PGCOMP - Programa de Pós-Graduação em Ciência da Computação
Universidade Federal da Bahia (UFBA)
Av. Adhemar de Barros, s/n - Ondina
Salvador, BA, Brasil, 40170-110

http://pgcomp.dcc.ufba.br
pgcomp@ufba.br

Projetos de software livre/código aberto (FLOSS) têm sido usados na Educação em Engenharia de Software (EES) como estratégia viável para trabalhar habilidades sócio-técnicas na formação de estudantes para atuar na indústria de software. Estudos relatam que o uso pedagógico de projetos FLOSS traz benefícios mas também desafios que podem comprometer a sua adoção efetiva no contexto educacional. Um desafio frequente é o da seleção de projetos. Em geral, o professor deve buscar projetos para uso pedagógico na disciplina manualmente, ou com apoio de diferentes ferramentas (não necessariamente projetadas para uso na Educação). Critérios válidos para a seleção de projetos FLOSS para EES incluem: tamanho do projeto, quantidade de testes, linguagem de programação, tamanho da comunidade e práticas utilizadas. A seleção de projetos FLOSS para uso pedagógico exige tempo e experiência do professor, e ainda há pouco suporte para apoiá-la com base em critérios reconhecidos e/ou definidos pelo professor. Assim, o ônus da seleção de projetos FLOSS pode atrapalhar o seu uso em disciplinas de Engenharia do Software (ES) . O objetivo desta pesquisa foi investigar a utilidade de uma abordagem para apoiar o professor na seleção automatizada e guiada por critérios sócio-técnicos, de projetos FLOSS para uso pedagógico na EES. Uma revisão da literatura identificou diversos tipos de critérios usados na seleção de projetos FLOSS. Um subconjunto de critérios de seleção foi escolhido, documentado, operacionalizado e implementado na ferramenta FlossSearch.Edu. A ferramenta foi avaliada por meio de dois estudos. No primeiro, estudantes de uma turma de ES usaram FlossSearch.Edu para selecionar projetos FLOSS, com base em critérios definidos pelo professor e, em seguida, avaliaram seu uso por meio de um instrumento baseado no Modelo de Aceitação de Tecnologia (TAM). No segundo estudo, de natureza quali-quantitativa, professores de diferentes instituições brasileiras de ensino superior, familiarizados com o uso pedagógico de projetos FLOSS, usaram FlossSearch.Edu em sessão individual e guiada por cenários pré-definidos. Cada professor relatou suas impressões em voz alta (método ``Think Aloud") e, ao final da sessão gravada, também avaliaram seu uso por meio de um questionário TAM. Em sua maioria, estudantes que usaram FlossSearch.Edu em sala de aula e professores que participaram do segundo estudo, avaliaram a ferramenta como útil, fácil de usar, com pretensão de usá-la no futuro. Diversas sugestões para melhoria foram recebidas e deverão guiar a evolução da ferramenta e novos estudos.

Keywords: Educação em Engenharia de Software, Software de Código Aberto, Software Livre, FLOSS, Critérios de Seleção de Projetos.

MSC | 112 | 2021

Seleção de Projetos de Código Aberto para Educação em Engenharia de Software

Moara Brito

UFBA

Seleção de Projetos de Código Aberto para Educação em Engenharia de Software

Moara Sousa Brito

Dissertação de Mestrado

Universidade Federal da Bahia

Programa de Pós-Graduação em Ciência da Computação

Fevereiro | 2021

PGCOMP - Programa de Pós-Graduação em Ciência da Computação
Universidade Federal da Bahia (UFBA)
Av. Adhemar de Barros, s/n - Ondina
Salvador, BA, Brasil, 40170-110

http://pgcomp.dcc.ufba.br
pgcomp@ufba.br

Projetos de software livre/código aberto (FLOSS) têm sido usados na Educação em Engenharia de Software (EES) como estratégia viável para trabalhar habilidades sócio-técnicas na formação de estudantes para atuar na indústria de software. Estudos relatam que o uso pedagógico de projetos FLOSS traz benefícios mas também desafios que podem comprometer a sua adoção efetiva no contexto educacional. Um desafio frequente é o da seleção de projetos. Em geral, o professor deve buscar projetos para uso pedagógico na disciplina manualmente, ou com apoio de diferentes ferramentas (não necessariamente projetadas para uso na Educação). Critérios válidos para a seleção de projetos FLOSS para EES incluem: tamanho do projeto, quantidade de testes, linguagem de programação, tamanho da comunidade e práticas utilizadas. A seleção de projetos FLOSS para uso pedagógico exige tempo e experiência do professor, e ainda há pouco suporte para apoiá-la com base em critérios reconhecidos e/ou definidos pelo professor. Assim, o ônus da seleção de projetos FLOSS pode atrapalhar o seu uso em disciplinas de Engenharia do Software (ES) . O objetivo desta pesquisa foi investigar a utilidade de uma abordagem para apoiar o professor na seleção automatizada e guiada por critérios sócio-técnicos, de projetos FLOSS para uso pedagógico na EES. Uma revisão da literatura identificou diversos tipos de critérios usados na seleção de projetos FLOSS. Um subconjunto de critérios de seleção foi escolhido, documentado, operacionalizado e implementado na ferramenta FlossSearch.Edu. A ferramenta foi avaliada por meio de dois estudos. No primeiro, estudantes de uma turma de ES usaram FlossSearch.Edu para selecionar projetos FLOSS, com base em critérios definidos pelo professor e, em seguida, avaliaram seu uso por meio de um instrumento baseado no Modelo de Aceitação de Tecnologia (TAM). No segundo estudo, de natureza quali-quantitativa, professores de diferentes instituições brasileiras de ensino superior, familiarizados com o uso pedagógico de projetos FLOSS, usaram FlossSearch.Edu em sessão individual e guiada por cenários pré-definidos. Cada professor relatou suas impressões em voz alta (método ``Think Aloud") e, ao final da sessão gravada, também avaliaram seu uso por meio de um questionário TAM. Em sua maioria, estudantes que usaram FlossSearch.Edu em sala de aula e professores que participaram do segundo estudo, avaliaram a ferramenta como útil, fácil de usar, com pretensão de usá-la no futuro. Diversas sugestões para melhoria foram recebidas e deverão guiar a evolução da ferramenta e novos estudos.

Keywords: Educação em Engenharia de Software, Software de Código Aberto, Software Livre, FLOSS, Critérios de Seleção de Projetos.

MSC | 112 | 2021

Seleção de Projetos de Código Aberto para Educação em Engenharia de Software
Moara Brito

UFBA

Seleção de Projetos de Código Aberto para Educação em Engenharia de Software

Moara Sousa Brito

Dissertação de Mestrado
nº 112 / 2021

Universidade Federal da Bahia

Programa de Pós-Graduação em
Ciência da Computação

Fevereiro | 2021

UNIVERSIDADE FEDERAL DA BAHIA
INSTITUTO DE MATEMÁTICA E ESTATÍSTICA

PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

SELEÇÃO DE PROJETOS DE CÓDIGO ABERTO PARA EDUCAÇÃO EM ENGENHARIA DE SOFTWARE

Moara Sousa Brito Lessa

DISSERTAÇÃO DE MESTRADO

Salvador, Ba
05 de Fevereiro de 2021

UNIVERSIDADE FEDERAL DA BAHIA
INSTITUTO DE MATEMÁTICA E ESTATÍSTICA

Moara Sousa Brito Lessa

SELEÇÃO DE PROJETOS DE CÓDIGO ABERTO PARA EDUCAÇÃO EM ENGENHARIA DE SOFTWARE

Trabalho apresentado ao PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO do INSTITUTO DE MATEMÁTICA E ESTATÍSTICA da UNIVERSIDADE FEDERAL DA BAHIA como requisito parcial para obtenção do grau de Mestre em CIÊNCIA DA COMPUTAÇÃO.

Orientadora: Christina von Flach Garcia Chavez

Salvador, Ba
05 de Fevereiro de 2021

"*Seleção de Projetos de Código Aberto para Educação em Engenharia de Software*"

Moara Sousa Brito

Dissertação apresentada ao Colegiado do Programa de Pós-Graduação em Ciência da Computação na Universidade Federal da Bahia, como requisito parcial para obtenção do Título de Mestre em Ciência da Computação.

Banca Examinadora

Profa. Dra. Christina von Flach Garcia Chavez (Orientadora-UFBA)

Prof. Dr. Rodrigo Rocha Gomes e Souza (UFBA)

Prof. Dr. Igor Fabio Steinmacher (UTFPR)

E de repente tudo dá certo, tudo acontece. Não é sorte. Nem acaso. É Deus. É o tempo de Deus. Com gratidão, dedico este trabalho a Ele.

AGRADECIMENTOS

Primeiramente agradeço a Deus, pela dádiva da vida e por me permitir realizar tantos sonhos. Obrigado por me permitir errar, aprender e crescer, por Seu infinito amor, pela Sua voz "invisível" que não me permitiu desistir e principalmente por fazer com que eu entendesse que o que for pra ser, Ele faz no tempo certo.

Agradeço à minha família, pelo contínuo apoio em todos esses anos, por acreditarem em mim, e não medirem esforços para a concretização dos meus sonhos. Ao meu esposo Breno Lessa, por ter caminhado ao meu lado, pela sua paciência, compreensão e ajuda prestada durante todo o mestrado. E claro por ter me ajudado no processo de construção da ferramenta.

Deus sabe qual personagem e em qual momento deve-se entrar nessa peça da vida. Ciente disso, agradeço a todos os amigos e colegas que de uma forma direta ou indireta, contribuíram, ou auxiliaram na elaboração do presente estudo, pela paciência, atenção e força que prestaram em momentos menos fáceis. Agradeço a meu amigo Juvenal e minha amiga Rose, que dividiram comigo as dificuldades e alegrias do Mestrado nos anos que estive em Salvador. Também agradeço a Fernanda Gomes, pelos artigos escritos em conjunto e, principalmente, pela preocupação e apoio constante.

A Prof. Christina von Flach, pela orientação, competência, profissionalismo e dedicação tão importantes. Obrigada por me acolher e acreditar em mim. Tenho certeza que não chegaria neste ponto sem o seu apoio. Você é sem dúvida um ser humano incrível e melhor orientadora que alguém poderia ter. Aqui manifesto a minha gratidão.

Por fim, o meu profundo agradecimento a todas as pessoas que contribuíram para a concretização desta dissertação, estimulando-me intelectual e emocionalmente.

RESUMO

Contexto: Projetos de software livre/código aberto (FLOSS) têm sido usados na Educação em Engenharia de Software (EES) como estratégia viável para trabalhar habilidades sócio-técnicas na formação de estudantes para atuar na indústria de software. Estudos relatam que o uso pedagógico de projetos FLOSS traz benefícios mas também desafios que podem comprometer a sua adoção efetiva no contexto educacional. Um desafio frequente é o da seleção de projetos. Em geral, o professor deve buscar projetos para uso pedagógico na disciplina manualmente, ou com apoio de diferentes ferramentas (não necessariamente projetadas para uso na Educação). Critérios válidos para a seleção de projetos FLOSS para EES incluem: tamanho do projeto, quantidade de testes, linguagem de programação, tamanho da comunidade e práticas utilizadas.
Problema: A seleção de projetos FLOSS para uso pedagógico exige tempo e experiência do professor, e ainda há pouco suporte para apoiá-la com base em critérios reconhecidos e/ou definidos pelo professor. Assim, o ônus da seleção de projetos FLOSS pode atrapalhar o seu uso em disciplinas de Engenharia do Software (ES).
Objetivo: O objetivo desta pesquisa foi investigar a utilidade de uma abordagem para apoiar o professor na seleção automatizada e guiada por critérios sócio-técnicos, de projetos FLOSS para uso pedagógico na EES.
Métodos de Pesquisa: Uma revisão da literatura identificou diversos tipos de critérios usados na seleção de projetos FLOSS. Um subconjunto de critérios de seleção foi escolhido, documentado, operacionalizado e implementado na ferramenta FlossSearch.Edu. A ferramenta foi avaliada por meio de dois estudos. No primeiro, estudantes de uma turma de ES usaram FlossSearch.Edu para selecionar projetos FLOSS, com base em critérios definidos pelo professor e, em seguida, avaliaram seu uso por meio de um instrumento baseado no Modelo de Aceitação de Tecnologia (TAM). No segundo estudo, de natureza quali-quantitativa, professores de diferentes instituições brasileiras de ensino superior, familiarizados com o uso pedagógico de projetos FLOSS, usaram FlossSearch.Edu em sessão individual e guiada por cenários pré-definidos. Cada professor relatou suas impressões em voz alta (método "Think Aloud") e, ao final da sessão gravada, também avaliaram seu uso por meio de um questionário TAM.
Resultados: Em sua maioria, estudantes que usaram FlossSearch.Edu em sala de aula e professores que participaram do segundo estudo, avaliaram a ferramenta como útil, fácil de usar, com pretensão de usá-la no futuro. Diversas sugestões para melhoria foram recebidas e deverão guiar a evolução da ferramenta e novos estudos.

Palavras-chave: Educação em Engenharia de Software, Software de Código Aberto, Software Livre, FLOSS, Critérios de Seleção de Projetos.

ABSTRACT

Context: Free/open source software (FLOSS) has been used in Software Engineering Education (SEE) as a viable strategy to address the need for developing students' technical and non-technical skills and to prepare them to face the increasingly challenging software industry. Some studies report that the pedagogical use of FLOSS projects has benefits but also challenges that may hinder their effective adoption in SEE.
One of such challenges is the selection of FLOSS projects. In general, the teacher must search for one or more projects for pedagogical use in the course, manually or with the support of different tools (not necessarily designed for use in Education). Some criteria for selecting a FLOSS project for pedagogical use are: project size, number of tests, programming language, community size and their practices.
Problem: The selection of FLOSS projects for pedagogical use in SEE requires effort and experience, and there are few tools that support such complex task while providing clear and well-defined selection criteria to be used by the teacher. Thus, the burden of selecting FLOSS projects can hinder their use and adoption in the context of SEE.
Objective: This research aimed to investigate the usefulness of an automated approach to support the teacher in the selection of FLOSS projects, guided by criteria that take into account their socio-technical aspects, for pedagogical use in SEE higher education.
Reasearch Method: A literature review provided a set of criteria that have been used for the selection of FLOSS projects. Then, a subset of such selection criteria was documented, operationalized and implemented in an open source tool called FlossSearch.Edu. The tool was evaluated by means of two empirical studies. In the first study, undergraduate students used FlossSearch.Edu, in the context of a software engineering 1-semester course, to select FLOSS projects based on criteria defined by the teacher, and then evaluated the use of the tool through a survey with questions based on the Technology Acceptance Model (TAM). In the second study, teachers from different Brazilian higher education institutions, already familiar with the pedagogical use of FLOSS projects, used FlossSearch.Edu in an individual session, guided by predefined scenarios, while reporting their impressions out loud ("Think Aloud"protocol). At the end of the recorded sessions, they evaluated the use of the tool by means of a TAM questionnaire.
Findings: Most of the students who used FlossSearch.Edu in the classroom and teachers who participated in the second study, evaluated the tool as useful and easy to use, and intended to use it in the future. They provided suggestions for improvement that should guide tool evolution and future studies.

Keywords: Software Engineering Education, Open Source Software, Free Software, FLOSS, Project Selection Criteria.

SUMÁRIO

LISTA DE FIGURAS

LISTA DE TABELAS

Capítulo 1

INTRODUÇÃO

Sistemas de software são parte fundamental de praticamente todas as áreas do conhecimento, tipos de indústria e tipos de negócios da Sociedade. Nesse contexto, a Educação em Engenharia de Software (EES)[1] desempenha um papel extremamente importante, dada a crescente demanda por profissionais bem qualificados para atuar na indústria de software (STATISTICS, 2019).

