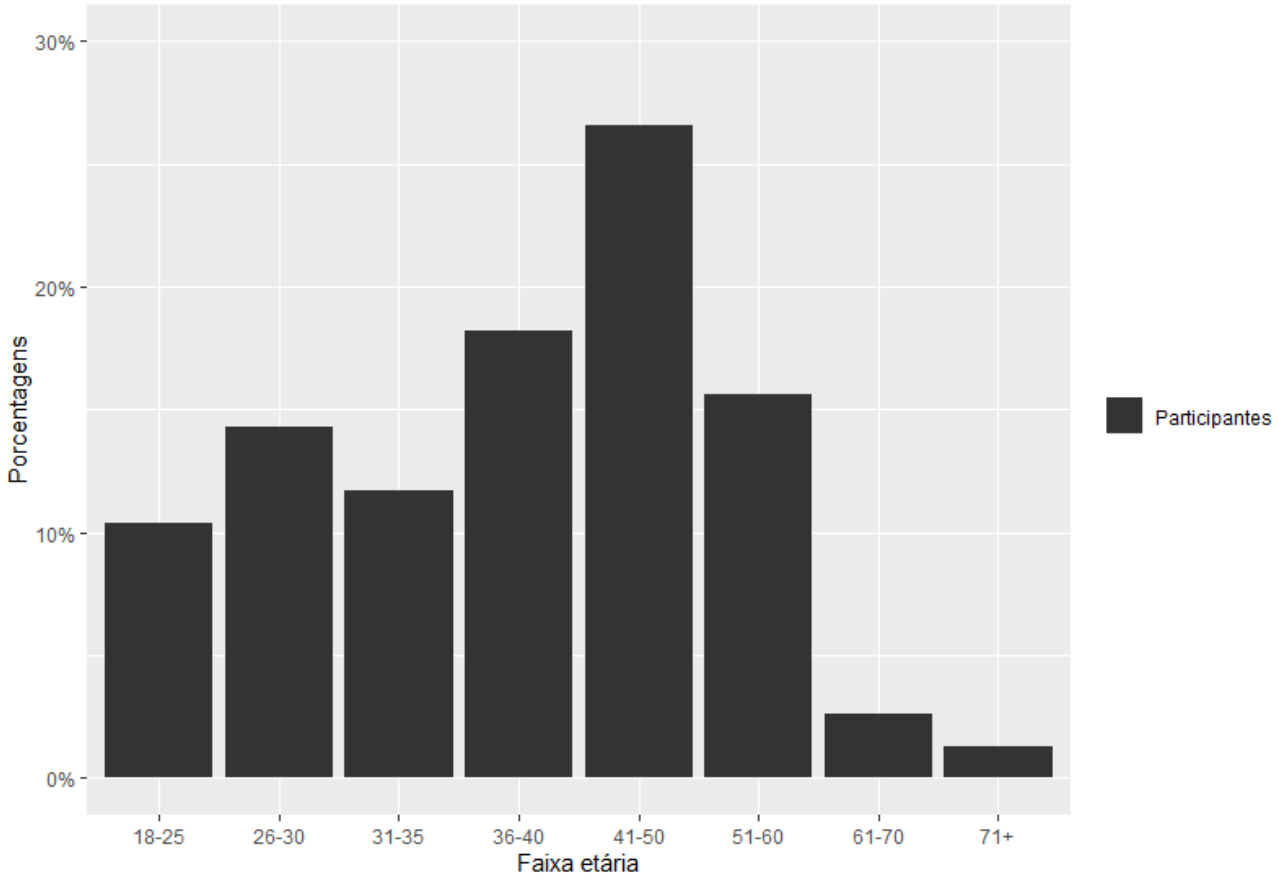

Figura 4.15: Distribuição da idade dos participantes do survey

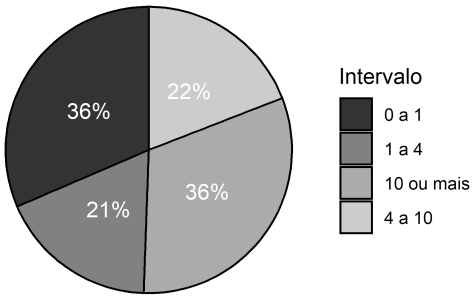

Figura 4.16: Distribuição do tempo de experiência dos participantes do survey

Percepção da contribuição dos estudos primários analisados

O conjunto de resultados obtidos demonstra um cenário muito positivo com relação à relevância percebida pela pesquisa em open-source, de tal maneira que as avaliações consideradas relevantes foram predominantes e as avaliações marcadas como "Very Important"foram muito significativas para todas as categorias, variando entre 20.08% a 23.69% (tabela

Categoria	Very Important	Important	Moderately Important	Slightly Important	Not Important
Practical Improvements	21.90%	31.54%	27.76%	12.91%	5.89%
Aspects analysis	20.08%	32.86%	24.75%	17.22%	5.09%
Advantages/Disadvantages	22.38%	33.11%	22.83%	15.07%	6.62%
Open-Source Newcomers	23.69%	33.82%	27.56%	12.2%	2.75%

Tabela 4.5: Avaliações por categoria

Cada uma das categorias de trabalhos obteve uma avaliação similar às demais em todas as opções de avaliação, com variações de 3.61% para as avaliações "Very Important", 2.28% para "Important", 4.93% para "Moderately Important", 5.02% para "Slightly Important"e 3.87% para "Not Important".

O número de avaliações "Not important"foi muito baixo no total (5.09%), atingindo um máximo de 6.62% na categoria "Advantages Disadvantages of Open Source"(tabela 4.5), indicando que 94.91% dos trabalhos são pelo menos levemente relevantes e que 80.56% dos trabalhos são pelo menos moderadamente relevantes. Já o número de avaliações "Very Important"representou uma boa parte dos resultados, atingindo o máximo de 23.69% na categoria "Open Source Newcomers".