A Educação em Engenharia de Software tem como objetivo promover a formação de profissionais com capacidade de atuar no desenvolvimento e evolução de sistemas de software (CSEE&T, 2016). Tal formação deve prover uma base teórica sólida na área de Engenharia de Software (ES), seus princípios e conceitos, atividades básicas (e.g, requisitos, projeto, testes, gerência de configuração e de lançamentos), práticas e ferramentas, e em áreas relacionadas, como gerência de projetos e interação humano-computador. Experiência prática, habilidades de comunicação, profissionalismo e educação continuada são dimensões que também devem ser consideradas (MISHRA; CAGILTAY; KILIC, 2007), incluindo capacidade de resolução de problemas, trabalho em equipe e ética, de modo a preparar os estudantes para a carreira na indústria de software (GUTICA, 2018). Nesse contexto, currículos de referência têm sido definidos por educadores e profissionais da área para guiar a criação de disciplinas e cursos de graduação em ES (CURRICULA, 2004, 2015) e estratégias pedagógicas têm sido criadas ou adaptadas para o ensino-aprendizagem de ES (e.g. *gamificação* (LELLI et al., 2020) e aprendizagem baseada em projetos (KOLMOS, 1996)).

Entretanto, apesar do amadurecimento da área de ES (BOURQUE; FAIRLEY, 2014) e da preocupação com a formação de profissionais (CURRICULA, 2015; ADCOCK et al., 2009), há uma insatisfação recorrente que permeia a indústria de software em relação à capacitação dos recém-egressos de cursos de nível superior para enfrentar desafios de uma área em constante evolução (PARNAS, 1999), (MEYER, 2001), (RADERMACHER; WALIA, 2013), (GAROUSI et al., 2019a).

[1] *Software Engineering Education (SEE)*

Radermacher e Walia (2013) entrevistaram gerentes e responsáveis por recrutamento de pessoal para compreender as deficiências de profissionais recém-graduados que atuavam em suas empresas de software. As principais deficiências reportadas estão relacionadas ao (uso de ferramentas de) gerenciamento de configuração, testes e habilidades de comunicação e recomendam que educadores ofereçam aos estudantes algum tipo de experiência com projetos do "mundo real". Garousi et al. (2019a) realizaram uma revisão sistemática da literatura para identificar as habilidades mais importantes para a indústria de software e sintetizar evidências sobre deficiências de formação encontradas em estudantes recém-egressos de cursos de engenharia de software. Eles recomendam que projetos de maior complexidade e tamanho, que explorem aspectos da prática profissional, sejam desenvolvidos por equipes estudantes (com ou sem parceira de empresas).

Deficiências na dimensão de *experiência prática* (MISHRA; CAGILTAY; KILIC, 2007), (RADERMACHER; WALIA, 2013) ou na *prática profissional* (GAROUSI et al., 2019a) incluem profissionalismo, trabalho em grupo e habilidades de comunicação, e estão associadas à falta de contato de estudantes com software da indústria. Diferentes estratégias para promover o contato com projetos de software e práticas da indústria têm sido utilizadas (BISHOP et al., 2016), por exemplo, *hackathons* (STEGLICH et al., 2020), cooperação com a indústria (KNUDSON; RADERMACHER, 2009; REICHLMAY, 2006) e projetos de software de código aberto ou FLOSS[2] (NASCIMENTO, 2017).

Atualmente, a indústria de software usa e depende de diversos projetos de software código aberto. Projetos FLOSS possuem características similares aos projetos da indústria de software e pouco frequentes na EES, incluindo, software de médio e grande porte em evolução, com arquitetura complexa, requisitos reais, uso de práticas e ferramentas modernas, desenvolvedores com diferentes habilidades e níveis de experiência que fazem parte de uma comunidade que demanda e implementa, colaborativamente, correções, modificações ou melhorias (NASCIMENTO, 2017). O uso de projetos FLOSS na EES permite a aproximação entre a realidade da construção de software e o ensino-aprendizagem realizado no ambiente acadêmico (NASCIMENTO; BITTENCOURT; CHAVEZ, 2015), o que pode aumentar a confiança dos estudantes quando forem iniciar suas atividades profissionais (CHAVEZ et al., 2011a). Além disso, o ciclo de vida de projetos FLOSS extrapola a duração de um semestre letivo e um projeto pode ser utilizado em outros semestres ou em outras disciplinas, permitindo que os alunos trabalhem em atividades práticas com software desenvolvido por terceiros (NASCIMENTO; BITTENCOURT; CHAVEZ, 2015). Finalmente, a comunicação, a dinâmica social e colaborativa do trabalho em equipe em grandes projetos podem ser vivenciadas em projetos FLOSS (HORSTMANN, 2009).

Nascimento (2017) trouxe evidências de que o uso de projetos FLOSS em disciplinas de graduação é viável e permitiu que vários objetivos de aprendizagem fossem alcançados, mostrando-se promissor para algumas áreas de conhecimento da engenharia de software. Em virtude de sua proximidade com a realidade da indústria de software, o uso de projetos FLOSS, como software a ser trabalhado pelos estudantes, proporcionou um ambiente autêntico de aprendizagem. Trabalhos posteriores reportaram que o uso de

[2] Ao longo do texto, utilizaremos o termo *software de código aberto* como referência para *software livre e de código aberto*, tradução para *Free/Libre and Open Source Software* ou, simplesmente, *FLOSS*, o acrônimo para o termo em inglês.

projetos FLOSS na EES representou uma experiência apropriada do mundo real (NASCIMENTO; CHAVEZ; BITTENCOURT, 2018) e ajudou a diminuir a lacuna entre teoria e prática (NASCIMENTO; CHAVEZ; BITTENCOURT, 2019), (SILVA et al., 2019).

Pinto et al. (2017) e Ferreira et al. (2018a) relataram que, na perspectiva dos professores que participaram de seu estudos e no contexto de uso de projetos FLOSS em disciplinas de ES, os benefícios superam os desafios, pois além de habilidades técnicas, os estudantes também podem desenvolver habilidades sociais. Pinto et al. (2017) destacam que contribuições em projetos FLOSS podem compor um portfólio a ser utilizado em futuras oportunidades de emprego.

1.1 PROBLEMA

Estudos apontam que o uso de projetos de código aberto pode trazer benefícios para o ensino-aprendizagem de ES, mas também introduzir dificuldades e desafios para professores e estudantes. A seleção de um projeto de software de código aberto adequado para utilização na educação em engenharia de software é um dos desafios a serem enfrentados (NASCIMENTO, 2017).

O problema da seleção de projetos de código aberto foi destacado no painel *How to Use Open Source Software in Education*, realizado na 47a. edição do *ACM Technical Symposium on Computing Science Education* e coordenado por Bishop et al. (2016). Observou-se que, enquanto há várias formas de utilizar projetos FLOSS na Educação, a crescente diversidade de projetos disponíveis representa um obstáculo para educadores na seleção de projetos adequados para integração em uma disciplina.

Estudos que investigam o problema da seleção de projetos reportam que, em geral, a seleção de projetos é baseada em critérios definidos pelo professor (ELLIS; HISLOP; PURCELL, 2013), o procedimento de seleção é manual e considerado trabalhoso (ELLIS et al., 2011; GOKHALE; SMITH; MCCARTNEY, 2012), exigindo tempo e experiência do professor. Ellis, Purcell e Hislop (2012) pontuam que, com o número crescente de projetos de código aberto, com diferentes tamanhos, complexidades, domínios e comunidades, selecionar um projeto adequado para trabalhar com alunos não é uma tarefa trivial, sendo necessária, muitas vezes, algum tipo de apoio ou orientação para a seleção. Ellis, Hislop e Purcell (2013) destacam que a seleção de projetos FLOSS para a participação dos estudantes pode ser assustadora para os membros do corpo docente que não estiverem envolvidos em grandes projetos de software. Finalmente, Gokhale, Smith e McCartney (2012) consideram que as dificuldades relacionadas à seleção de projetos FLOSS podem desestimular outros professores que desejam adotar essa abordagem em suas disciplinas. Tal esforço na seleção de projetos FLOSS deve ser amenizado, de modo que não atrapalhe a adoção de projetos FLOSS na SEE (SMITH et al., 2014).

Nesta pesquisa de mestrado, investigamos se a organização de critérios de seleção de projetos e sua implementação em uma ferramenta para apoio automatizado à seleção de projetos de código aberto podem atenuar o problema apresentado.

1.2 OBJETIVOS

O *objetivo geral* desta pesquisa foi investigar o problema da seleção de projetos de código aberto para uso pedagógico na educação em engenharia de software e prover uma solução para apoiar o professor, de forma automática e sistemática, na tarefa de selecionar projetos para uso em disciplinas de engenharia de software.

Com base no objetivo geral definido e considerando o contexto de uso pedagógico de projetos de código aberto, são *objetivos específicos* desta pesquisa:

O1. Investigar critérios de seleção de projetos de código aberto, usos conhecidos e formas de organização.

A caracterização de critérios de seleção, usos conhecidos e formas de organização tem o propósito compreender e fundamentar o problema da seleção de projetos de código aberto para uso na EES e prover os alicerces para a construção da solução proposta.

O2. Investigar a viabilidade de operacionalizar a seleção de projetos de código aberto baseada em critérios.

Nesta pesquisa, o termo *operacionalizar* foi usado para denotar meios para alcançar objetivos, mapeando critérios de seleção de projetos de código aberto a características de projetos hospedados em repositórios públicos de software. A operacionalização de um critério de seleção inclui a documentação de informações que possam sistematizar e guiar seu uso na seleção de projetos, manual ou automatizada, com base em informações disponibilizadas por plataformas que hospedam ou indexam repositórios de projetos de código aberto.

O3. Investigar a utilidade de uma ferramenta de software para automatizar, com base em critérios definidos pelo professor, a seleção de projetos de código aberto hospedados em repositórios de software.

A implementação de uma ferramenta serve como prova de conceito para a seleção automatizada de projetos de código aberto hospedados em repositórios de software, mas a ferramenta deve ser útil para professores e estudantes no contexto da EES. Ela também pode servir de apoio para estudos que necessitam selecionar projetos de código aberto para uso em diferentes cenários.

1.3 QUESTÃO DE PESQUISA

Considerando os benefícios de utilização de projetos FLOSS na Educação em Engenharia de Software e a quantidade de projetos disponíveis em repositórios de software, a questão geral abordada por esta dissertação é:

> *Como selecionar projetos de código aberto adequados, de forma sistemática, em repositórios de software, para uso na Educação em Engenharia de Software?*

A questão geral de pesquisa é desdobrada em três questões de pesquisa específicas:

RQ1. Quais são os principais critérios utilizados na seleção de projetos de código aberto para o uso na EES?

A seleção de projetos de código aberto pode ser feita pelo professor que, utilizará um ou mais critérios para encontrar um projeto adequado para uso na disciplina. Esta questão de pesquisa está relacionada ao objetivo **O1**. Uma revisão da literatura pode trazer evidências para respondê-la.

RQ2. Quais são as informações presentes em repositórios de software que podem ser utilizadas para seleção de projetos de código aberto para uso na EES? Repositórios de software guardam diversos tipos de informação sobre projetos de código aberto (versões, linguagens de programação utilizadas, contribuidores, *commits*, etc.) que podem ser recuperadas e usadas para a seleção de projetos. Esta questão de pesquisa está relacionada ao objetivo **O2**.

RQ3. Quão útil pode ser uma ferramenta que automatiza, com base em critérios de seleção definidos pelo professor, a seleção de projetos de código aberto para uso na EES? A utilidade de uma ferramenta para seleção automática de projetos de código aberto para uso na EES deve ser avaliada sob a perspectiva de seus potenciais usuários, professores e alunos. Esta questão de pesquisa está relacionada ao objetivo **O3**.

1.4 VISÃO GERAL DA PESQUISA

Neste trabalho, foram utilizados diferentes métodos e técnicas para abordar o problema de pesquisa desta dissertação, alcançar os objetivos definidos e responder às questões de pesquisa. A Figura 1.1 apresenta uma visão geral da pesquisa, estruturada em etapas: (i) mapeamento sistemático da literatura, (ii) implementação de uma ferramenta, como prova de conceito, (iii) avaliação quali-quantitativa da ferramenta, por meio de *survey* desenhado segundo os critérios definidos pelo modelo TAM (Technology Acceptance Model) e de sessões baseadas no método *think aloud*. O detalhamento do processo de pesquisa encontra-se no Capítulo 3.

1.5 ESCOPO

O escopo desta pesquisa restringe-se ao problema da seleção de projetos de software de código aberto para uso pedagógico na EES, ou seja, como objeto de aprendizagem de atividades teóricas e práticas em uma abordagem baseada em projetos.

Neste trabalho, assumimos que a atividade de *seleção de Projetos FLOSS para uso pedagógico* é realizada em 4 passos: (1) definição (em geral, pelo professor) de características desejáveis para um ou mais projetos FLOSS a serem utilizados na disciplina, (2) filtragem de projetos com base em tais características, (3) análise de um subconjunto de projetos retornados, e (4) escolha de um ou mais projetos FLOSS para uso.

Foram considerados projetos de código aberto em repositórios de software, nomeadamente um subconjunto dos projetos hospedados na plataforma *GitHub*, um dos maiores

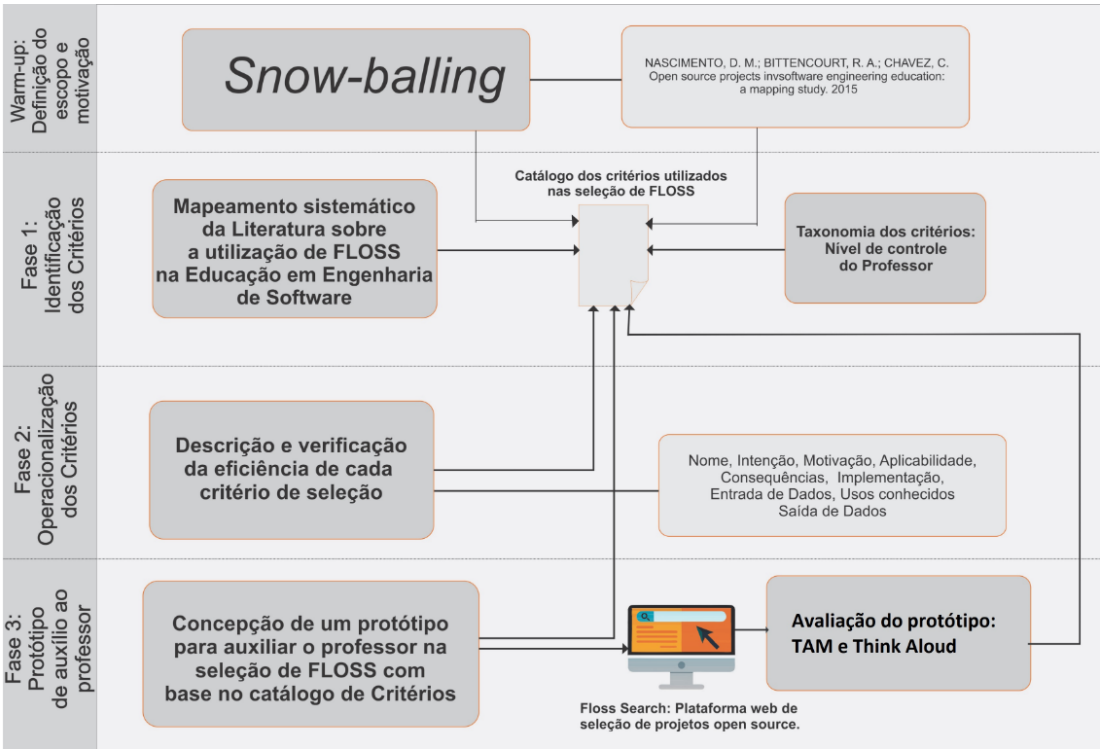

Figura 1.1 Desenho da Pesquisa.

repositórios de projetos de código aberto (CHATZIASIMIDIS; STAMELOS, 2015), contando com mais de cinco milhões de desenvolvedores e que, segundo Kalliamvakou et al. (2014), colaboram com mais de dez milhões de projetos. Entretanto, a abordagem proposta pode ser estendida para trabalhar com outras plataformas.

Há diversas áreas da Engenharia de Software em que os projetos FLOSS podem ser utilizados. Entretanto, a abordagem para seleção de projetos proposta não considera sub-áreas específicas como critérios (e.g. testes, manutenção, requisitos). Tal abordagem exigiria um tempo maior de pesquisa e análise mais minuciosa de objetivos pedagógicos da disciplina de SE, a fim de obter um outro conjunto de critérios. Portanto, este trabalho apresenta um conjunto preliminar de critérios que podem ser utilizados de forma genérica.

Projetos hospedados no *GitHub* possuem seu próprio repositório e suas informações podem ser recuperadas por meio uma API fornecida pelo *GitHub* (BAUTISTA; FELIU, 2015). Além da API do *Github*, este trabalho utiliza a API do *Black Duck Open Hub* para obter informações que não são disponibilizadas pelo *GitHub*.

1.6 CONTRIBUIÇÕES

No contexto de uso pedagógico de projetos de código aberto em disciplinas de ES de cursos de graduação, as principais contribuições desta pesquisa são:

- Apresentação de uma visão geral atualizada (BRITO et al., 2018) de como projetos de código aberto têm sido usados no processo de ensino-aprendizagem de ES;

- Identificação de critérios de seleção de projetos de código aberto (BRITO et al., 2018), a partir da revisão e atualização de um mapeamento sistemático anterior (NASCIMENTO; BITTENCOURT; CHAVEZ, 2015);
- Elaboração de catálogo de critérios para seleção de projetos de código aberto. Na literatura, artigos relatam como os projetos foram utilizados, mas não orientam sobre o uso sistemático de critérios para a seleção de projetos;
- Desenvolvimento da ferramenta `FlossSearch.edu` para apoiar professores e estudantes na seleção de projetos de código aberto para uso na EES.

No mesmo contexto, algumas contribuições indiretas incluem:

- Atenuar dificuldades associadas à utilização e eventual contribuição em projetos de código aberto;
- Incentivo à adoção de projetos de código aberto no contexto da educação em engenharia de software;
- Estímulo ao contato e contribuição de professores e estudantes nos projetos de código aberto estudados (engajamento com a comunidade e retribuição);
- Valorização e difusão da pesquisa que envolve software livre, educação em engenharia de software e sua combinação, por meio de publicações científicas e participação em eventos acadêmicos.

1.7 ARTIGOS PUBLICADOS

Esta pesquisa resultou em quatro publicações científicas[3]:

1. Moara Sousa Brito, Fernanda Gomes Silva, Christina von Flach G. Chavez, Debora Maria Coelho Nascimento, Roberto Almeida Bittencourt: FLOSS in software engineering education: an update of a systematic mapping study. SBES 2018 (Education Track): 250-259. DOI: ⟨https://doi.org/10.1145/3266237.3266249⟩. DBLP key: `conf/sbes/BritoSCNB18`.

   Este artigo apresenta uma atualização do mapeamento sistemático realizado em Nascimento, Bittencourt e Chavez (2015). Ele cobre o período de 2013 a 2017, classificando e resumindo os artigos incluídos para discutir tendências e identificar lacunas de pesquisa no contexto do uso de projetos FLOSS na SEE.

2. Fernanda Gomes Silva, Moara Sousa Brito, Jenifer Vieira Toledo Tavares, Christina von Flach G. Chavez: FLOSS in Software Engineering Education: Supporting the Instructor in the Quest for Providing Real Experience for Students. SBES 2019 (Education Track): 234-243. DOI: ⟨https://doi.org/10.1145/3350768.3353815⟩. DBPL key: `conf/sbes/SilvaBTC19`.

[3]Fonte: ⟨https://dblp.org/⟩

Este artigo relata uma experiência do ensino de modelagem orientada a objetos com diagramas de classes, usando projetos FLOSS. A equipe de pesquisa apoiou o professor em algumas atividades durante o planejamento de aulas, dentre elas, a seleção de projetos FLOSS para uso em sala de aula, e o desenho de atividades práticas de modelagem com o projeto FLOSS selecionado. A seleção do projeto FLOSS usou critérios categorizados como *"Inside Control"* (ver Capítulo 4).

3. Fernanda Gomes Silva, Moara Sousa Brito Lessa, Nádia da Luz Lopes, Christina von Flach G. Chavez: Teaching UML Models with FLOSS Projects: A study carried out during the period of social isolation imposed by the COVID-19 pandemic. SBES 2020 (Education Track): 483-492. DOI: ⟨https://doi.org/10.1145/3422392.3422491⟩. DBPL key: `conf/sbes/SilvaLLC20`.

   Este artigo apresenta um relato de experiência sobre o uso de projetos FLOSS no ensino de ES na graduação, com virtualização total das aulas pela necessidade de isolamento social de estudantes e professores. Projetos FLOSS foram usados em atividades práticas de modelagem orientada a objetos, para trabalhar conteúdos de diagramas de classe, de sequência e de atividades da UML. A turma foi dividida em grupos e cada grupo de alunos poderia selecionar um projeto FLOSS. O professor definiu as seguintes restrições para seleção do projeto: (i) projeto FLOSS implementado na linguagem Java, (ii)projeto FLOSS com mais de 10 *releases*, e (iii) projeto FLOSS com mais de 5.000 linhas de código. Para dar suporte à busca e seleção de projetos adequados aos critérios estabelecidos pelo professor, os estudantes utilizaram a ferramenta `FlossSearch.Edu`.

4. Moara Sousa Brito Lessa, Christina von Flach G. Chavez: An Approach for Selecting FLOSS Projects for Education. SBES 2020 (Education Track): 463-472. DOI: ⟨https://doi.org/10.1145/3422392.3422492⟩. DBPL key: `conf/sbes/LessaC20`.

   Este artigo apresentou a abordagem proposta para a seleção de projetos FLOSS com base em critérios sócio-técnicos, com avaliação a partir da perspectiva dos alunos. Eles usaram `FlossSearch.Edu` para selecionar projetos FLOSS com base em critérios definidos pelo professor e responderam a uma pesquisa baseada no Modelo de Aceitação de Tecnologia (TAM). Foi observado que a ferramenta desempenhou um papel importante no processo de seleção de projetos FLOSS. Em sua maioria, os alunos que usaram a ferramenta afirmaram que a mesma era útil, fácil de usar e que pretendiam usá-la no futuro.

1.8 PRÊMIOS E INDICAÇÕES

O pôster *Seleção de projetos de Código Aberto para Educação em Engenharia de Software* apresentou resultados iniciais desta pesquisa no Workshop de Estudantes do PGCOMP (WEPGCOMP 2018), evento anual do Programa de Pós-graduação em Ciência da Computação da UFBA, sendo premiado como "melhor apresentação de pôster" pelo público, e indicado como "segunda melhor apresentação de pôster" pela banca avaliadora composta por professores permanentes do PGCOMP.

1.9 ESTRUTURA DA DISSERTAÇÃO

Essa dissertação está organizada da seguinte forma. O Capítulo 2 apresenta conceitos sobre educação em engenharia de software, projetos de código aberto e seu uso pedagógico, e introduz o problema de seleção de projetos baseada em critérios. O Capítulo 3 apresenta os métodos de pesquisa utilizados ao longo deste trabalho. O Capítulo 4 apresenta os resultados obtidos na identificação e classificação de critérios de seleção para projetos de código aberto, bem como sua organização na forma de um catálogo de critérios de seleção. O Capítulo 5 apresenta a ferramenta `FlossSearch.edu`. O Capítulo 6 relata os resultados da avaliação da experiência de uso de professores e alunos em relação à abordagem desenvolvida. O Capítulo 7 apresenta uma discussão sobre a pesquisa realizada e o Capítulo 8 apresenta as conclusões e oportunidades de pesquisa para o futuro.

Capítulo 2

PROJETOS DE CÓDIGO ABERTO NA EDUCAÇÃO EM ENGENHARIA DE SOFTWARE

Neste capítulo, apresentamos uma breve fundamentação sobre a Educação em Engenharia de Software, projetos de código aberto, seu uso no processo de ensino-aprendizagem de engenharia de software e o problema da seleção de projetos para uso pedagógico em tal contexto.

2.1 EDUCAÇÃO EM ENGENHARIA DE SOFTWARE

A Engenharia de Software é a área do conhecimento que se preocupa com a aplicação de conhecimento teórico e prático na construção e evolução de sistemas de software (BOURQUE; FAIRLEY, 2014). A ES é considerada um campo da Ciência da Computação, complexo, multi-facetado e em rápida evolução, com oportunidades e desafios recorrentes (BOEHM, 2011), incluindo a educação e formação, em cursos de graduação, de profissionais qualificados para atuar no mercado de trabalho (CLEAR et al., 2016; REUTER; BESLMEISL; MOTTOK, 2017).

A Educação em Engenharia de Software tem como objetivo promover a formação de profissionais com capacidade de atuar no desenvolvimento e evolução de sistemas de software (CSEE&T, 2016). A partir da década de 60, quando o termo *engenharia de software* começou a ser utilizado, a ES passou a fazer parte, como matéria e/ou disciplina(s), dos currículos de cursos de Computação (SHACKELFORD et al., 2006) e afins. A partir do anos 80, surgiram os primeiros cursos de graduação em Engenharia de Software e, mais tarde, os primeiros currículos de referência (CURRICULA, 2004, 2015).

2.1.1 Currículos de Referência

Os currículos têm como objetivo prover orientação para as instituições acadêmicas sobre o que deve constituir os cursos, seus conteúdos e competências que devem ser desenvolvidas durante a graduação (NASCIMENTO, 2017). Diversas organizações que definem currículos de referência para cursos de Ciência da Computação (IEEE; ACM, 2013; SBC,

2005) e Engenharia de Software (CURRICULA, 2015) realçam a utilização de projetos realistas na EES.

O currículo de referência para Ciência da Computação da ACM/IEEE-CS[1], *Computer Science Curricula* 2013 - CS2013 (IEEE; ACM, 2013) apresenta recomendações curriculares para 18 áreas do conhecimento da Computação. Uma dessas áreas é a Engenharia de Software, definida como *a área do conhecimento que se preocupa com a aplicação de teoria, conhecimento e prática para construção eficaz e eficiente de sistemas de software confiáveis que satisfazem requisitos de clientes e usuários.*

A disciplina de Engenharia de Software deve abranger: todos os métodos de desenvolvimento de software; ciclo de vida de software; noções de requisitos; especificação, implementação de design; teste de software; avaliação de software e seus princípios; manutenção; e gerenciamento de desenvolvimento de software. Deve-se ainda preocupar com a qualidade em todas as fases do processo de desenvolvimento de software, como os benefícios e boas práticas na reutilização de software, e com a importância da atividade prática (CURRICULA, 2015).

Os alunos devem tomar consciência das diferenças entre técnicas e princípios gerais de engenharia de software e as técnicas e princípios necessários para abordar questões específicas de sistemas especializados. Os sistemas especializados incluem: sistemas de tempo real, sistemas cliente-servidor, sistemas distribuídos, sistemas paralelos, sistemas web, jogos e computação móvel. Sendo assim, os alunos devem aprender a aplicar os conceitos aprendidos em engenharia de software ao participar de um projeto. Tais projetos devem exigir que os alunos trabalhem em equipe para desenvolver um sistema de software ao longo da maior parte de seu ciclo de vida (IEEE; ACM, 2013).

A Figura 2.1 apresenta um resumo das 10 sub-áreas de Engenharia de Software identificadas no currículo da ACM/IEEE-CS (IEEE; ACM, 2013). Cada sub-área agrega um ou mais tópicos, identificados como “Core” ou “Eletivos”.

	Core-Tier1 horas	Core-Tier2 horas	Inclui Eletiva
SE/Processos de Software	2	1	Sim
SE/Gerenciamento de Projetos de Software		2	Sim
SE/Ferramentas e Ambientes		2	Não
SE/Engenharia de Requisitos	1	3	Sim
SE/Design de Software	3	5	Sim
SE/Construção de Software		2	Sim
SE/Verificação e Validação de Software		4	Sim
SE/Evolução do Software		2	Sim
SE/Confiabilidade de Software		1	Sim
SE/Métodos Formais			Sim

Figura 2.1 Engenharia de Software (27 horas de aula teórica) (IEEE; ACM, 2013)

[1] *Association for Computing Machinery* (ACM) e *Institute of Electrical and Electronics Engineers Computer Society* (IEEE-CS).

São tópicos da sub-área SE/Ferramentas e Ambientes [2 horas em tópicos Core-Tier2][2]: Gerenciamento de configuração de software e controle de versão; Gerenciamento de *Release*; Análise de requisitos e ferramentas de modelagem de *design*; Ferramentas de teste, incluindo ferramentas de análise estática e dinâmica; Ambientes de programação que automatizam partes de processos de construção de programas (por exemplo, compilações automatizadas); Integração contínua; Conceitos e mecanismos de integração de ferramentas.

Cada sub-área também possui resultados de aprendizagem associados, com um nível de domínio de conteúdo: familiaridade (*O que você sabe sobre isso?*), uso (*O que você sabe sobre como fazer isso?*), avaliação (*Por que você faria isso?*) (IEEE; ACM, 2013). Os resultados de aprendizagem esperados para a sub-área SE/Ferramentas e Ambientes são:

1. Descreva a diferença entre gerenciamento de configuração de software centralizado e distribuído. [Familiaridade]

2. Descreva como o controle de versão pode ser usado para ajudar no gerenciamento da *release* de software. [Familiaridade]

3. Identifique itens de configuração e use uma ferramenta de controle de código fonte em um pequeno projeto baseado em equipe. [Uso]

4. Descreva como as ferramentas de teste estáticas e dinâmicas disponíveis podem ser integradas no ambiente de desenvolvimento de software. [Familiaridade]

5. Descreva os problemas importantes na seleção de um conjunto de ferramentas para o desenvolvimento de um determinado sistema de software, incluindo ferramentas para rastreamento de requisitos, modelagem de projeto, implementação, automação de compilação e testes. [Familiaridade]

6. Demonstre a capacidade de usar ferramentas de software para apoiar o desenvolvimento de um produto de software de tamanho médio. [Uso]

No Brasil, o Ministério da Educação (MEC) e a Sociedade Brasileira de Computação (SBC) propuseram, separadamente, diretrizes para a formulação de currículos na área da Computação. De acordo com a proposta de diretrizes curriculares, disciplinas e cursos de ES devem incluir formas de integração entre teoria e prática (MEC, 2016). A necessidade de contato dos estudantes com a prática profissional em Engenharia de Software também é enfatizada nos Currículo de Referência da SBC e Referenciais Curriculares (SBC, 2005).

2.1.2 Prática Profissional

A necessidade de contato dos estudantes com a prática profissional em engenharia de software é enfatizada no currículo de referência da SBC para cursos de graduação na

[2]Um currículo deve incluir todos os tópicos da camada Core-Tier1, e quase todos os tópicos da camada Core-Tier2.

área de Computação e Informática, em sua versão de 2005. A Engenharia de Software é tratada como uma das 23 matérias associadas ao núcleo de Tecnologia da Computação, núcleo que compreende "matérias que representam um conjunto de conhecimento agregado e consolidado que capacitam o aluno para a elaboração de solução de problemas nos diversos domínios de aplicação" (SBC, 2005). O conteúdo sugerido para o tópico T7 - Engenharia de Software inclui (SBC, 2005): Processo de Desenvolvimento de Software. Ciclo de Vida de Desenvolvimento de Software. Qualidade de Software. Técnicas de Planejamento e Gerenciamento de Software. Gerenciamento de Configuração de Software. Engenharia de Requisitos. Métodos de Análise e de Projeto de Software. Garantia de Qualidade de Software. Verificação, Validação e Teste. Manutenção. Documentação. Padrões de desenvolvimento. Reuso. Engenharia Reversa. Reengenharia. Ambientes de desenvolvimento de software.

Observa-se que os currículos devem apoiar as necessidades dos alunos para adquirir novas habilidades que são necessárias para lidar com a aquisição de conhecimento e capacitá-los para enfrentar os desafios exigidos pela indústria de software. (BENNANI et al., 2012). De acordo com o currículo de referência de Ciência da Computação, os alunos podem aprender melhor conceitos e práticas de engenharia de software ao participar de um projeto (IEEE; ACM, 2013). Sendo assim, além dos métodos tradicionais de ensino-aprendizagem, métodos alternativos foram introduzidos, como a aprendizagem baseada em projetos (BENNANI et al., 2012).