De modo a permitir uma melhor visualização dos trabalhos, realizamos a construção da fórmula a seguir, na qual definimos *"Very Important"*, *"Important"*, *"Moderately Important"*, *"Slightly Important"* e *"Not Important"* respectivamente como VI, I, MI, SI e NI. Com isso, os seguintes scores foram então calculados:

- **MR-score** – Muito relevante: A porcentagem de avaliações "Very Important"

$$\frac{VI}{VI + I + MI + SI + NI}$$

- **R-score** – Relevante: A porcentagem de avaliações "Very Important", "Imporant"ou "Moderately Important"

$$\frac{VI + I + MI}{VI + I + MI + SI + NI}$$

- **PR-score** – Pouco relevante: A porcentagem de avaliações "Slightly Important"ou "Not Important"

$$\frac{SI + NI}{VI + I + MI + SI + NI}$$

Com os scores calculados para todos os resultados, fomos capazes de determinar os 5 trabalhos mais bem avaliados de acordo com a relevância de pesquisa. Os resultados podem ser vistos na tabela 4.6, na qual os dois primeiros artigos com maior score pertencem à categoria "Open Source Newcomers", justificando o motivo dessa categoria ter atingido a melhor avaliação quanto a sua relevância (tabela 4.5).

Posição	ID	Título	MR-Score	R-Score	PR-Score
#1	E24	Teaching Software Engineering with Free Open Source Software	0.274	0.904	0.096
#2	E23	Let Me In: Guidelines for the Successful Onboarding of Newcomers to Open Source Projects	0.268	0.901	0.099
#3	E12	Characterizing and predicting blocking bugs in open source projects	0.395	0.895	0.105
#4	E6	Why Does Code Review Work for Open Source Software Communities?	0.263	0.882	0.118
#5	E11	Modelling Risks in Open Source Software Component Selection	0.227	0.880	0.120

Tabela 4.6: Os cinco trabalhos com maior score de relevância (R-score)

Categoria	MR-score	R-score	PR-score
Practical Improvements	0.219	0.812	0.188
Aspects analysis	0.201	0.777	0.223
Advantages/Disadvantages	0.224	0.783	0.217
Open-Source Newcomers	0.237	0.851	0.150

Tabela 4.7: Scores por categoria

Os scores também foram calculados para as categorias, de modo que a categoria com o melhor R-score foi a "Open Source Newcomers"(0.851). Essa categoria também obteve o melhor MR-score, atingindo 0.237 de pontuação. Os scores para as categorias podem ser vistos na tabela 4.7.

Com o exposto, somos capazes de concluir que os desenvolvedores e pesquisadores de fato consideram que os trabalhos de pesquisa em Engenharia de Software voltados ao *open-source* como relevantes, em especial aqueles que tratam de questões associadas à iniciação de novos desenvolvedores com o modelo.

Capítulo 5

Resultados

5.1 Discussão dos Resultados

Os resultados individuais foram compatíveis entre si e muito favoráveis para os trabalhos de pesquisa de Engenharia de Software focados no modelo open-source, apresentando uma média de 78.80% de avaliações positivas com relação à relevância dessa literatura.

5.1.1 Dados demográficos gerais

Considerando os dois surveys realizados, com relação ao papel desses participantes dentro das comunidades open-source, tivemos que 72.15% dos participantes são desenvolvedores que contribuem com projetos. Também verificamos que 24.85% são pesquisadores e que 10.15% se tratam de usuários que também fazem parte dessas comunidades. A distribuição geral dos participantes pode ser vista na figura 5.1. Destaca-se que o número de pesquisadores foi mais alto no *survey* realizado com profissionais pesquisadores e desenvolvedores, no qual a porcentagem foi de 30.8% dos participantes, em contraste com 18.9% no *survey* com estudantes.

38.55% dos participantes possuem entre 1 a 4 anos de experiência, seguidos pela faixa de 4 anos ou mais, que atingiu uma porcentagem considerável de 35.55% de representatividade. Além disso, 26.05% possui um ano ou menos de experiência. Cabe destacar que o *survey* com desenvolvedores e pesquisadores contou com uma porcentagem muito mais alta de respondentes com mais de 4 anos de experiência (58.5%) do que o *survey* realizado com estudantes (12.6%), estando esse fato de acordo com as características de cada grupo considerado, aonde parece plausível acreditar que estudantes de graduação possuam bem menos experiência. Esses dados podem ser vistos na figura 5.2.

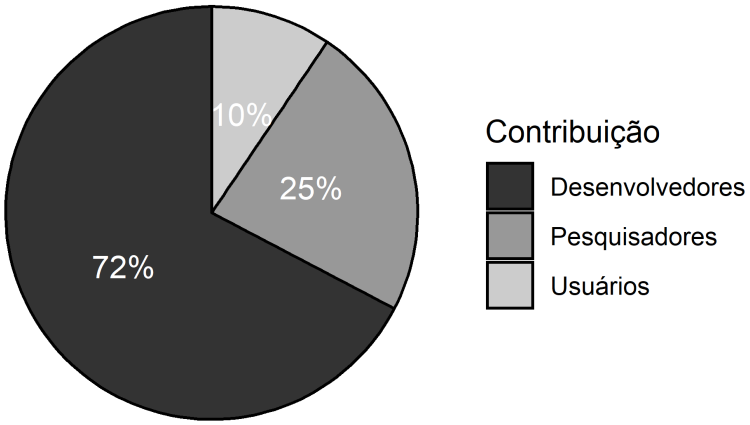

Figura 5.1: *Tipo de contribuição com o modelo open-source*

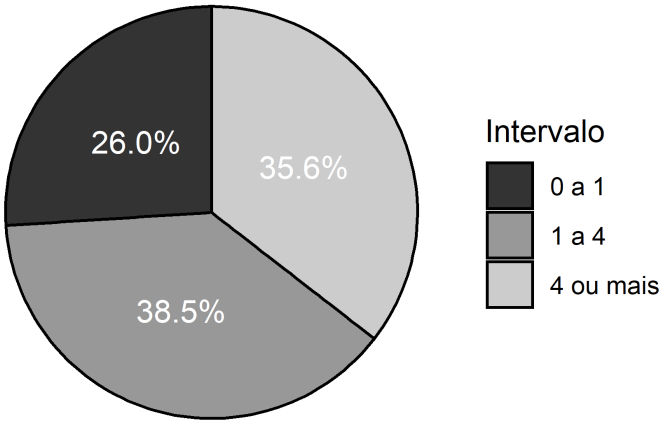

Figura 5.2: *Tempo de contribuição com o modelo open-source*

Como as comunidades *open-source* também contam com membros que são estudantes de graduação de cursos de ciência da computação e afins, a porcentagem geral de respondentes desse grupo acabou sendo de 54.55%

5.1.2 Avaliações gerais

As avaliações positivas gerais podem ser vistas na tabela 5.1, na qual temos uma média de avaliação positiva de 78.80%, sendo que a categoria "Open Source Newcomers"(ou a sua equivalente: "Open Source Teaching") foi a melhor avaliada no cenário geral, atingindo 80.49% de associação positiva com relação a sua relevância. Já a categoria com menor

Categorias	Positivas	Neutras	Negativas
Aspect Analysis	80.20%	15.68%	4.59%
Practival improvements	79.33%	16.88%	3.83%
Advantages / Disadvantages	75.16%	20.25%	4.10%
Open Source Newcomers	80.49%	16.38%	3.06%

Tabela 5.1: Avaliações gerais das categorias

avaliação positiva foi a "Advantages/Disadvantages of Open Source", com 75.16%. Uma comparação entre as avaliações positivas para cada categoria pode ser vista na figura 5.3

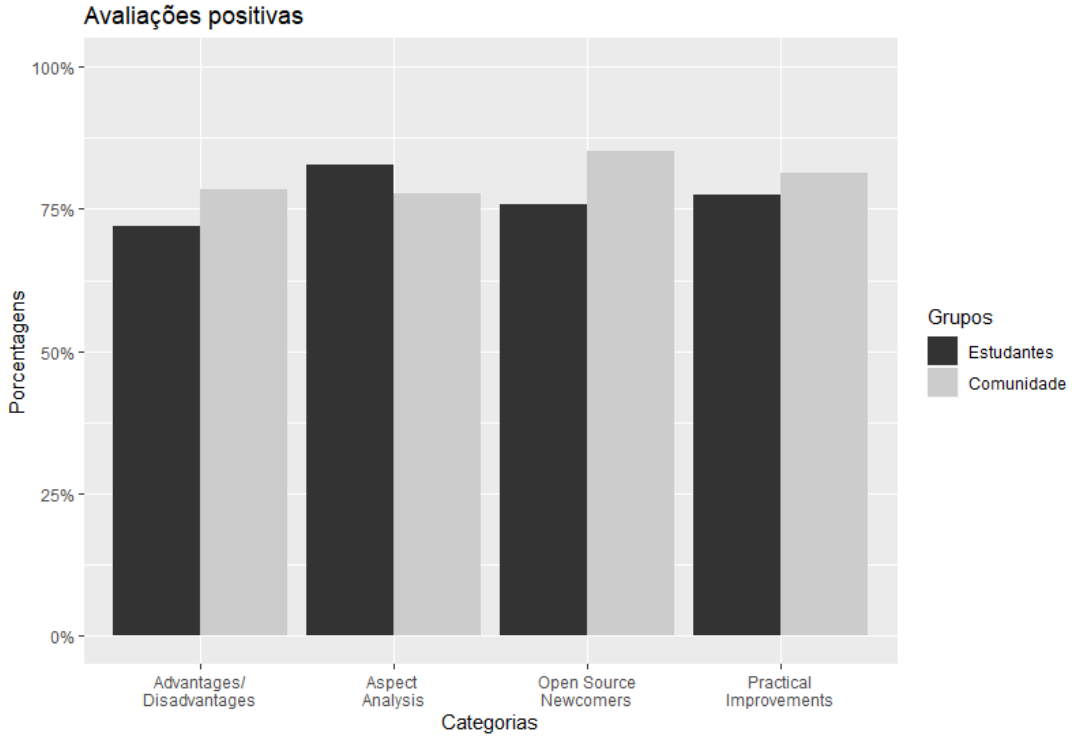

Figura 5.3: *Comparação entre as avaliações positivas das categorias para cada survey*

Considerando as avaliações negativas gerais (tabela 5.1), percebemos que a média entre as categorias foi muito baixa, representando 3.90% das avaliações. As diferenças negativas gerais entre as categorias variou entre 3.06% (categoria "Open Source Newcomers/Teaching") e 4.59% (categoria "Open Source Aspect Analysis"). A porcentagem de avaliações neutras foi considerável, e atingiu um máximo de 20.25% na categoria "Advantages/Disadvantages of Open Source"e um mínimo de 15.68% na categoria "Open Source Aspect Analysis", sendo que no *survey* com estudantes essa média foi mais alta do que no *survey* com as comunidades open-source, calculadas respectivamente em 20.24% e 14.35%.

A média de variação entre as diferentes avaliações das categorias foi consideravelmente baixa, sendo calculado em 5.36%. Tendo em vista que esse valor representa a média de discordância geral entre as avaliações feitas pelos dois grupos, percebemos que houve uma

Categorias	Positivas	Neutras	Negativas
Aspect Analysis	5.02%	3.08%	2.60%
Practival improvements	3.74%	7.94%	4.12%
Advantages / Disadvantages	6.32%	10.36%	3.05%
Open Source Newcomers	9.17%	8.35%	0.61%

Tabela 5.2: Diferenças entre as avaliações das categorias pelos dois surveys

certa concordância entre as avaliações de ambas populações de surveys. As diferenças entre as avaliações podem ser vistas na tabela 5.2. Para as avaliações individuais, as avaliações negativas foram as que mais demonstraram concordância, diferindo em média apenas 2.60% entre os dois surveys. Já as avaliações neutras foram as que menos concordaram entre si, diferindo em média 7.43% entre os surveys. Por fim, as avaliações positivas diferiram em média 6.06%. Do ponto de vista das categorias, a categoria que mais diferiu na análise dos dois grupos foi a "Advantages/Disadvantages of Open Source", chegando a 6.58% de diferença e a que menos diferiu foi a "Open Source Aspect Analysis", com apenas 3.57%.