2.1.3 Problemas no Ensino de ES

Razali e Chitsaz (2010) argumentam que a abordagem tradicional de ensino em disciplinas de engenharia de software é principalmente teórica e unidirecional, tornando o processo de aprendizagem menos interativo. Para Nandigam, Gudivada e Hamou-Lhadj (2008), aulas tradicionais e expositivas, que apresentam apenas princípios e conceitos da ES, dificilmente atraem a atenção dos alunos, pois não há uma conexão clara sobre como eles podem ser usados na prática. Sun (2011) relata que, devido à falta de prática em projetos de software, os alunos falham ao aplicar métodos e técnicas de engenharia de software apropriados para resolver problemas. Andrade et al. (2017) argumenta que o ensino tradicional de ES, voltado para métodos de desenvolvimento, deve ser transformado para refletir a demanda por software mais complexo, já que ainda existe uma lacuna considerável entre os tópicos trabalhados nos cursos e os conhecimentos práticos e habilidades exigidos pela indústria (BOLINGER et al., 2010).

Muitos recém-formados em engenharia de software enfrentam dificuldades quando iniciam suas carreiras profissionais, devido ao desalinhamento das habilidades aprendidas em sua educação universitária com o que é necessário na indústria (GAROUSI et al., 2019b). Essas dificuldades podem ser causadas pela falta de experiência em escrever e entender programas grandes, e a falta de oportunidades para inspecionar e manter código escrito por outros (NANDIGAM; GUDIVADA; HAMOU-LHADJ, 2008). Isso se deve aos seguintes fatos: a prática de ensinar engenharia de software a alunos de graduação geralmente envolve alunos que escrevem pequenos programas do zero (POSTEMA; MILLER; DICK, 2001); os alunos geralmente trabalham individualmente em tarefas de programação para

atender a tarefas específicas fornecidas pelo professor e/ou desenvolvem projetos de software que geralmente não serão utilizados, mantidos ou estendidos (BUFFARDI, 2016).

Buffardi (2016) ressalta ainda que métodos tradicionais de ensino não refletem o desenvolvimento de software profissional e suas práticas, limitando oportunidades para que estudantes de ES tenham contato e experiência no mundo real durante seus estudos acadêmicos (ELLIS et al., 2007). Disciplinas de engenharia de software não deveriam ser desenvolvidas exclusivamente na sala de aula (MARQUES; QUISPE; OCHOA, 2014) e o ensino de princípios e conceitos deveria ter o respaldo de participação ativa dos alunos em projetos reais(MOORE; POTTS, 1994).

Rajlich (2010) argumenta que os projetos de software acadêmicos propostos para os alunos não incorporam desafios encontrados por profissionais da indústria, que têm que lidar com sistemas legados, de tamanho e complexidade maiores, com uso de muitos recursos e alta expectativa sobre a qualidade de código. Para Marques, Quispe e Ochoa (2014), faz-se necessário trabalhar com projetos reais na EES, para que os alunos tenham experiência prática com técnicas e boas práticas adotadas na indústria de software.

Enfim, há um certo consenso sobre a necessidade de currículos e cursos de ES refletirem, adequadamente, o desenvolvimento de software do mundo real para que os alunos obtenham uma melhor compreensão e experiência com as práticas e técnicas utilizadas na indústria (HECKMAN; KING; WINTERS, 2015)(RADERMACHER; WALIA, 2013).

2.1.4 Uso de Projetos na EES

Há diversos métodos para o ensino-aprendizagem de engenharia de software, a saber, cooperação com a indústria (KNUDSON; RADERMACHER, 2009; REICHLMAY, 2006), uso de cenários e projetos realistas (NAUMAN; UZAIR, 2007; TVEDT; TESORIERO; GARY, 2002), *Hackathons* (STEGLICH et al., 2020), *gamificação* (LELLI et al., 2020), e aprendizagem baseada em projetos (KOLMOS, 1996; JEREMIC; JOVANOVIC; GASEVIC, 2009). Este trabalho se concentra no método de aprendizagem baseada em projetos.

A Aprendizagem Baseada em Projetos (*Project-based Learning*) organiza a aprendizagem em torno de projetos (KOLMOS, 1996). Segundo Prince e Felder (2006), é um dos métodos indutivos mais conhecidos na educação em engenharia e tem sido usado universalmente. Abdallah, Toffolon e Warin (2008) afirmam que a aprendizagem baseada em projetos faz parte da cultura instrucional desde Dewey, Decroly e Freinet, e se opõe à pedagogia behaviorista que se baseia em uma transmissão unilateral e passiva do conhecimento do professor para o aluno (ABDALLAH; TOFFOLON; WARIN, 2008). É um método concreto do modelo construtivista baseado na abordagem pedagógica sócio-construtivista e permite a *aprendizagem ativa* (ROGOZAN; HOTTE; ABDULRAB, 2002). Em outras palavras, podemos dizer que contém os componentes pedagógicos do método construtivista, onde os alunos desempenham um papel ativo no processo de aprendizagem (NATTASSHA; AZIZAH, 2015).

Warren (2002) defende a necessidade de um paradigma de aprendizagem ativa, que reconheça que a atividade do aluno é fundamental para o processo de aprendizagem. A aprendizagem baseada em projetos permite aos alunos interagir, colaborar e construir conhecimento, em equipe, sobre projetos desenvolvidos em sala de aula (QUEK, 2010).

Cada equipe tem como objetivo gerar um produto final, de acordo com os objetivos do projeto delineados antecipadamente e endossados pelo tutor (BENNANI et al., 2012).

A aprendizagem baseada em projetos permite que os alunos vivenciem e tenham autoconsciência de aprender e criar seu próprio conhecimento (ANUYAHONG, 2018), aumentando sua motivação e engajamento. Para Srinivasa e Sowmya (2016), a aprendizagem baseada em projetos traz diversos benefícios para os alunos, dentre eles, uma melhor compreensão e retenção mais duradoura do conhecimento teórico quando comparada ao ensino tradicional, e capacidade de aplicação do conhecimento, podendo reproduzir o que aprenderam em novas situações.

Em disciplinas de cursos de Computação, incluindo a engenharia de software, há utilização de projetos pequenos e, em geral, acadêmicos por natureza. Porém, tais projetos nem sempre representam adequadamente cenários realistas e desafios da indústria de software (DAUN et al., 2016), importantes para o ensino-aprendizagem da ES. Considerando-se que, na aprendizagem baseada em projetos, deve-se utilizar um autêntico problema do mundo real (BANAKHR; IQBAL; SHAUKAT, 2018), projetos de código aberto podem ser utilizados em métodos baseados em projetos.

2.2 PROJETOS DE SOFTWARE DE CÓDIGO ABERTO (FLOSS)

Projetos de código aberto têm sido amplamente usados em várias áreas da tecnologia de informação (AMAN et al., 2017). O modelo de software de código aberto tornou-se uma força motriz importante no desenvolvimento de software, atraindo voluntários distribuídos globalmente. Além disso, o número de vagas de emprego que valorizam a experiência com código aberto tem aumentado regularmente (DINIZ et al., 2017).

As licenças são elemento fundamental de projetos de código aberto e dão liberdades essenciais aos usuários para executar um software para qualquer propósito, estudar e modificar o software, e redistribuir cópias do software original ou modificado (TERCEIRO, 2012). O código-fonte dos projetos FLOSS está sempre disponível publicamente e os projetos têm um repositório sob controle de versão acessível (CHAVEZ et al., 2011b). Além do código-fonte, outros artefatos (modelos, conjuntos de teste, entre outros) e informações relacionados ao desenvolvimento de software também estão disponíveis para o público e podem ser revisados e ampliados por qualquer um com interesse e permissão (CHAVEZ et al., 2011b).

Na maioria dos projetos FLOSS, os desenvolvedores estão espalhados por vários locais diferentes do mundo. Em projetos com alta dispersão geográfica, a comunicação é realizada principalmente por meio eletrônico (CHAVEZ et al., 2011b).

Do ponto de vista da ES, há diversos aspectos importantes de projetos de código aberto, incluindo o seu modelo de processo de desenvolvimento, práticas e ferramentas adotadas. Um projeto FLOSS começa quando um desenvolvedor individual ou uma organização decide tornar um produto de software disponível ao público, de forma que ele possa ser livremente utilizado, modificado e redistribuído (TERCEIRO, 2012). Uma versão inicial é lançada e divulgada nos canais apropriados para utilização do software. Os primeiros usuários do software podem ser os próprios desenvolvedores, que poderão revisar o código-fonte e propor mudanças, corrigir defeitos ou adicionar recursos. Essas alterações

podem ser enviadas de volta aos desenvolvedores originais sob a forma de *patches*. As mudanças propostas serão analisadas pelos líderes do desenvolvimento, verificando se as alterações propostas deverão ser aprovadas ou não para serem incluídas na versão oficial (KON et al., 2011).

O processo de desenvolvimento de projetos de software livre normalmente faz uso de um sistema de controle de versão, conforme ilustrado na Figura 2.2. O repositório de controle de versão geralmente está disponível publicamente para leitura. Porém, o acesso de escrita é restrito a um grupo limitado de desenvolvedores. Outros desenvolvedores precisam ter os seus *patches* revisados por um desenvolvedor com as permissões necessárias para que as suas contribuições sejam adicionadas ao repositório oficial do projeto (CHAVEZ et al., 2011b).

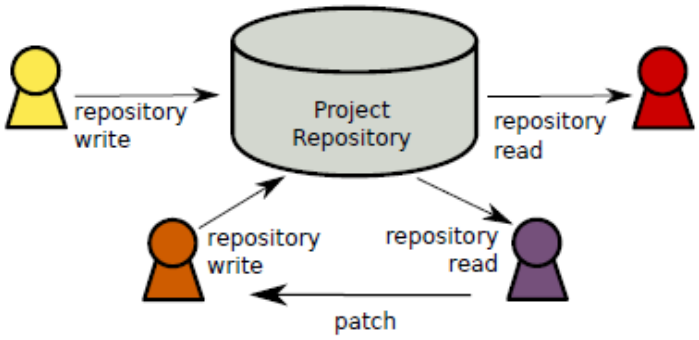

Figura 2.2 Processo de desenvolvimento de software livre (TERCEIRO, 2012).

Além do sistema de controle de versão, a maioria dos projetos de software livre usa um ambiente bastante padronizado para colaboração entre seus desenvolvedores, incluindo um *issue tracker* (ou *bug tracker*) para organizar atividades pendentes, em andamento e futuras. Em geral, também incluem uma ou mais listas de correspondência para coordenação e comunicação entre a equipe e um sistema de gerenciamento de conteúdo baseado na web para escrita de documentação (KON et al., 2011).

Amorim et al. (2017) identificaram mais de 40 práticas em uso na dimensão técnica do ecossistema do KDE, dentre elas: "cada pessoa escolhe o trabalho que deseja realizar entre as tarefas disponíveis no ecossistema (arquivo para traduzir, código para desenvolver, aplicativos para testar, etc)", e "a revisão de código deve ser realizada antes do lançamento da release".

Considerando a relevância de projetos FLOSS e de seu processo de desenvolvimento na indústria de software, este trabalho assume que tais projetos são bons representantes de software do mundo real e de boas práticas usadas no desenvolvimento e evolução de projetos de software para serem utilizados na Educação em Engenharia de Software.

2.3 PROJETOS FLOSS NA EDUCAÇÃO EM ENGENHARIA DE SOFTWARE

Nesta dissertação, um projeto FLOSS é tratado como um objeto pedagógico que pode ser observado, criticado e modificado ao longo do ensino-aprendizagem de conceitos e práticas da engenharia de software. Cabe ao professor planejar como aproveitá-lo em uma ou mais atividades do curso[3].

Projetos FLOSS têm sido utilizados para diferentes fins educacionais (DORODCHI; DEHBOZORGI, 2016) e para trabalhar diferentes tópicos e aspectos da engenharia de software (NASCIMENTO; BITTENCOURT; CHAVEZ, 2015). São *projetos reais* em diferentes domínios, complexidades e comunidades – que possibilitam ao aluno o contato com código-fonte, práticas, pessoas, discussões e recursos reais. Os projetos possuem *licenças de software* e ficam hospedados em repositórios públicos sob *controle de versão*, possibilitando que o aluno aprenda, estude e entenda produto de software e seus processos de desenvolvimento e evolução, com acesso a versões anteriores do projeto para investigar e acompanhar as mudanças realizadas. Além disso, os alunos trabalham com software desenvolvido e mantido por outras pessoas, permitindo que sejam trabalhadas habilidades em tópicos relacionados a *manutenção de software*, por exemplo, *compreensão, correção de erros, e adição de novos recursos*. A disponibilidade do código-fonte permite que técnicas de *engenharia reversa* estudadas sejam colocadas em prática e que modelos, de diferentes propósitos e níveis de abstração, possam ser construídos e analisados. Aspectos teóricos e práticos da área de *teste de software* podem ser trabalhados em projetos FLOSS, do uso de testes existentes e ambientes de teste à criação de novos testes.

Projetos de código aberto promovem a *aprendizagem colaborativa* e em equipe, envolvendo a participação e colaboração entre estudantes, que podem desempenhar diferentes papéis relacionados aos processos de software. Em geral, projetos de código aberto possuem documentação, discussões, e uma comunidade de usuários e desenvolvedores. Assim, alunos de ES podem estudar e modificar software real, entender melhor os princípios fundamentais, e entender a importância e variedade de comunicação que ocorre em projetos reais (KUSSMAUL, 2016).

Os projetos FLOSS também disponibilizam o histórico das modificações, possibilitando aos alunos a verificação e compreensão de quem contribuiu no desenvolvimento do projeto, oferecendo a oportunidade para que suas próprias contribuições sejam visíveis. O uso de licenças de software que garantem que um projeto de software é e permanece livre, de código aberto, evita problemas de propriedade intelectual e desenvolvimento proprietário que podem ocorrer com o envolvimento dos alunos em projetos fechados, com empresas conveniadas (HISLOP et al., 2015).

2.3.1 Motivação para Uso Pedagógico de Projetos FLOSS

O professor precisa estar motivado para utilizar uma abordagem de ensino-aprendizagem baseada em projetos de código aberto na Educação em Engenharia de Software. Nesse contexto, a definição de estratégias e disponibilização de recursos para motivar professores

[3]Ferramentas de apoio aos alunos em suas atividades de desenvolvimento de software (editores, compiladores, etc.), ainda que necessárias, não foram estudadas.

a utilizar projetos FLOSS é importante, pois o ônus do planejamento para uso pode ser maior do que seus benefícios.

A Figura 2.3 apresenta algumas características de projetos FLOSS que os aproximam de projetos de código fechado desenvolvido em empresas (NASCIMENTO, 2017). que Tais resultados podem motivar professores e reforçam a motivação deste trabalho em apoiar o professor na seleção de projetos FLOSS para uso na EES.

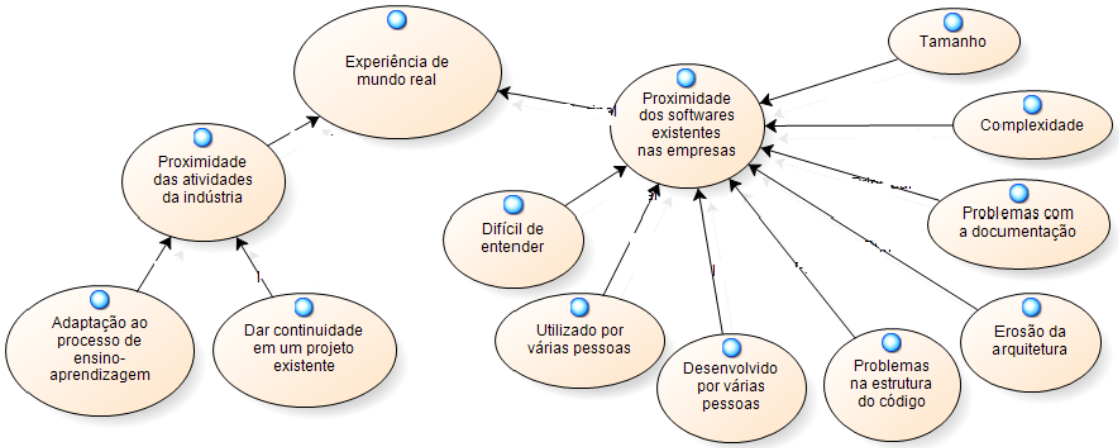

Figura 2.3 Características de projetos FLOSS que se aproximam dos projetos existentes em empresas. (NASCIMENTO, 2017)

2.3.2 Estímulo ao Uso Pedagógico de Projetos FLOSS

Há diversas formas para estimular ou incentivar o uso de projetos FLOSS na EES. Ellis, Purcell e Hislop (2012) apresentam um *framework* de critérios de avaliação que pode ser usado para determinar se projetos FLOSS podem ser utilizados na EES (viabilidade, adequação, maturidade, estabilidade, perspectivas futuras). Os autores ainda ressaltam que a adequação do projeto depende do curso e do conteúdo. O professor deve ter em mente o conteúdo que ele quer abordar com aquele projeto.