Ainda tratando das diferenças entre as avaliações dos grupos de cada survey, destacamos que a avaliação com maior concordância entre os grupos foi na categoria "Open Source Newcomers/Teaching"com relação a sua avaliação negativa de relevância (tabela 5.2), chegando a apenas 0.61% de diferença, sendo 3.36% para estudantes e 2.75% para comunidade open-source. Já a avaliação com maior discordância foi na categoria "Advantages/Disadvantages of Open Source", atingindo 10.36% de diferença entre as avaliações de cada grupo (tabela 5.2), com 25.43% para estudantes e 15.07% para comunidade open-source.

5.2 Ameaças à validade

Nós reconhecemos a existência das seguintes ameaças à validação deste trabalho.

O *survey* feito com estudantes considerou a elaboração de sumários para os artigos apresentados, os quais foram elaborados pelos autores deste trabalho, dessa forma, é possível que detalhes que modificariam a avaliação feita pelo estudante tenham sido removidos, causando uma perda na acurácia do survey. No caso dos surveys com a comunidade open-source, essa ameaça não esteve presente, já que para esse grupo consideramos os próprios resumos originais dos artigos. Essa ameaça poderia ser tratada com a utilização dos resumos originais dos artigos, como foi feito com as comunidades open-source, entretanto, essa alteração traria outras ameaças em compensação, como a possível dificuldade dos alunos entenderem os resultados ou mesmo uma queda na retenção desses respondentes.

Devido ao perfil dos respondentes do primeiro survey, é possível que os estudantes não conheçam suficientemente alguns dos conceitos envolvidos nos resultados fornecidos pelos trabalhos, o que resultaria provavelmente nesses participantes avaliando os artigos de forma neutra ou, no pior caso, de forma positiva ou negativa. Entretanto, acreditamos que essa seja uma ameaça inerente à proposta do survey, já que essa é justamente uma das características que desejamos comparar quando o interesse reside na percepção por parte de estudantes e não de profissionais ou pesquisadores altamente experientes.

Além disso, diferente do *survey* com estudantes, no qual os resumos estavam escritos na língua nativa dos participantes, no caso das comunidades open-source, que se encontram geograficamente distribuídas, é possível que os participantes tenham sofrido algum tipo de interferência ao julgar os trabalhos devido ao fato de os resumos estarem exclusivamente em inglês. Isso dificilmente poderia ser contornado, já que as comunidades geralmente não restringem a participação de um membro com base na sua localização geográfica.

Existe também uma ameaça ao trabalho com relação à representatividade dos estudantes de graduação, já que foram considerados apenas estudantes de uma única universidade (Universidade de Brasília). No caso do *survey* com as comunidades, entretanto, essa ameaça não esteve presente, isso porque foram consideradas diversas comunidades diferentes e cada comunidade possui indivíduos dos mais variados perfis. Idealmente o *survey* com estudantes teria sido disseminado em várias universidades não só do mesmo país mas também do mundo. Apesar disso, acreditamos que os alunos da universidade de Brasília possuem o conhecimento necessário para representar de forma satisfatória o perfil de um aluno de graduação em um curso relacionado à computação.

Com relação aos mapeamentos sistemáticos da literatura realizados, a metodologia adotada para aumentar o índice de variabilidade dos resultados e ao mesmo tempo reduzir a quantidade de trabalhos se mostra como uma ameaça a ser considerada, já que trabalhos com resultados similares poderiam ser avaliados de formas diferentes pelos participantes. Apesar disso, esse foi um mecanismo essencial para a proposta desse trabalho, já que se tratou de uma intervenção ativa por parte dos autores que filtraram toda a literatura considerada selecionando um conjunto manuseável de trabalhos. As alternativas para essa abordagem resultariam em uma quantidade impraticável de trabalhos ou em conjuntos menores que seriam capazes de capturar apenas partes limitadas do cenário da pesquisa em open-source.

5.3 Trabalhos futuros

Trabalhos futuros podem incluir a expansão desta pesquisa, superando algumas limitações declaradas, aumentando o número de universidades ou considerando apenas profissionais

experientes de código aberto. Dessa forma, poderíamos fornecer um feedback mais eficaz aos pesquisadores. Outra possibilidade é avaliar as motivações dos participantes da pesquisa em relação às avaliações fornecidas. Com uma quantidade maior de participantes, isso poderia ser alcançado sem prolongar o tempo consumido, o que poderia afetar a qualidade dos resultados.

Capítulo 6

Conclusão

Neste trabalho estávamos preocupados com a relevância dos trabalhos de pesquisa em Engenharia de Software focados no modelo open-source.

Para investigar a relevância percebida dessa literatura por estudantes de graduação, realizamos um mapeamento sistemático da literatura de forma a encontrar trabalhos capazes de representar a diversidade de resultados disponíveis da Engenharia de Software voltados ao modelo, a qual resultou em 38 artigos selecionados, em seguida esses trabalhos identificados foram então sumarizados pelos autores e um *survey* com estudantes de graduação do curso de ciência da computação e afins da universidade de Brasília foi conduzido, no qual obtivemos 376 avaliações de sumários.

Também investigamos a relevância percebida a partir da perspectiva dos profissionais pesquisadores e desenvolvedores das comunidades open-source, realizando para tal, um novo mapeamento sistemático com o objetivo de aumentar a variabilidade dos resultados identificados por essa pesquisa, o qual resultou em 24 artigos. Após a realização do novo mapeamento, foi conduzido um *survey* com esses profissionais, em que se considerou, em vez de sumários, os próprios resumos dos artigos identificados, resultando em um total de 1789 avaliações de artigos.