O trabalho de Meneely, Williams e Gehringer (2008) apresenta um repositório de código aberto, denominado *Repository for Open Software Education* (ROSE)[4], que armazena projetos e opiniões dos professores sobre como estes projetos podem ser utilizados no ambiente acadêmico. Embora a ideia de um repositório de projetos considerados utéis para o ensino seja bastante interessante, atualmente esse repositório possui apenas três projetos, disponibilizados pelos próprios desenvolvedores do ROSE. Infelizmente, o repositório não está ativo e projetos com potencial para uso no ensino não estão armazenados no repositório.

Ellis et al. (2013) descrevem um modelo para o desenvolvimento do corpo docente baseado em experiências com abordagens acadêmicas e FLOSS prévias ao desenvolvimento do corpo docente. Um workshop de desenvolvimento de professores com duração

[4]Disponível em: http://realsearchgroup.com/rose

de 1,5 dias foi realizado com o objetivo de preparar membros do corpo docente na área de Computação para ministrar cursos nos quais os alunos participariam de projetos de desenvolvimento de software HFOSS (Software Humanitário Livre e de Código Aberto), oferecendo uma visão mais concreta das atividades relacionadas à contribuição em projetos FLOSS que poderiam ser incorporadas ao currículo de um curso específico. O workshop contou com 24 instrutores e três representantes dos projetos HFOSS e FLOSS.

Gokhale, Smith e McCartney (2012) propuseram seleção manual e uma abordagem de avaliação para uma coleção FLOSS baseada em quatro etapas: (1) seleção inicial, (2) avaliação da complexidade do código (métricas), (3) compilação do projeto e análise de design e (4) revisão da documentação. Os autores concluíram que a metodologia de avaliação manual proposta é demorada e cara, e pode desestimular quem deseja adotá-la. Nesse contexto, generalizaram a abordagem para avaliar a adequação do FLOSS, para considerar as opiniões de professores e alunos após a sua utilização. Depois de enviar um curso de engenharia de software duas vezes usando FLOSS selecionado manualmente, esses projetos seriam classificados com base nas visões combinadas do professor, do assistente de ensino e da equipe de alunos em três grupos: "completamente bem-sucedido", "moderadamente bem-sucedido" 'e' "sem sucesso". Com essa abordagem, seria possível fornecer informações aos professores sobre a aplicabilidade do FLOSS utilizado em outros períodos letivos.

Ellis, Hislop e Burdge (2017) propuseram uma abordagem para selecionar projetos HFOSS para a contribuição de alunos. Os critérios usados na seleção dos projetos foram: licença do software, linguagem de programação, design ativo, número de colaboradores, tamanho, rastreamento de problemas (*issue tracking)*, aceitação de novos colaboradores, normas da comunidade e base de usuários.

2.4 SELEÇÃO DE PROJETOS PARA EES

A seleção de projetos de código aberto para contribuição, de modo geral, é um problema relevante e reconhecido. Por exemplo, Terceiro, Souza e Chavez (2012) documentaram padrões para fomentar e ajudar contribuidores em potencial no engajamento em projetos FLOSS. Dentre eles, Padrões de Seleção (*Selection Patterns*) fornecem orientações para a busca e seleção de projetos para contribuir, com base em alguns tipos de critérios baseados nos objetivos do potencial contribuidor (TERCEIRO; SOUZA; CHAVEZ, 2012).

A seleção de projetos de código aberto para uso em disciplinas de engenharia de software não é uma tarefa simples (BISHOP et al., 2016) e diversos trabalhos reportam que a seleção de projetos de código aberto para uso em disciplinas pode ser um problema (JACCHERI; OSTERLIE, 2007; MENEELY; WILLIAMS; GEHRINGER, 2008; GOKHALE; SMITH; MCCARTNEY, 2012; ELLIS; PURCELL; HISLOP, 2012; ELLIS; HISLOP; PURCELL, 2013; SMITH et al., 2014; ELLIS; HISLOP; BURDGE, 2017).

Jaccheri e Osterlie (2007) relatam o processo de seleção de projetos FLOSS para que os alunos atuem como desenvolvedores e pesquisadores. Após a compilação de uma lista de projetos conhecidos, cada um dos projetos selecionados foi avaliado de acordo com 6 critérios: (1) comunidade com 10-50 desenvolvedores ativos; (2) comunidade deve permitir a entrada de novos contribuidores; (3) técnicas (planejamento de projetos, etc.)

devem ter sido utilizadas no desenvolvimento do projeto; (4) implementação em Java ou C++; (5) listas de e-mail, bate-papo e rastreamento de bugs disponíveis; (6) deve ser útil para o pesquisador. Critérios importantes foram considerados na avaliação de um subconjunto de projetos conhecidos para apoiar a seleção pelo professor. Porém, os processos de busca e avaliação de projetos foram realizados manualmente, ao contrário da abordagem proposta nessa dissertação.

Ellis, Purcell e Hislop (2012) apresentam uma abordagem para a seleção de projetos FLOSS com base em uma estrutura de critérios de avaliação. No entanto, o processo de seleção e avaliação apresentado é manual, demandando muito tempo dos professores. Esta pesquisa de mestrado utilizou alguns dos critérios sugeridos por Ellis, Purcell e Hislop (2012), porém de forma mais organizada e em uma abordagem automatizada.

Para Ellis, Hislop e Purcell (2013), a seleção de um projeto adequado pode ser difícil devido à grande quantidade de projetos disponíveis e diversidade de características a serem consideradas (tamanho, complexidade, domínio e comunidade). Em seu estudo, Smith et al. (2014) corroboram essa afirmação, destacando que, como os projetos de código aberto apresentaram diversidade em termos de tamanho, complexidade e qualidade, a seleção de projetos adequados foi considerada um obstáculo.

Smith et al. (2014) discutem sobre a busca por projetos FLOSS com foco no ensino de manutenção e evolução do software. Foi feito um trabalho manual para preparar e avaliar os projetos para integração, observando se esses projetos atendiam a determinadas condições/restrições definidas, tais como: tamanho de código pré-estabelecido, linguagem de programação, domínio da aplicação, projeto modular, atividade recente, e boa documentação. Os autores relatam que 12 horas foram utilizadas para examinar os projetos de software candidatos, e três computadores foram mantidos funcionando, realizando downloads de projetos escritos na linguagem Java que pareciam interessantes conforme uma inspeção manual prévia. Eles concluíram que a pesquisa revelou-se bastante trabalhosa, contrariando as expectativas de que seria bem rápida e simples. A abordagem proposta neste trabalho buscou atenuar o trabalho de selecionar projetos FLOSS por meio do suporte automatizado para seleção e busca de projetos.

Em relação a abordagem proposta por Gokhale, Smith e McCartney (2012) (detalhada na seção 2.3.2), embora possa haver eficiência ao trabalhar com os mesmos projetos em atividades de ensino, pode ser necessário selecionar novos projetos FLOSS cada vez que o curso é oferecido (ELLIS et al., 2013). Neste trabalho, a seleção de projetos FLOSS deve atender a um conjunto de critérios definidos pelo professor, possibilitando, se necessário, que um novo conjunto de projetos seja considerado em cada semestre da disciplina de engenharia de software.

A Figura 2.4 apresenta fatores que podem influenciar a seleção de um projeto FLOSS para uso na EES (NASCIMENTO, 2017). O fator *domínio do software* está relacionado à principal área do projeto (e.g. jogos), *tecnologia do projeto* refere-se às tecnologias utilizadas (e.g. linguagem de programação), *situação do código* refere-se a organização do código para seu manuseio por vários desenvolvedores, o fator *situação da documentação* está relacionado à existência ou não de documentação e qualidade associada e, por fim, o fator *atendimento da proposta didática* está relacionado à possibilidade de trabalhar um tópico específico de ES com o projeto.

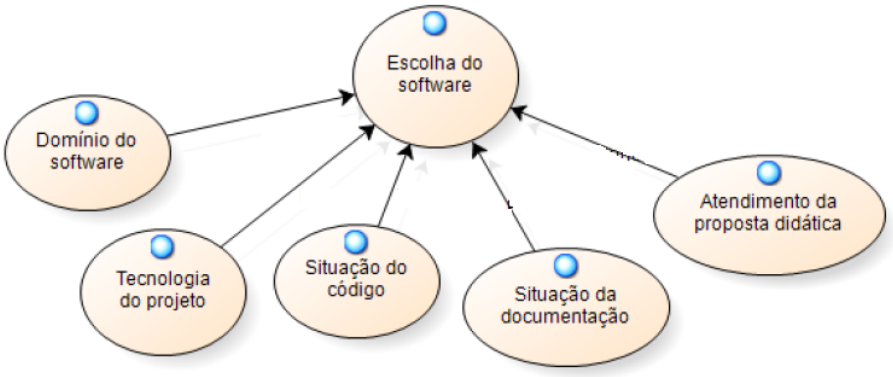

Figura 2.4 Fatores que podem influenciar a seleção de projeto (NASCIMENTO, 2017).

2.5 CONSIDERAÇÕES FINAIS

A integração de experiência prática na Educação em Engenharia de Software é necessária, mas não se pode perder de vista os objetivos de aprendizagem e tópicos definidos nos currículos de referência da área. O uso de projetos de código aberto mostra-se como uma alternativa viável para tal integração, pois oferece a professores e alunos liberdade para explorar código-fonte, artefatos e práticas usadas de projetos reais, podendo desenvolver habilidades técnicas e sociais necessárias para a formação de profissionais para trabalhar na indústria de software. Nesse contexto, esta dissertação de mestrado propõe uma abordagem sistemática, com suporte ferramental, para apoiar o professor no desafio de selecionar projetos de código aberto para utilização no ensino-aprendizagem de ES.

Capítulo

3

MÉTODOS DE PESQUISA

Este capítulo apresenta o processo e os métodos usados nesta pesquisa (Seção 3.1), descrevendo os métodos de pesquisa usados para identificação de critérios de seleção de projetos FLOSS (Seção 3.2) e para avaliação da abordagem proposta e representada pela ferramenta FlossSearch.Edu (Seção 3.3). Dois estudos foram realizados para coletar dados sobre comportamentos e opiniões dos professores e estudantes sobre a utilidade percebida, facilidade de uso percebida e atitude em relação à seleção de projetos FLOSS para EES com a ferramenta proposta.

3.1 PROCESSO DE PESQUISA

***Warm-up* - Definição do Escopo e Motivação**

A motivação para trabalhar com o problema da seleção de projetos de código aberto para uso pedagógico na EES teve como ponto de partida os resultados e desafios reportados em um mapeamento sistemático sobre uso de projetos de código aberto no ensino-aprendizagem de engenharia de software (NASCIMENTO; BITTENCOURT; CHAVEZ, 2015). Para confirmar a relevância do problema nesse contexto, analisamos outros trabalhos, identificados manualmente por meio da técnica de *snowballing* (WEBSTER; WATSON, 2002) aplicadas aos estudos que trataram explicitamente do problema de seleção de projetos FLOSS.

Identificação e Classificação de Critérios

A seleção de projetos de código aberto para uso pedagógico na EES pode considerar diversas características dos projetos e suas comunidades. O mapeamento sistemático sobre uso de projetos de código aberto no ensino-aprendizagem de engenharia de software realizado por Nascimento, Bittencourt e Chavez (2015) trouxe evidências, com base em trabalhos publicados até 2013, sobre a existência e uso explícito de critérios definidos pelo professor para a seleção de projetos. Nesta pesquisa, uma atualização desse mapeamento sistemático foi realizada, resultando na

inclusão e discussão de 33 estudos primários publicados entre 2014 e 2017 (BRITO et al., 2018). O mapeamento atualizado representa um ativo importante para este novo campo de pesquisa que trata de questões de ensino-aprendizagem de engenharia de software com projetos FLOSS.

Para responder à questão de pesquisa **RQ1**, relacionada à identificação de critérios usados na seleção de projetos de código aberto, um subconjunto dos estudos primários trazidos pelos dois mapeamentos sistemáticos (NASCIMENTO; BITTENCOURT; CHAVEZ, 2015; BRITO et al., 2018), foi usado para para identificar, classificar, descrever e organizar diversos critérios de seleção de projetos, resultando em um *catálogo de critérios de seleção*.

A classificação dos critérios de seleção (conforme o objetivo **O1**) considerou o *nível de controle* desejado pelo professor sobre o projeto a ser selecionado. Neste trabalho, foram considerados dois *níveis de controle* do professor, introduzidos por Nascimento, Bittencourt e Chavez (2015): *Controle Interno* e *Nenhum Controle*. Detalhes sobre os métodos utilizados e resultados são apresentados no Capítulo 4.

Organização e Operacionalização dos Critérios.

Os estudos primários identificados pelos mapeamentos sistemáticos (NASCIMENTO; BITTENCOURT; CHAVEZ, 2015; BRITO et al., 2018) apresentam uma descrição simples ou apenas mencionam os critérios de seleção de projetos usados. Em geral, não há orientação sobre como devem ser usados para a seleção de projetos de código aberto adequados, na perspectiva do professor.

Neste trabalho, para alcançar o objetivo **O2** e responder à questão de pesquisa **RQ2**, a organização dos critérios de seleção seguiu o modelo de catálogo, com uma descrição individual de cada critério de seleção, inspirada no formato de padrões de projeto (GAMMA, 1995), conhecido por professores e estudantes. O catálogo de critérios de seleção é apresentado no Capítulo 4.

Desenvolvimento e Avaliação de uma Ferramenta.

A ferramenta `FlossSearch.edu` foi desenvolvida como prova de conceito para a seleção automática de projetos de código aberto com base em critérios de seleção pré-definidos. O Capítulo 5 apresenta a ferramenta.

Duas avaliações foram realizadas. A primeira avaliação, um estudo piloto, foi realizada apenas com estudantes. A segunda e principal avaliação foi realizada por meio de um estudo experimental com um grupo de professores de engenharia de software, escolhidos por conveniência. Nos dois estudos, levantamentos com os participantes foram realizados, com instrumento (questionário) baseado no Modelo de Avaliação de Tecnologia TAM (LAITENBERGER; DREYER, 1998). No segundo estudo apenas, sessões individuais com uso do protocolo de *Think Aloud* (ERICSSON; SIMON, 1993) foram realizadas com professores que utilizam projetos FLOSS como objeto de estudo em suas disciplinas.

3.2 IDENTIFICAÇÃO DE CRITÉRIOS DE SELEÇÃO

Para identificar critérios de seleção de projetos, fizemos uma revisão exploratória, por meio da releitura dos artigos mapeados por Nascimento, Bittencourt e Chavez (2015), com atenção voltada para artigos que mencionavam uso ou preocupação com critérios para selecionar projetos, seguida pela aplicação da técnica de *snowballing* para identificar outros artigos relevantes (Seção 3.2.1). Os resultados preliminares da revisão exploratória motivaram a realização de uma atualização do mapeamento sistemático de Nascimento, Bittencourt e Chavez (2015) para incluir trabalhos mais recentes, publicados entre 2013 e 2018, identificar novos critérios de seleção, ou encontrar outras evidências sobre a utilização dos critérios já identificados (Seção 3.2.2).

3.2.1 Revisão Exploratória

O mapeamento sistemático da literatura sobre o uso de projetos FLOSS na EES conduzido por Nascimento, Bittencourt e Chavez (2015) reportou 72 trabalhos publicados até 2013. Dentre eles, 7 trabalhos continham informações explicitas sobre a seleção de projetos, em especial, com a necessidade de seleção feita com base em restrições ou definidas pelo professor (JACCHERI; OSTERLIE, 2007; ELLIS; PURCELL; HISLOP, 2012; PAPADOPOULOS; STAMELOS; MEISZNER, 2013; BUCHTA et al., 2006; GOKHALE; SMITH; MCCARTNEY, 2013b, 2012; HSIEH et al., 2013). Com base em tais resultados, (i) estudos primários que se preocuparam com o *problema da seleção de projetos* foram revisitados, e (ii) a técnica de *snowballing* foi aplicada às suas referências bibliográficas, buscando identificar outros estudos relacionados ao problema da seleção de projetos FLOSS. Ao final, um conjunto inicial de critérios para seleção de projetos foi identificado, dentre os usados de modo recorrente, como potenciais candidatos para a seleção automatizada de projetos.