Os resultados de ambos os mapeamentos revelaram conjuntos de trabalhos com um alto fator de diversidade em relação aos tipos das contribuições fornecidas. Tais resultados foram agrupados em categorias de acordo com a classificação de contribuição, as quais foram divididas posteriormente em seções para a realização de cada um dos surveys.

A condução dos surveys teve resultados muito positivos a favor da pesquisa em open-source, revelando que 78.80% de todos os participantes consideram a literatura disponível relevante. Para cada grupo, essas porcentagens foram de 77.02% para os estudantes de graduação e de 80.57% para as comunidades open-source. A categoria de resultados da literatura mais bem avaliada (80.49% de avaliações positivas) foi a categoria que contém trabalhos que analisam os aspectos envolvidos no ensino e aprendizagem do modelo

open-source ou na integração de novos membros a essas comunidades. Já a categoria que recebeu o maior número de avaliações como sem relevância, apesar de estas serem consideravelmente baixas (média de 3.9%), foi a categoria que contém trabalhos que analisam aspectos relacionados ao desenvolvimento de projetos com características específicas do modelo *open-source* (4.59% de avaliações negativas).

Diante dos resultados obtidos, verificamos que de fato a literatura de Engenharia de Software focada em *open-source* é considerada tanto por estudantes de graduação quanto pela comunidade *open-source* como relevante. Esses dados podem ajudar tanto pesquisadores a acessar a opinião dessas populações quanto a relevância dos tipos de contribuições fornecidas, quanto desenvolvedores que estejam interessados em obter uma visão geral do cenário de pesquisa no modelo open-source.

Referências

[1] (2019). Licenses standards opensource. 9

[2] (2019). The Open Source Definition | Open Source Initiative. 7

[3] Agrawal, A., Rahman, A., Krishna, R., Sobran, A., and Menzies, T. (2018). We don't need another hero?: The impact of "heroes"on software development. In *2018 IEEE/ACM 40th International Conference on Software Engineering: Software Engineering in Practice Track (ICSE-SEIP)*, pages 245–253, New York, NY, USA. ACM. 6. 20, 21

[4] Alami, A., Cohn, M. L., and Wasowski, A. (2019). Why does code review work for open source software communities? *2019 IEEE/ACM 41st International Conference on Software Engineering (ICSE)*, pages 1073–1083. 33

[5] Alami, A. and Wasowski, A. (2019). Affiliated participation in open source communities. 33

[6] Aldaeej, A. and Badreddin, O. (2016). Towards promoting design and UML modeling practices in the open source community. In *2016 IEEE/ACM 38th International Conference on Software Engineering Companion (ICSE-C)*, pages 722–724, New York, NY, USA. ACM. 10. 20

[7] Alfayez, R., Behnamghader, P., Srisopha, K., and Boehm, B. (2017). How does contributors involvement influence open source systems. In *2017 IEEE 28th Annual Software Technology Conference (STC)*, pages 1–8. IEEE. 20. 20

[8] An, S., Wang, X., and Liu, L. (2011). A reference model for evaluating performance of open source community. In *2011 International Conference on Electrical and Control Engineering*, pages 31–34, 10.1109/ICECENG.2011.6057372. IEEE. 18. 20, 21

[9] Aparicio, M. and Costa, C. J. (2012). Macroeconomics Leverage Trough Open Source. In *Proceedings of the Workshop on Open Source and Design of Communication*, OSDOC '12, pages 19–24, New York, NY, USA. ACM. event-place: Lisboa, Portugal. 1

[10] Arslan, O. A. (2014). Economic benefits of free and open source software: An evaluation for health sector. 33

[11] Arya, D., Wang, W., Guo, J. L. C., and Cheng, J. (2019). Analysis and detection of information types of open source software issue discussions. *2019 IEEE/ACM 41st International Conference on Software Engineering (ICSE)*, pages 454–464. 33

[12] Barcomb, A. (2014). Volunteer attraction and retention in open source communities. In *OpenSym*. 8

[13] Bayati, S. (2018). Poster: Understanding newcomers success in open source community. In *2018 IEEE/ACM 40th International Conference on Software Engineering: Companion (ICSE-Companion)*, pages 224–225, New York, NY, USA. ACM. 30. 20, 21

[14] Begel, A. and Zimmermann, T. (2014). Analyze this! 145 questions for data scientists in software engineering. *Proceedings of the 36th International Conference on Software Engineering - ICSE 2014*. 12

[15] Behera, R. K., Rath, S. K., Misra, S., León, M., and Adewumi, A. (2019). Machine learning approach for reliability assessment of open source software. In *ICCSA*. 33

[16] Bianco, V. d., Lavazza, L., Morasca, S., and Taibi, D. (2011). A survey on open source software trustworthiness. *IEEE Software*, 28(5):67–75. 26. 20, 21

[17] Bosu, A. and Sultana, K. Z. (2019). Diversity and inclusion in open source software (oss) projects: Where do we stand? *2019 ACM/IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM)*, pages 1–11. 33

[18] Buchta, J., Petrenko, M., Poshyvanyk, D., and Rajlich, V. (2006). Teaching evolution of open-source projects in software engineering courses. In *2006 22nd IEEE International Conference on Software Maintenance*, pages 136–144. IEEE. 33. 20

[19] Carillo, K., Huff, S., and Chawner, B. (2017). What makes a good contributor? understanding contributor behavior within large free/open source software projects – a socialization perspective. *The Journal of Strategic Information Systems*, 26(4):322 – 359. 33

[20] Carillo, K., Huff, S. L., and Chawner, B. (2014). What makes a good contributor? understanding contributor behavior within large free/open source software projects - a socialization perspective. *J. Strategic Inf. Sys.*, 26:322–359. 9

[21] Carrington, D.A. (2008). What can software engineering students learn from studying open source software? carrington, d.a. *International Journal of Engineering Education*, pages 729–737. 29. 20

[22] Carver, J. C., Dieste, O., Kraft, N. A., Lo, D., and Zimmermann, T. (2016a). How Practitioners Perceive the Relevance of ESEM Research. In *Proceedings of the 10th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement - ESEM '16*, pages 1–10, Ciudad Real, Spain. ACM Press. 1

[23] Carver, J. C., Dieste, O., Kraft, N. A., Lo, D., and Zimmermann, T. (2016b). How practitioners perceive the relevance of esem research. *Proceedings of the 10th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement - ESEM '16*. 12

[24] Chesbrough, H. and Brunswicker, S. (2014). A Fad or a Phenomenon?: The Adoption of Open Innovation Practices in Large Firms. *Research-Technology Management*, 57:16–25. 1

[25] Crowston, K. and Howison, J. (2006). Hierarchy and centralization in free and open source software team communications. *Knowledge, Technology and Policy*, 18:65–85. 8

[26] dblp team (2019). What is the scope of dblp? Accessed: 2019-09-21. 30

[27] de Oliveira, E. C. C., Conte, T., Cristo, M., and Mendes, E. (2016). Software project managers' perceptions of productivity factors: Findings from a qualitative study. In *ESEM*. 11

[28] DeKoenigsberg, G. (2008). How successful open source projects work, and how and why to introduce students to the open source world. In *2008 21st Conference on Software Engineering Education and Training*, pages 274–276. 38. 20

[29] Delorey, D. (2007). Observational Studies of Software Engineering Using Data from Software Repositories. *Theses and Dissertations*. 6

[30] Depoorter, G. (2014). Open source software: a social and economic innovation. 1

[31] Devanbu, P., Zimmermann, T., and Bird, C. (2016). Belief & evidence in empirical software engineering. *Proceedings of the 38th International Conference on Software Engineering - ICSE '16*. 12

[32] Dorodchi, M. and Dehbozorgi, N. (2016). Utilizing open source software in teaching practice-based software engineering courses. In *2016 IEEE Frontiers in Education Conference (FIE)*, pages 1–5. 37. 20

[33] Ebert, C. (2007). Open source drives innovation. *IEEE Software*, 24(3):105–109. 9. 20, 22

[34] Fan, Q., Wang, H., Yin, G., and Wang, T. (2015). Ranking open source software based on crowd wisdom. In *2015 6th IEEE International Conference on Software Engineering and Service Science (ICSESS)*, pages 966–972, 10.1109/ICSESS.2015.7339215. IEEE. 19, 20

[35] Fielding, R. T. (2005). Software architecture in an open source world. In *Proceedings. 27th International Conference on Software Engineering, 2005. ICSE 2005.*, pages 43–, 10.1109/ICSE.2005.1553541. IEEE. 8. 19, 20

[36] Franch, X., Fernández, D. M., Oriol, M., Vogelsang, A., Heldal, R., Knauss, E., Travassos, G. H., Carver, J. C., Dieste, O., and Zimmermann, T. (2017). How do Practitioners Perceive the Relevance of Requirements Engineering Research? An Ongoing Study. In *2017 IEEE 25th International Requirements Engineering Conference (RE)*, pages 382–387. ISSN: 2332-6441. 1

[37] Franch, X., Fernández, D. M., Oriol, M., Vogelsang, A., Heldal, R., Knauss, E., Travassos, G. H., Carver, J. C., Dieste, O., and Zimmermann, T. (2017). How do practitioners perceive the relevance of requirements engineering research? an ongoing study. In *2017 IEEE 25th International Requirements Engineering Conference (RE)*, pages 382–387. 10

[38] Fronchetti, F., Wiese, I. S., Pinto, G., and Steinmacher, I. (2019). What attracts newcomers to onboard on oss projects? tl;dr: Popularity. In *OSS*. 33

[39] Garcia, H. V., Shihab, E., and Nagappan, M. (2018). Characterizing and predicting blocking bugs in open source projects. *Journal of Systems and Software*, 143:44–58. 33

[40] Gary, K., Koehnemann, H., Blakley, J., Goar, C., Mann, H., and Kagan, A. (2009). A case study: Open source community and the commercial enterprise. In *2009 Sixth International Conference on Information Technology: New Generations*, pages 940–945, 10.1109/ITNG.2009.313. IEEE. 22. 20, 21

[41] Gokhale, S. S., Smith, T., and McCartney, R. (2012). Integrating open source software into software engineering curriculum: Challenges in selecting projects. In *2012 First International Workshop on Software Engineering Education Based on Real-World Experiences (EduRex)*, pages 9–12, Piscataway, NJ, USA. IEEE Press. 32. 20

[42] Google (2019). About google scholar. Accessed: 2019-09-21. 30

[43] Grant, M. J. and Booth, A. (2009). A typology of reviews: an analysis of 14 review types and associated methodologies. *Health Information & Libraries Journal*, 26(2):91–108. 13

[44] Halkidi, M., Spinellis, D., Tsatsaronis, G., and Vazirgiannis, M. (2011). Data mining in software engineering. *Intelligent Data Analysis Journal (IDA)*, 15:413–441. 6

[45] Hawthorne, M. J. and Perry, D. E. (2005). Software engineering education in the era of outsourcing, distributed development, and open source software: challenges and opportunities. In *Proceedings. 27th International Conference on Software Engineering, 2005. ICSE 2005.*, pages 643–644, Berlin, Heidelberg. Springer-Verlag. 36. 20, 23

[46] He, Z., Yan, H., and Liu, C. (2013). Improvement on ABDOM-qd and its application in open-source community software defect discovery process. In *2013 Fourth World Congress on Software Engineering*, pages 84–90, 10.1109/WCSE.2013.18. IEEE. 11. 20, 21

[47] Hedberg, H. and Iivari, N. (2009). Integrating hci specialists into open source software development projects. In *OSS*. 8

[48] Hjelsvold, R. and Mishra, D. (2019). Exploring and expanding gse education with open source software development. *TOCE*, 19:12:1–12:23. 33

[49] Höst, M. and Oručević-Alagić, A. (2011). A systematic review of research on open source software in commercial software product development. *Inf. Softw. Technol.*, 53(6):616–624. 27. 20