3.2.2 Atualização de Mapeamento Sistemático

O objetivo de um mapeamento sistemático da literatura é proporcionar uma visão geral e abrangente dos estudos existentes sobre um determinado tema (KITCHENHAM; DYBA; JORGENSEN, 2004).

Neste trabalho, uma atualização do mapeamento sistemático de Nascimento, Bittencourt e Chavez (2015) foi conduzida para complementar as evidências que haviam sido coletadas até 2012[1] sobre o uso pedagógico de projetos de código aberto na EES, com novos estudos realizados entre 2013 e 2018. O interesse mais específico desta pesquisa de mestrado sobre critérios de seleção de projetos motivou a realização de uma análise detalhada dos estudos selecionados com relato sobre o processo de seleção de projetos.

A Figura 3.1 apresenta o processo que orientou a atualização do mapeamento sistemático original conduzido por Nascimento, Bittencourt e Chavez (2015). Etapas do mapeamento anterior, marcadas com (*), foram reutilizadas. A ferramenta StArt[2] (State

[1] Apesar do estudo ter sido publicado em 2015, o processo de busca do mapeamento sistemático de Nascimento, Bittencourt e Chavez (2015) foi realizado em 2012.

[2] http://lapes.dc.ufscar.br/tools/start_tool

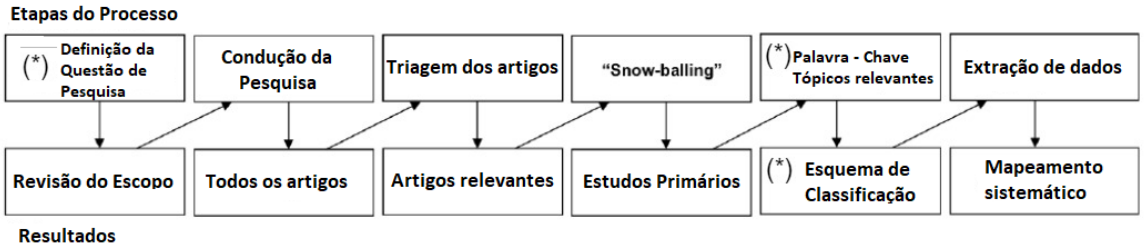

Figura 3.1 Processo de atualização de mapeamento, adaptado de (PETERSEN et al., 2008)

of the Art através da Revisão Sistemática) apoiou o processo de mapeamento e planilhas foram usadas para organizar os dados coletados.

Três questões de pesquisa (a questão principal e duas questões secundárias) do mapeamento original (NASCIMENTO; BITTENCOURT; CHAVEZ, 2015) foram reutilizadas na atualização. A seguinte pergunta orientou nosso mapeamento: *RQ1. Como os projetos de código aberto são usados na educação de engenharia de software?* O escopo da revisão reutilizou as sequências de busca e os critérios de inclusão/exclusão para pesquisa e triagem dos artigos. Quatro critérios de inclusão (IC) e 20 critérios de exclusão (CE) definidos por Nascimento, Bittencourt e Chavez (2015) foram reutilizados. Por exemplo: (IC.1) Estudos que abordam o uso de projetos FLOSS para aprender/ensinar Engenharia de Software devem ser incluídos, independentemente de sua aplicação em cursos de Engenharia de Software ou em outros cursos da área; (EC.1) Documentos redigidos em idiomas diferentes do inglês devem ser excluídos; (EC.2) Documentos cujo texto completo não está disponível devem ser excluídos; e (EC.3) Estudos cujo conteúdo principal não esteja relacionado ao aprendizado ou ensino de Engenharia de Software devem ser excluídos. O protocolo e os dados do mapeamento sistemático original podem ser encontradas em seu website[3] e o protocolo e dados da nossa atualização foram disponibilizados como material suplementar[4].

A busca foi realizada no período de 22 de março a 28 de abril de 2018. A estratégia de busca incluiu as bases: Engineering Village[5], IEEE Xplore[6], ACM[7], Scopus[8], Springer[9] and Science Direct[10]. Foram recuperados 18.056 artigos, dos quais 13.924 foram duplicados (vários estudos foram indexados por mais de uma biblioteca digital), levando a 4.132 trabalhos selecionados para triagem. Foram aplicados dois filtros durante a triagem. Primeiro, em parceria com uma aluna de doutorado, pertencente ao mesmo grupo de pesquisa, foram examinados, individualmente, o título e o resumo de cada estudo primário, marcando cada estudo como incluído ou excluído. Segundo, os resultados fo-

[3]https://sites.google.com/site/dmcnascimento/mapping
[4]https://github.com/Moara/mapping/blob/master/Disponibilizar/Protocolo.pdf
[5]http://www.engineeringvillage.com
[6]http://ieeexplore.ieee.org
[7]http://portal.acm.org
[8]http://www.info.sciverse.com/scopus
[9]http://www.springer.com
[10]http://www.elsevier.com

ram revisados pela orientadora da pesquisa, analisando conflitos e tomando a decisão final. Nesta primeira etapa, 39 estudos foram incluídos. Então, os estudos restantes foram reexaminados, com base na leitura da introdução e da conclusão. Novamente foi feito o processo de comparação dos resultados, restando 33 estudos primários selecionados para classificação.

Informações sobre o título, autores, detalhes de afiliação dos autores, local e ano de publicação de cada artigo selecionado foram extraídas, e algumas informações adicionais necessárias para classificá-lo segundo as facetas definidas no estudo de Nascimento, Bittencourt e Chavez (2015).

Nesta dissertação, apenas os novos resultados para a Faceta 9 relacionada à *escolha de projetos* são detalhados. A Tabela 3.1 apresenta as categorias da Faceta 9 e sua descrição.

Tabela 3.1 Faceta 9 - Escolha de Projetos

Categoria	Descrição
Pré-definido	Professor decide o projeto no qual o aluno irá trabalhar.
Lista de Escolha	Os alunos podem escolher qualquer projeto da lista fornecida pelo corpo docente / equipe.
Livre Escolha	Os estudantes devem procurar e decidir com qual projeto FLOSS (de seu interesse) trabalharão.
Não especificado	O artigo não menciona questões relacionadas à escolha do projeto.
Não se aplica	O trabalho não está relacionado a uma experiência em que é necessário escolher um projeto.

3.3 AVALIAÇÃO DA ABORDAGEM

A avaliação da abordagem proposta com estudantes e professores foi feita em dois estudos, realizados em momentos distintos. No primeiro estudo, a avaliação da abordagem com estudantes de graduação foi realizada com base nas respostas dadas a um questionário fundamentado no Modelo de Aceitação de Tecnologia *Technology Acceptance Model* (TAM), introduzido por Davis (DAVIS, 1989). No segundo estudo, a avaliação da abordagem por professores foi realizada com base no textos transcritos da gravação de dez sessões de *Think Aloud* (ERICSSON; SIMON, 1993), virtuais e individuais, e nas respostas dada ao questionário desenhado com base no modelo TAM.

3.3.1 Modelo de Aceitação de Tecnologia (TAM)

O modelo TAM visa avaliar a utilidade percebida, a facilidade de uso percebida e intenção de uso futuro de, por exemplo, uma técnica ou ferramenta. Considera-se que a percepção do usuário sobre a utilidade e facilidade de uso de uma tecnologia são determinantes para o comportamento de aceitação da tecnologia pelo usuário e seu uso prático (LAITENBERGER; DREYER, 1998).

Uma ferramenta com percepção de utilidade alta indica que o usuário acredita na existência de uma relação positiva entre uso e desempenho (DAVIS, 1989). A facilidade de uso percebida refere-se ao “grau” em que uma pessoa acredita que usar uma determinada

ferramenta seria fácil (DAVIS, 1989). Tanto a utilidade percebida quanto a facilidade de uso estão correlacionadas com a intenção de uso futuro, e podem ser consideradas determinantes dos comportamentos de aceitação da ferramenta (DAVIS, 1989).

O modelo TAM tem sido amplamente aplicado na avaliação de tecnologia. Dessa forma, acredita-se que esse modelo seja adequado para a avaliação da facilidade de uso, da utilidade percebida e intenção de uso futuro da abordagem para seleção de projetos FLOSS implementada na ferramenta FlossSearch.Edu.

A Tabela 3.2 apresenta o modelo de avaliação TAM utilizado com os professores e a Tabela 3.3 apresenta o modelo de avaliação TAM utilizado com estudantes. Ambos tomaram como base modelos TAM adaptados (DAVIS, 1989; LAITENBERGER; DREYER, 1998; STEINMACHER et al., 2016), compostos por conjuntos de questões que medem cada um dos três construtos principais: "utilidade percebida" (Ui), "facilidade de uso" (Ei) e "uso futuro auto-previsto" (Si). Os mesmos itens apresentados por Laitenberger e Dreyer (1998), com base no conjunto originalmente proposto por Davis (1989). foram utilizados. Na adaptação do instrumento de avaliação, utilizou-se o FlossSearch.Edu como objeto do questionário e a seleção de projetos FLOSS para uso na EES como processo investigado.

Para medição subjetiva, usou-se a escala Likert (MCIVER; CARMINES, 1981) nas questões sobre utilidade percebida e facilidade de uso, apresentadas aos participantes na forma de um conjunto de afirmações para quenos participantes respondessem de acordo com seu próprio nível de concordância ou discordância. Quanto às possibilidades de resposta, esta pesquisa adotou a escala de seis pontos sem valor neutro disponível usada por Steinmacher et al. (2016), de modo que os participantes pudessem expressar claramente sua tendência para uma avaliação positiva ou negativa. Os participantes deveriam selecionar uma dentre as seis respostas a seguir: *Discordo Extremamente*, *Discordo*, *Discordo Levemente*, *Concordo Levemente*, *Concordo* e *Concordo Extremamente*.

Por fim, três questões abertas foram incluídas no instrumento aplicado (Tabela 3.4) para obter a opinião dos estudantes sobre os critérios de seleção de projetos FLOSS e uma visão geral da ferramenta FlossSearch.Edu.

3.3.2 Think Aloud (TA)

O método *Think Aloud* (TA) busca dar suporte para o rastreamento de processos cognitivos de participantes durante atividades de tomada de decisão (AMORIM et al., 2017). Protocolos de TA são amplamente usados por profissionais de usabilidade para tentar identificar temas relevantes sobre os quais os usuários possam estar pensando. Parte-se do que foi verbalizado pelos participantes para obter *insights* sobre os comportamentos estudados, por exemplo, busca na web, ou para avaliar o conteúdo e a facilidade de uso de um *website* (OLMSTED-HAWALA et al., 2010; AMORIM et al., 2017).

Os protocolos podem utilizar o *pensamento simultâneo em voz alta*, em que participantes verbalizam seus pensamentos durante a execução de uma tarefa, ou o *pensamento retrospectivo em voz alta*, em que participantes o fazem após a conclusão da tarefa (ERICSSON; SIMON, 1993). Um dos protocolos mais difundidos é o de TA simultâneo (OLMSTED-HAWALA et al., 2010), baseado nas técnicas de análise de proto-

colo de Ericsson e Simon (1993).

Tabela 3.2 Itens usados para medir a utilidade, facilidade de uso e intenção de uso no futuro na perspectiva do professor.

Utilidade Percebida	
U1	Seria difícil selecionar um projeto de código aberto com as características desejadas sem FlossSearch.Edu.
U2	Usar FlossSearch.Edu me dá mais controle sobre meu trabalho de seleção.
U3	Usar FlossSearch.Edu melhora meu desempenho na seleção do projeto apropriado.
U4	FlossSearch.Edu atende às minhas necessidades relacionadas à seleção de projetos.
U5	Usar FlossSearch.Edu economiza tempo na seleção de um projeto adequado.
U6	Usar o FlossSearch.Edu me permite selecionar um projeto mais rápido do que selecioná-lo de outra forma.
U7	Usar o FlossSearch.Edu facilita o uso de projetos de código aberto na aula de Engenharia de Software.
U8	Usar FlossSearch.Edu aumenta minha eficácia na seleção de projetos.
U9	Usar FlossSearch.Edu torna meu trabalho de seleção de projetos mais fácil.
U10	Em geral, acho que FlossSearch.Edu pode ser útil no desenvolvimento de minhas tarefas na disciplina.
Facilidade de Uso	
E1	Aprender como operar FlossSearch.Edu foi fácil para mim.
E2	Acho fácil fazer a FlossSearch.Edu fazer o que eu quero.
E3	Costumo ficar confuso quando uso a FlossSearch.Edu.
E4	Eu cometo erros frequentemente ao usar FlossSearch.Edu.
E5	Interagir com FlossSearch.Edu costuma ser frustrante.
E6	Preciso consultar a guia "Sobre" com frequência ao usar o Floss Search.Edu.
E7	Minha interação com a FlossSearch.Edu foi clara e compreensível.
E8	É fácil lembrar como realizar tarefas usando Floss Search.Edu.
E9	FlossSearch.Edu é rígido e inflexível para interagir.
E10	No geral, acho FlossSearch.Edu fácil de usar.
Intenção de Uso Futuro	
S1	Supondo que FlossSearch.Edu esteja disponível para selecionar qualquer projeto FLOSS, prevejo que o usarei no futuro.
S2	Eu preferiria usar FlossSearch.Edu para me apoiar na seleção de projetos FLOSS com base em características especificadas.

Tabela 3.3 Itens usados para medir a utilidade, facilidade de uso e intenção de uso no futuro na perspectiva do aluno.

Utilidade Percebida	
U1	Seria difícil selecionar um projeto de código aberto com as características desejadas sem FlossSearch.Edu.
U2	Usar FlossSearch.Edu me dá mais controle sobre meu trabalho de seleção.
U3	Usar o FlossSearch.Edu melhora meu desempenho na seleção do projeto adequado.
U4	O FlossSearch.Edu atende às minhas necessidades relacionadas a seleção de projetos.
U5	Usar o FlossSearch.Edu me poupa tempo na seleção de um projeto adequado.
U6	O uso do FlossSearch.Edu me permite selecionar um projeto mais rapidamente do que selecionar utilizando outra forma.
U7	Usar o FlossSearch.Edu facilita para que eu utilize projetos de código aberto, na aula de Engenharia de Software.
U8	Usar o FlossSearch.Edu aumenta minha eficácia na seleção de projetos.
U9	O uso do FlossSearch.Edu facilita o meu trabalho de seleção de projetos.
U10	No geral, acho que o FlossSearch.Edu pode ser útil no desenvolvimento de minhas tarefas na disciplina.
Facilidade de Uso	
E1	Aprender como operar FlossSearch.Edu foi fácil para mim.
E2	Acho fácil fazer a FlossSearch.Edu fazer o que eu quero.
E3	Costumo ficar confuso quando uso a FlossSearch.Edu.
E4	Eu cometo erros frequentemente ao usar FlossSearch.Edu.
E5	Interagir com o FlossSearch.Edu é muitas vezes frustrante.
E6	Preciso consultar a aba “About” frequentemente ao usar o FlossSearch.Edu.
E7	Minha interação com a FlossSearch.Edu foi clara e compreensível.
E8	É fácil lembrar como realizar tarefas usando Floss Search.Edu.
E9	O FlossSearch.Edu é rígido e inflexível para interagir.
E10	No geral, acho FlossSearch.Edu fácil de usar.
Intenção de Uso Futuro	
S1	Supondo que o FlossSearch.Edu estaria disponível para selecionar qualquer projeto FLOSS, eu prevejo que vou utilizá-lo no futuro.
S2	Eu preferiria usar o FlossSearch.Edu na seleção de projetos, me apoiando na busca por projetos FLOSS com características descritas no cenário da disciplina de Tópicos de Engenharia de Software.

Tabela 3.4 Questões Abertas

Nº	**Questão**
1	Em relação às características do projeto que foram apresentadas no cenário, você conseguiu identificar os projetos que atendiam a essas restrições? Como você realizou o processo de busca? Utilizou algum filtro disponível no FlossSearch.Edu ou outra ferramenta externa?
2	Sobre os critérios de seleção apresentados na ferramenta FlossSearch.Edu, você sugere a inclusão de algum outro? Qual?
3	De maneira geral, dê a sua opinião sobre a ferramenta FlossSearch.Edu.

Capítulo 4

IDENTIFICAÇÃO, CLASSIFICAÇÃO E CATALOGAÇÃO DE CRITÉRIOS PARA SELEÇÃO DE PROJETOS DE CÓDIGO ABERTO

O mapeamento sistemático original (NASCIMENTO; BITTENCOURT; CHAVEZ, 2015) e a atualização realizada no escopo desta pesquisa (FERREIRA et al., 2018b), organizaram os estudos primários em nove facetas ou dimensões, cada uma com 2 ou mais categorias. Este capítulo apresenta os novos estudos primários trazidos pela atualização mapeamento sistemático e dez critérios de seleção de projetos FLOSS. Os estudos primários foram categorizados na *Faceta 9 - Seleção de Projetos*, segundo o *grau de liberdade na escolha de projetos pelos estudantes* (Seção 4.1) e, os estudos primários e dez critérios de seleção escolhidos no contexto desta pesquisa foram categorizados segundo o *nível de controle do professor sobre o projeto* (Seção 4.2). Por fim, um catálogo de critérios de seleção para projetos FLOSS, proposto para documentar os dez critérios de seleção escolhidos nesta pesquisa e promover seu reuso é apresentado (Seção 4.3).

4.1 CLASSIFICAÇÃO SEGUNDO A LIBERDADE NA ESCOLHA DO PROJETO

Nesta parte do trabalho, apresentamos apenas a classificação dos estudos primários segundo a *Faceta 9 - Seleção de Projetos* e suas cinco categorias, dada a sua relevância para o escopo da pesquisa. As categorias desta faceta representam o *grau de liberdade* dos estudantes na escolha de projetos FLOSS para uso nas atividades da disciplina.

A Tabela 4.1 apresenta os 33 estudos primários selecionados e classificados segundo as cinco categorias da Faceta 9 (ver Tabela 3.1 no capítulo anterior). Uma análise dos artigos que tratam de problemas relacionados à seleção de projetos é apresentada a seguir, organizada por categoria.

- **Projeto Pré-definido** Em sete estudos, os alunos trabalharam em projetos pré-definidos. Por exemplo, Morgan e Jensen (2014) escolheram o projeto Ubuntu para todos os seus alunos trabalharem, uma vez que tal projeto tem várias opções para

Tabela 4.1 Estudos Classificados na *Faceta 9 - Seleção de Projetos*

Categoria	**Estudos**	#
Pré-definido	Chen, Memon e Luo (2014), MacKellar, Sabin e Tucker (2015), Kussmaul et al. (2017), Holmes et al. (2014), Ellis et al. (2014), Diniz et al. (2017), Hislop et al. (2015)	7
Lista para escolha	Gokhale, Smith e McCartney (2013b), Buffardi (2015), Fernandes e Barbosa (2016), Papadopoulos, Stamelos e Meiszner (2013), Hislop et al. (2015)	5
Livre escolha	Deursen et al. (2017), Krutz, Malachowsky e Reichlmayr (2014), Smith et al. (2014), Smith, Gokhale e McCartney (2014), Rekha e Adinarayanan (2014), Cicirello (2017), Fernandes et al. (2013), Pinto et al. (2017)	8
Não especificado	Buffardi (2016), Villarrubia e Kim (2015), Krishnamoorthy, Appasamy e Scaffidi (2013), Gokhale, Smith e McCartney (2013a), Ding et al. (2014), Dorodchi e Dehbozorgi (2016), Ellis et al. (2013, 2015), Kussmaul (2016), Mishra, Hacaloglu e Mishra (2014)	10
Não se aplica	Bowring e Burke (2016), Castelluccia e Visaggio (2013), Hislop e Ellis (2017)	3

contribuidores, incluindo documentação, design, desenvolvimento, bugs e testes. Diniz et al. (2017) escolheram o JabRef, um projeto FLOSS consolidado, onde um dos autores faz parte da equipe de manutenção do projeto, o que facilitou sua instrumentação.

- **Lista para Escolha** Cinco estudos reportam a seleção de projetos com base em uma *lista de projetos pré-escolhidos* pelo professor. Gokhale, Smith e McCartney (2013b) analisaram e avaliaram manualmente cerca de 1.000 projetos hospedados em vários repositórios FLOSS e disponibilizaram para escolha uma lista com dezessete projetos compatíveis com a experiência dos alunos. Os projetos incluídos variaram em tamanho de 5.500 a 10.500 linhas de código. Smith, Gokhale e McCartney (2014) listaram dez projetos de código aberto para explorar, experimentar e oferecer para escolha dos alunos. Por fim, no estudo reportado por Papadopoulos, Stamelos e Meiszner (2013), os alunos escolheram o projeto a partir de uma lista apresentada em sala de aula.

- **Livre Escolha** Na maioria dos estudos primários trazidos pela atualização do mapeamento, em que os autores mencionaram explicitamente a forma como os projetos foram selecionados, os alunos trabalharam em projetos de "livre escolha", divergindo do mapeamento sistemático original que reportou que, na maioria dos estudos, os alunos trabalhavam em projetos *pré-definidos*.

  Porém, observou-se que a livre escolha não era tão livre assim. Oito artigos cuja seleção de projetos foi de livre escolha, dentre eles, os estudos de Deursen et al.

(2017) e de Krutz, Malachowsky e Reichlmayr (2014), a escolha do projeto de código aberto do mundo real a ser usado no curso SE pelos alunos foi supervisionada pelo professor, de modo que os projetos escolhidos atendessem a certas condições e restrições, por exemplo, tamanho do projeto, atividade e número de desenvolvedores. Abordagens desse tipo fortalecem a nossa premissa de que, para selecionar projetos FLOSS adequados para uso pedagógico, é necessário definir critérios de seleção para guiar a filtragem uniforme dos projetos. O estudo apresentado por Smith, Gokhale e McCartney (2014) analisou aos motivos que alunos de um curso de engenharia de software levaram em consideração na definição de elementos para guiar sua escolha de projetos de código aberto.

Como no mapeamento anterior, também identificamos estudos que forneceram uma lista de opções de projetos FLOSS, mas os alunos tinham a liberdade de escolher um projeto que não estivesse na lista (FERNANDES et al., 2013).

4.2 CLASSIFICAÇÃO SEGUNDO O NÍVEL DE CONTROLE

Em seu mapeamento sistemático, Nascimento, Bittencourt e Chavez (2015) apresentaram três níveis de controle do professor sobre as atividades dos estudantes no projeto: *Controle Total*, *Controle Interno* e *Nenhum controle*. Neste trabalho foram considerados apenas dois níveis de controle: *Controle Interno* e *Nenhum Controle*.

Professores que optam pelo *Controle Interno*[1] de projetos em sua disciplina, fazem uma cópia independente (*fork*) de um projeto FLOSS, preparam atividades práticas e avaliam a contribuição dos alunos. Após essas etapas, contribuições podem ser opcionalmente submetidas à aprovação da comunidade que mantém o software. Em disciplinas em que o nível *Nenhum Controle*[2] é adotado, o professor atua como um mentor externo que acompanha as atividades realizadas pelos alunos diretamente no projeto FLOSS, com base em tarefas e solicitações da própria comunidade. A comunidade de um projeto pode incorporar ou não a contribuição dos alunos no repositório de origem do projeto.

Dez critérios de seleção foram escolhidos para compor esta pesquisa. A Tabela 4.2 apresenta os critérios de seleção e suas respectivas categorias. O uso de alguns critérios de seleção, por exemplo, *comunidade ativa*, é recomendado quando os alunos pretendem interagir e atender às solicitações da comunidade do projeto (*Nenhum Controle*), mas nem tanto quando alunos realizam atividades propostas pelo professor em uma ramificação do projeto (*Controle Interno*).

A Tabela 4.2 também mostra as referências dos estudos primários que evidenciaram cada um dos dez critérios de seleção usados nesta pesquisa. A descrição detalhada dos critérios de seleção e seus usos conhecidos é apresentada na Seção 4.3.

[1] *Inside Control*
[2] *No Control*

Tabela 4.2 Estudos que evidenciam cada Critério. (*Todos os critérios pertencentes a "Controle Interno", também pertencem a "Nenhum Controle")

Categoria	**Critério**	**Estudos**
Controle Interno	Linguagem de Programação	(Ellis, Hislop e Burdge (2017), Ellis, Purcell e Hislop (2012), Gokhale, Smith e McCartney (2013b), Rekha e Adinarayanan (2014), Gokhale, Smith e McCartney (2012), Papadopoulos, Stamelos e Meiszner (2013))
	Tamanho do Projeto	(Ellis, Hislop e Burdge (2017), Ellis, Purcell e Hislop (2012), Jaccheri e Osterlie (2007), Rekha e Adinarayanan (2014), Gokhale, Smith e McCartney (2013b))
	Maturidade	(Jaccheri e Osterlie (2007), Ellis, Purcell e Hislop (2012))
	Domínio	(Buchta et al. (2006), Gokhale, Smith e McCartney (2013b), Rekha e Adinarayanan (2014))
Nenhum Controle*	Número de Contribuidores	(Jaccheri e Osterlie (2007), Ellis, Hislop e Burdge (2017), Pinto et al. (2017), Papadopoulos, Stamelos e Meiszner (2013))
	Issue Tracker	(Ellis, Hislop e Burdge (2017), Holmes et al. (2014), Ferreira et al. (2018a))
	Comunidade Ativa	(Shibuya e Tamai (2009), Ellis, Purcell e Hislop (2012), Nascimento (2017), Pinto et al. (2017), Balali et al. (2018))
	Projeto Ativo	(Pinto et al. (2017), Ellis, Hislop e Burdge (2017), Ferreira et al. (2018a))
	Aceitação de Contribuidor	(Jaccheri e Osterlie (2007), Ellis, Purcell e Hislop (2012), Papadopoulos, Stamelos e Meiszner (2013), Ellis, Hislop e Burdge (2017))
	Principais Contribuidores	(Hsieh et al. (2013), Morgan e Jensen (2014), Holmes et al. (2014), Pinto et al. (2017))

4.3 UM CATÁLOGO DE CRITÉRIOS DE SELEÇÃO DE PROJETOS DE CÓDIGO ABERTO

A Figura 4.1 apresenta os dez critérios de seleção usados nesta pesquisa, classificados segundo o *nível de controle* que o professor deseja exercer sobre o projeto FLOSS. Os critérios de seleção associados a *Controle Interno* também podem ser usados por professores que optaram por *Nenhum Controle*. Os critérios foram documentados na forma de um *catálogo de critérios de seleção de projetos FLOSS*. A Tabela 4.3 apresenta o formato seguido na descrição de cada critério de seleção.

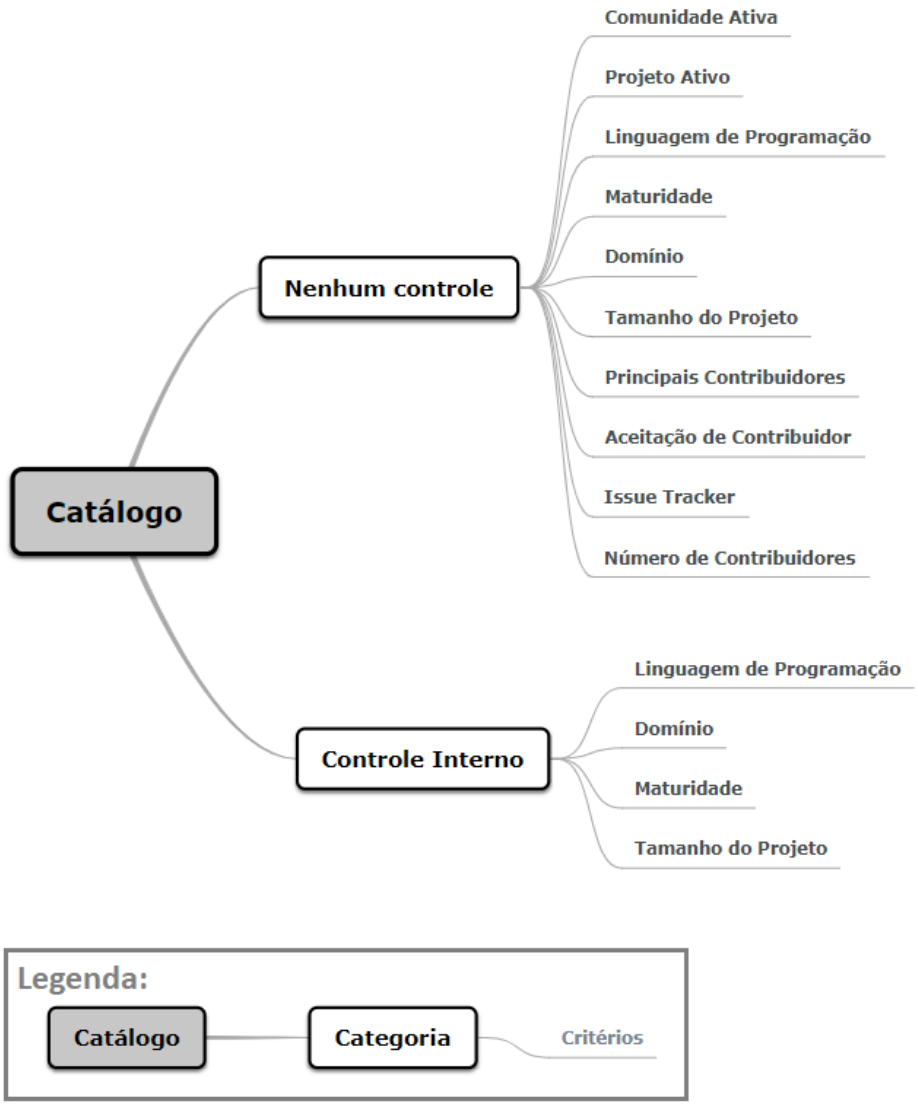

Figura 4.1 Critérios de seleção de projetos FLOSS usados nesta pesquisa.

Tabela 4.3 Formato para Descrição de Critério de Seleção

Elemento	Descrição
Nome	Nome intuitivo para o critério de seleção.
Intenção	Propósito do critério de seleção.
Categoria	Classificação do uso do critério segundo o nível de controle sobre o projeto (Controle Interno e/ou Sem Controle).
Motivação	Motivação para uso do critério na seleção de projetos.
Aplicabilidade	Quando usar o critério.
Consequências	Consequências de usar o critério.
Operacionalização	Breve descrição de solução.
Implementação	Descrição de uma alternativa para implementação do critério.
Usos Conhecidos	Alguns usos do critério a partir da revisão da literatura.

4.3.1 Linguagem de Programação

Nome: `Linguagem de Programação`.

Intenção: Selecionar projetos implementados em uma linguagem de programação específica.

Categoria: Controle Interno e Sem Controle.

Motivação: Um professor deseja utilizar um projeto FLOSS hospedado no *GitHub* para trabalhar com atividades de manutenção de software com seus alunos. Milhares de projetos estão disponíveis neste repositório, mas o professor deseja utilizar projetos que foram implementados em linguagens de programação já utilizadas por seus alunos anteriormente.

Esse problema pode ser resolvido usando o critério `Linguagem de Programação`. Se o projeto usar uma linguagem de programação conhecida pelos alunos, a falta de familiaridade com a linguagem será um obstáculo a menos a ser superado (ELLIS; PURCELL; HISLOP, 2012).

Aplicabilidade: Use o critério `Linguagem de Programação` para filtrar projetos FLOSS implementados em uma linguagem de programação específica ou que possuem a maior parte de seu código implementado em uma dada linguagem.

Consequências: A seleção baseada em `Linguagem de Programação` retorna projetos em que linguagem de programação escolhida pelo professor ou estudante é a linguagem predominante. Por outro lado, para projetos implementados com mais de uma linguagem de programação, a busca de projetos por `Linguagem de Programação` retornará apenas projetos em que tal linguagem de programação é predominante. Isso pode ser considerado uma limitação. Quando a busca for realizada tendo como parâmetro uma linguagem de

programação popular e muito utilizada, um número muito grande de projetos pode ser retornado, e outro critério pode ajudar a refinar a busca.