[50] Iaffaldano, G., Steinmacher, I., Calefato, F., Gerosa, M. A., and Lanubile, F. (2019). Why do developers take breaks from contributing to oss projects? a preliminary analysis. In *SoHeal@ICSE*. 8

[51] Jokonya, O. (2015). Investigating open source software benefits in public sector. *2015 48th Hawaii International Conference on System Sciences*, pages 2242–2251. 33

[52] Joode, R. v. W. d. and Bruijne, M. d. (2006). The organization of open source communities: Towards a framework to analyze the relationship between openness and reliability. In *Proceedings of the 39th Annual Hawaii International Conference on System Sciences (HICSS'06)*, volume 6, pages 118b–118b. IEEE. 3. 20, 22

[53] Käfer, V., Graziotin, D., Bogicevic, I., Wagner, S., and Ramadani, J. (2018). Poster: Communication in open-source projects–end of the e-mail era? *2018 IEEE/ACM 40th International Conference on Software Engineering: Companion (ICSE-Companion)*, pages 242–243. 8

[54] Kazman, R., Goldenson, D., Monarch, I., Nichols, W., and Valetto, G. (2016). Evaluating the effects of architectural documentation: A case study of a large scale open source project. *IEEE Transactions on Software Engineering*, 42(3):220–260. 7. 20

[55] Kenett, R. S., Franch, X., Susi, A., and Galanis, N. (2014). Adoption of free libre open source software (floss): A risk management perspective. *2014 IEEE 38th Annual Computer Software and Applications Conference*, pages 171–180. 33

[56] Lampropoulos, A., Ampatzoglou, A., Bibi, S., Chatzigeorgiou, A., and Stamelos, I. (2018). React - a process for improving open-source software reuse. *2018 11th International Conference on the Quality of Information and Communications Technology (QUATIC)*, pages 251–254. 33

[57] Laplante, P., Gold, A., and Costello, T. (2007). Open source software: Is it worth converting? *IT Professional*, 9(4):28–33. 24. 20, 23

[58] Leban, G. (2013). Semantic tools for improving software development in open source communities. In *Proceedings of the 12th International Semantic Web Conference (Posters & Demonstrations Track) - Volume 1035*, ISWC-PD '13, pages 97–100, Aachen, Germany, Germany. CEUR-WS.org. 13. 20

[59] Liu, C. (2005). Enriching software engineering courses with service-learning projects and the open-source approach. In *Proceedings. 27th International Conference on Software Engineering, 2005. ICSE 2005.*, pages 613–614, New York, NY, USA. ACM. 35. 20, 23

[60] Lo, D., Nagappan, N., and Zimmermann, T. (2015a). How Practitioners Perceive the Relevance of Software Engineering Research. In *Proceedings of the 2015 10th Joint Meeting on Foundations of Software Engineering*, ESEC/FSE 2015, pages 415–425, New York, NY, USA. ACM. event-place: Bergamo, Italy. 1

[61] Lo, D., Nagappan, N., and Zimmermann, T. (2015b). How practitioners perceive the relevance of software engineering research. *10th Foundations of Soft.* 10, 12

[62] Lo, D., Nagappan, N., and Zimmermann, T. (2015c). How practitioners perceive the relevance of software engineering research. In *Proceedings of the 2015 10th Joint Meeting on Foundations of Software Engineering*, pages 415–425. ACM. 24, 34

[63] Marsan, J., Templier, M., Marois, P., Adams, B., Carillo, K., and Mopenza, G. L. (2019). Toward solving social and technical problems in open source software ecosystems: Using cause-and-effect analysis to disentangle the causes of complex problems. *IEEE Software*, 36:34–41. 9

[64] Masuda, A., Matsuodani, T., and Tsuda, K. (2019). Team activities measurement method for open source software development using the gini coefficient. *2019 IEEE International Conference on Software Testing, Verification and Validation Workshops (ICSTW)*, pages 140–147. 33

[65] Meyer, A. N., Fritz, T., Murphy, G. C., and Zimmermann, T. (2014). Software developers' perceptions of productivity. In *SIGSOFT FSE*. 11

[66] Milewicz, R., Pinto, G., and Rodeghero, P. (2019). Characterizing the roles of contributors in open-source scientific software projects. In *Proceedings of the 16th International Conference on Mining Software Repositories*, MSR '19, pages 421–432, Piscataway, NJ, USA. IEEE Press. 33

[67] Morgan, L. and Finnegan, P. (2014). Beyond free software: An exploration of the business value of strategic open source. *J. Strategic Inf. Sys.*, 23:226–238. 33

[68] Ostrowski, J. (2019). 20 Years of Open Source. 1

[69] Pandey, R. K. and Tiwari, V. (2011). Reliability issues in open source software. 7, 8

[70] Papadopoulos, P. M., Stamelos, I. G., and Cerone, A. (2014). Using open source projects in higher education: A two-way certification framework. In *Information Technology and Open Source: Applications for Education, Innovation, and Sustainability*, Lecture Notes in Computer Science, pages 274–280, 10.1007/978-3-642-54338-8_22. Springer Berlin Heidelberg. 28. 20, 23

[71] Petersen, K., Feldt, R., Mujtaba, S., and Mattsson, M. (2008). Systematic mapping studies in software engineering. In *Proceedings of the 12th International Conference on Evaluation and Assessment in Software Engineering*, EASE'08, pages 68–77, Swindon, UK. BCS Learning & Development Ltd. 13

[72] Pinto, G., Ferreira, C., Souza, C., Steinmacher, I., and Meirelles, P. (2019). Training software engineers using open-source software: The students' perspective. In *2019 IEEE/ACM 41st International Conference on Software Engineering: Software Engineering Education and Training (ICSE-SEET)*, pages 147–157. 33

[73] Pinto, G. H. L., Filho, F. F., Steinmacher, I., and Gerosa, M. A. (2017). Training software engineers using open-source software: The professors' perspective. In *2017 IEEE 30th Conference on Software Engineering Education and Training (CSEE T)*, pages 117–121. IEEE. 31. 20

[74] Poba-Nzaou, P., Uwizeyemungu, S., and Saada, M. (2019). Critical barriers to business intelligence open source software adoption. *IJBIR*, 10:59–79. 33

[75] Pressman, R. S. (2011). *Engenharia de software: uma abordagem profissional*. Porto Alegre (Rs): Amgh. 5

[76] Riehle, D., Riemer, P., Kolassa, C., and Schmidt, M. F. (2014). Paid vs. volunteer work in open source. *2014 47th Hawaii International Conference on System Sciences*, pages 3286–3295. 8

[77] Robles, G., González-Barahona, J. M., Cervigón, C., Capiluppi, A., and Izquierdo-Cortázar, D. (2014). Estimating development effort in free/open source software projects by mining software repositories: A case study of OpenStack. In *Proceedings of the 11th Working Conference on Mining Software Repositories*, MSR 2014, pages 222–231, New York, NY, USA. ACM. 20, 22

[78] Rudzki, J., Kiviluoma, K., Poikonen, T., and Hammouda, I. (2009). Evaluating quality of open source components for reuse-intensive commercial solutions. In *2009 35th Euromicro Conference on Software Engineering and Advanced Applications*, pages 11–19, Patras, Greece. IEEE. 14. 20, 21

[79] Russo, D. (2016). Benefits of open source software in defense environments. 33

[80] Schroeder, W. J., Ibanez, L., and Martin, K. M. (2004). Software process: the key to developing robust, reusable and maintainable open-source software. In *2004 2nd IEEE International Symposium on Biomedical Imaging: Nano to Macro (IEEE Cat No. 04EX821)*, pages 648–651 Vol. 1, Arlington, VA, USA, USA. IEEE. 12. 20, 21

[81] Schuwer, R., Genuchten, M. v., and Hatton, L. (2015). On the impact of being open. *IEEE Software*, 32(5):81–83. 21. 20, 23

[82] Siena, A., Morandini, M., and Susi, A. (2014). Modelling risks in open source software component selection. In *ER*. 33

[83] Sommerville, I. (2007). *Engenharia de software*. Pearson. 5

[84] Soto, M. and Ciolkowski, M. (2009). The QualOSS open source assessment model measuring the performance of open source communities. In *2009 3rd International Symposium on Empirical Software Engineering and Measurement*, pages 498–501, Lake Buena Vista, FL, USA. IEEE. 16. 20, 22

[85] Steinmacher, I., Conte, T., Treude, C., and Gerosa, M. A. (2016). Overcoming open source project entry barriers with a portal for newcomers. *2016 IEEE/ACM 38th International Conference on Software Engineering (ICSE)*, pages 273–284. 33

[86] Steinmacher, I., Gerosa, M. A., Conte, T., and Redmiles, D. F. (2019a). Overcoming social barriers when contributing to open source software projects. *Computer Supported Cooperative Work (CSCW)*, 28:247–290. 9

[87] Steinmacher, I., Treude, C., and Gerosa, M. A. (2019b). Let me in: Guidelines for the successful onboarding of newcomers to open source projects. *IEEE Software*, 36:41–49. 33

[88] Stol, K. and Fitzgerald, B. (2015). Inner source–adopting open source development practices in organizations: A tutorial. *IEEE Software*, 32(4):60–67. 23. 20, 23

[89] Syed Mohamad, S. M. and McBride, T. (2008). Reliability growth of open source software using defect analysis. In *2008 International Conference on Computer Science and Software Engineering*, volume 2, pages 662–667, Hubei, China. IEEE. 25. 20, 22

[90] Tafliovich, A., Estrada, F., and Caswell, T. (2019). Teaching software engineering with free open source software development: An experience report. 33

[91] Tamura, Y. and Yamada, S. (2016). Comparison of big data analyses for reliable open source software. In *2016 IEEE International Conference on Industrial Engineering and Engineering Management (IEEM)*, pages 1345–1349, Bali, Indonesia. IEEE. 5. 20

[92] Tomayko, J. E. (2000). A historian's view of software engineering. In *Thirteenth Conference on Software Engineering Education and Training, 6-8 March, 2000, Austin, Texas, USA*, page 101. 6

[93] Toth, K. (2006). Experiences with open source software engineering tools. *IEEE Software*, 23(6):44–52. 34. 20, 23

[94] Vetrò, A., Ognawala, S., Fernández, D. M., and Wagner, S. (2015). Fast Feedback Cycles in Empirical Software Engineering Research. In *2015 IEEE/ACM 37th IEEE International Conference on Software Engineering*, volume 2, pages 583–586. ISSN: 0270-5257, 1558-1225. 6

[95] Wang, X., Kuzmickaja, I., Stol, K., Abrahamsson, P., and Fitzgerald, B. (2014). Microblogging in open source software development: The case of drupal and twitter. *IEEE Software*, 31(4):72–80. 19, 20

[96] Wang, Y., Guo, D., and Shi, H. (2007). Measuring the evolution of open source software systems with their communities. *SIGSOFT Softw. Eng. Notes*, 32(6):1–12. 20

[97] Wazlawick, R. (2013). *Engenharia De Software*. Elsevier Editora Ltda. 5

[98] Weber, S. and Luo, J. (2014). What makes an open source code popular on git hub? In *2014 IEEE International Conference on Data Mining Workshop*, pages 851–855, Shenzhen, China. IEEE. 4. 20

[99] Xia, X., Wan, Z., Kochhar, P. S., and Lo, D. (2019). How practitioners perceive coding proficiency. *2019 IEEE/ACM 41st International Conference on Software Engineering (ICSE)*, pages 924–935. 10, 33

[100] Ye, Y. and Kishida, K. (2003). Toward an understanding of the motivation open source software developers. In *Proceedings of the 25th International Conference on Software Engineering*, ICSE '03, pages 419–429, Washington, DC, USA. IEEE Computer Society. 20, 22