Operacionalização: A operacionalização do critério `Linguagem de Programação` é simples, em geral, por meio de serviços de busca da plataforma que hospeda os repositórios de software ou por consulta via API disponibilizada para programadores.

Implementação: A consulta via API V3 do *GitHub*[3] utiliza um *Objeto Linguagem* para representar a linguagem de programação encontrada nos repositórios.

Usos Conhecidos: Rekha e Adinarayanan (2014) reportam a seleção de projetos feita por alunos em repositórios como *SourceForge*, *GitHub* ou *GoogleCode*, usando o critério `Linguagem de Programação` para selecionar projetos implementados em linguagens conhecidas pelos alunos (C, C ++, Java). Na mesma linha, Gokhale, Smith e McCartney (2012) relatam ao uso do critério `Linguagem de Programação` na seleção inicial de projetos em vários repositórios de projetos FLOSS para identificar entre 50 e 60 projetos interessantes, em linguagens conhecidas pelos alunos de ES. Papadopoulos, Stamelos e Meiszner (2013) relatam um cenário em que alunos poderiam escolher projetos FLOSS que melhor atendessem às suas necessidades. Além de considerar os critérios de `Maturidade` e `Projeto Ativo`, os alunos deveriam usar `Linguagem de Programação` para indicar linguagem de programação com a qual tivessem maior familiaridade.

4.3.2 Número de Contribuidores

Nome: `Número de Contribuidores`.

Intenção: Seleção de projetos FLOSS com base em um número específico de contribuidores, ou em um intervalo especificado (número mínimo e máximo de contribuidores).

Motivação: Considere que dois professores desejam utilizar um projeto FLOSS hospedado no *GitHub* para ensinar Evolução de Software para seus alunos. Um dos professores está procurando um projeto com poucos contribuidores (até 20 contribuidores), pois ele acredita que quanto maior o número de contribuidores, maior o número de problemas devido ao esforço de comunicação. Ele chegou a essa conclusão baseado no trabalho de Seo et al. (2014), que relatam que as dependências entre os membros da equipe podem aumentar quando o tamanho da equipe é grande, e essas dependências são suscetíveis a problemas. Já outro professor possui uma opinião contrária, e acredita que os contribuidores possam ajudar seus alunos, tirando possíveis dúvidas, e está procurando por projetos com mais de 200 contribuidores. Ele se baseia no estudo de Ellis, Hislop e Burdge (2017), que relatam que os contribuidores são ótimos recursos tanto para professores como para estudantes, para aprenderem sobre um novo projeto, sua cultura e normas. Esse problema poderá ser resolvido usando o critério `Número de Contribuidores` para selecionar projetos com

[3]Versão usada nesta pesquisa

base no número de contribuidores que cada professor definir.

Aplicabilidade: Use o critério `Número de Contribuidores` quando for selecionar projetos que possuem um número de contribuidores específico, ou um número de contribuidores entre um intervalo estabelecido (mínimo-máximo).

Consequências: O critério de seleção `Número de Contribuidores` apresenta ao menos uma desvantagem: O critério é quantitativo e conta todos os usuários que já contribuíram com o projeto, independente do tipo de contribuição. dessa forma conta apenas o número de pessoas que realizaram *commit* no projeto, não sendo verificado portanto se determinado usuário fez somente um *commit* ou vários.

Implementação: O critério `Número de Contribuidores` pode ser implementado com a API V3 do *GitHub*, que possui o objeto *Repository*. Um Repositório contém o conteúdo de um projeto, incluindo a lista de colaboradores associados ao repositório, cujo tamanho pode ser calculado para fornecer o total de colaboradores.

Usos Conhecidos: Pinto et al. (2017) reportaram que professores recomendaram a seus alunos a seleção de um projeto grande (`Tamanho do Projeto`), não descontinuado (`Projeto Ativo`) e com poucos contribuidores (`Número de Contribuidores`). Ellis, Hislop e Burdge (2017) descreve uma abordagem para guiar um contribuidor em potencial na seleção de um projeto HFOSS (*Humanitarian Free and Open Source Software*) e, dentre os critérios apresentados, encontra-se `Número de Contribuidores`. Jaccheri e Osterlie (2007) também utilizaram o critério `Número de Contribuidores` para selecionar, de uma lista de projetos FLOSS conhecidos, projetos com mais de 10 e menos de 50 desenvolvedores ativos. Papadopoulos, Stamelos e Meiszner (2013) reporta que seus alunos poderiam escolher livremente um projeto FLOSS, desde que houvesse atividade no projeto (`Projeto Ativo`) e possibilidade de *feedback* para o aluno (`Comunidade Ativa`). Na estratégia recomendada, o critério `Número de Contribuidores` foi utilizado em conjunto com outros critérios (lista de discussão ativa / fórum e um *feed* atualizado na página principal do projeto).

4.3.3 Aceitação de Contribuidor

Nome: `Aceitação de Contribuidor`.

Intenção: Retornar projetos FLOSS que possivelmente aceitam contribuição de um membro externo ao projeto.

Motivação: Um professor realizou a leitura do trabalho de Nascimento (2017), que afirma que projetos FLOSS representam uma experiência apropriada do mundo real e, com base nisso, decidiu utilizar um projeto FLOSS hospedado no *GitHub* para ensinar construção de software para seus alunos. Seu objetivo foi proporcionar uma oportunidade

para que os alunos trabalhem no código-fonte do projeto, realizem alterações, e enviem o código de volta ao projeto, para que este seja revisado e possivelmente aceito pela comunidade do projeto. De acordo com Ellis, Purcell e Hislop (2012), os professores devem identificar maneiras pelas quais os alunos podem fazer contribuições úteis para os projetos. Sendo assim, esse problema pode ser resolvido selecionando os projetos que possuem indícios de que novos contribuidores serão aceitos.

Aplicabilidade: Use o critério `Aceitação de Contribuidor`, quando desejar obter projetos FLOSS com indícios de que um novo contribuidor será aceito na comunidade.

Consequências: O critério `Aceitação de Contribuidor` tem os seguintes benefícios e desvantagens: i) Ele retorna projetos com indícios de que *commits* realizados por contribuidores externos a comunidade do projeto, serão aceitos. ii) Os projetos retornados possuem apenas indícios de aceitação, mas a aceitação do *commit* de fato, não pode ser garantida. iii) A busca por projetos é realizada com base em indicadores que podem ser adequados para identificar projetos que aceitam novos contribuidores. Entretanto, podem existir outros indicadores úteis ainda não identificados.

Operacionalização: Indícios de que o projeto aceita um novo contribuidor são: i) Links para uma página de descrição do projeto; ii) Informações sobre formas de se envolver no projeto; e iii) Informações sobre como se conectar à comunidade (ELLIS; HISLOP; BURDGE, 2017). Na página do *GitHub*, nas abas correspondentes a *Issues* do projetos, encontra-se a seguinte informação: *"You can also take a look at the Open Source Guide. Issues labeled help wanted can be good first contributions"*. Sendo assim, poderá ser realizada uma busca por *labels* específicos como: ***help wanted e/ou good first issue***.

Implementação: Para implementar critério `Aceitação de Contribuidor`, pode-se utilizar a API V3 do *GitHub*, que possui o objeto *Label*. Um *Label* categoriza *Issues* ou *Milestones* de um determinado repositório.

Usos Conhecidos: Alguns trabalhos relataram o uso do critério `Aceitação de Contribuidor` na seleção de projetos FLOSS para uso na EES. Jaccheri e Osterlie (2007) usaram o critério `Aceitação de Contribuidor` em uma lista de projetos FLOSS conhecidos para selecionar projetos cuja comunidade permite a entrada de novos contribuidores. Papadopoulos, Stamelos e Meiszner (2013) recomendaram aos alunos que olhassem a *tag Help Wanted* em plataformas que hospedam o projeto FLOSS, pois há muitos projetos em busca de desenvolvedores. Ellis, Purcell e Hislop (2012) apresentaram um modelo de avaliação de projetos FLOSS, com recomendação para observar se o projeto possui um ponto de partida facilmente identificável para novos participantes que descreve como novos contribuidores podem se juntar à comunidade e fazer contribuições.

4.3.4 Issue Tracker

Nome: `Issue Tracker`.

Intenção: Selecionar projetos FLOSS que possuem *Issue Tracker* (rastreamento de problemas) disponível e que mostra a localização das *Issues* em aberto.

Motivação: Considere que um professor escolheu aleatoriamente no *GitHub* um projeto FLOSS para que seus alunos desenvolvam habilidades de evolução de software. No entanto, o professor nunca teve contato com o código-fonte do projeto escolhido. Ele desconhece as funcionalidades do sistema já implementadas e/ou funcionalidades que poderão ser trabalhadas por seus alunos, mas lembra que, de acordo com Liao et al. (2018), o *GitHub* fornece um sistema de *Issue Tracker* para cada projeto, no qual a equipe de desenvolvimento define e relata problemas e funcionalidades do sistema – o que pode ajudar seus alunos a participar mais facilmente do desenvolvimento de projetos de código aberto. A *Issue* é um lugar para discutir idéias, aprimoramentos, tarefas e erros de um projeto. Como a aprendizagem do aluno é melhor suportada por projetos que possuem um roteiro que inclui o desenvolvimento de novos recursos, e um processo para identificar como novos recursos são priorizados (ELLIS; PURCELL; HISLOP, 2012), o problema desse professor poderá ser resolvido selecionando os projetos que possuem *Issue Tracker* disponível. As *Issues* podem fornecer informações sobre o andamento de um projeto, listando aquelas que se encontram em aberto, ou seja, problemas que ainda não foram resolvidos. Podem também trazer informações úteis sobre o que foi resolvido, como foi resolvido e contribuidores envolvidos na discussão e resolução.

Aplicabilidade: Use o critério `Issue Tracker`, quando: i) Desejar obter projetos FLOSS que possuem Issues abertas; ii) Desejar obter o link contendo as *Issues* em aberto dos projetos FLOSS disponíveis; iii) Desejar estudar a solução de problemas enfrentados pelos contribuidores, analisando informações relacionadas as *issues* abertas ou fechadas.

Consequências: O critério `Issue Tracker` tem os seguintes benefícios e desvantagens: i) Ele retorna todos os projetos que possuem *Issues* em aberto. Esse critério é utilizado como parâmetro de busca para localizar todos os projetos que possuem *Issues* com o status OPEN. ii) A listagem será feita baseada no status disponível no repositório. Não é possível garantir se, de fato, aquela *Issue* ainda não foi implementada. iii) Alguns projetos são hospedados no *GitHub*, mas mantêm seu sistema de *Issue Tracker* fora dele (BISSYANDÉ et al., 2013). iv)

Implementação: Para implementar o critério `Issue Tracker`, a API V3 do *GitHub*, que possui o objeto *Issue*, pode ser utilizada. Por padrão, a criação de um projeto no *GitHub* é acompanhada pela criação de um repositório com controle de versões e um *Issue Tracker* correspondente para o projeto. No entanto, o desenvolvedor pode desativar a *Issue Tracker*, tornando esse recurso inacessível aos usuários (BISSYANDÉ et al., 2013). A

Issue possui dois estados possíveis *OPEN* (ABERTO) e *CLOSED* (FECHADO). Sendo assim, pode ser verificada a existência e o status das *Issues* de todos os projetos, retornando aquelas que possuem status ABERTO. Essa verificação pode ser feita através do objeto *Issue*.

Usos Conhecidos: Há diversos trabalhos, não relacionados à educação em engenharia de software, que realizam estudos com base no *Issue Tracker* disponível nos projetos (BISSYANDÉ et al., 2013), (WANG et al., 2015) e (BELLOMO et al., 2016). Steinmacher et al. (2013) realizaram um estudo de caso para verificar como as primeiras interações nas listas de discussão e na *Issue Tracker* influenciam a decisão dos recém-chegados de contribuir para projetos FLOSS. O uso do critério *Issue Tracker* pode trazer projetos com listas de discussão que despertem a vontade de alunos e professores de contribuir com tais projetos. No contexto da Educação em Engenharia de Software, Ellis, Hislop e Burdge (2017) reportaram o uso do critério *Issue Tracker* na seleção de projetos. Holmes et al. (2014) relataram que, no programa Projetos de Código Aberto de Graduação (UCOSP), os projetos FLOSS em estudo deveriam manter um *Issue Tracker* ativo, e que os projetos mais eficazes também deveriam identificar em suas *Issues* problemas específicos para iniciantes. Morgan e Jensen (2014) relatam que os alunos escolheram os próprios projetos para trabalhar, mas que deveriam usar o critério *Issue Tracker* para escolher projeto com *Issue Tracker* ativa. Ferreira et al. (2018a) relatam que a escolha de tarefas para os alunos trabalharem foi feita com base nas *Issues* do projeto, conforme relatou um aluno: "normalmente, os alunos iam atrás de tarefas no Issue Tracker".

4.3.5 Comunidade Ativa

Nome: `Comunidade Ativa`.

Intenção: Selecionar projetos FLOSS com indícios de que sua comunidade é ativa.

Motivação: Considere que um professor deseja utilizar projetos FLOSS para que seus alunos tenham a oportunidade de ver como os princípios teóricos de programação são aplicados em contextos de mundo real. O professor se interessou pelo trabalho de Morgan e Jensen (2014), que relatou que os alunos deveriam fazer ao menos uma contribuição para o projeto do Ubuntu[4]. Dessa forma, ele também planeja solicitar que seus alunos façam ao menos uma contribuição para um projeto escolhido. No entanto, tal professor não costuma contribuir com projetos hospedados no *GitHub*, e possivelmente terá dificuldades para sanar possíveis dúvidas de seus alunos sobre como contribuir com o projeto.

Para Balali et al. (2018), uma maneira bem conhecida de ajudar os recém-chegados a superar as barreiras iniciais de contribuição é através da orientação. Dessa forma, esse problema poderá ser resolvido selecionando projetos que possuem uma comunidade ativa de usuários para dar suporte aos alunos, caso necessário. Esse suporte indica uma capacidade de resposta dos desenvolvedores aos usuários (alunos que desejam contribuir

[4]⟨https://github.com/ubuntu⟩

com o projeto) (ELLIS; PURCELL; HISLOP, 2012).

Aplicabilidade: Use o critério `Comunidade Ativa`, quando: i) Desejar obter projetos FLOSS com indícios de que sua comunidade é ativa; ii) Desejar obter projetos FLOSS em que dúvidas de alunos possam ser sanadas pela comunidade do projeto.

Consequências: O critério `Comunidade Ativa` tem os seguintes benefícios e desvantagens: i) De acordo com Ellis, Purcell e Hislop (2012), embora seja difícil avaliar o nível de atividade de uma `Comunidade Ativa`, há meios de identificá-lo. ii) Ele retorna projetos com indícios de que novatos (novos desenvolvedores) poderão ter suas dúvidas respondidas, considerando que há comentários e desenvolvedores ativos na lista, propensos a interagir com eles. iii) Os projetos retornados possuem apenas indícios de comunidade ativa, mas o suporte da comunidade aos novatos, de fato, não pode ser garantido. iv) Em alguns períodos, um projeto pode ter comunidade ativa e em outros não.

Operacionalização: O critério `Comunidade Ativa` pode considerar comentários na lista de *Issues* e considerar uma data/período de interesse. A seleção retornará projetos que possuem indícios de que sua comunidade é ativa, com base no período/data indicado pelo professor. A busca verifica se, a partir de determinada data, existe um histórico de comentários nas *Issues*.

Implementação: A API V3 do *GitHub* pode ser utilizada para implementar o critério `Comunidade Ativa`. O objeto *Issue* possui a lista de *Issues* e seus comentários, e a existência de comentários nas *Issues* após determinada data pode ser testada.

Usos Conhecidos: O uso do critério `Comunidade Ativa` para a seleção de projetos em que se deseja realizar uma contribuição é muito importante, em especial no cenário classificado como *Nenhum Controle* (NASCIMENTO, 2017), ou seja, quando o aluno deve contribuir com o projeto diretamente no *GitHub*. Shibuya e Tamai (2009) afirmam que uma das dificuldades para iniciar a contribuição em um projeto é quando não existe nenhuma resposta dos membros da comunidade para as perguntas feitas. Ellis, Purcell e Hislop (2012) e Pinto et al. (2017) afirmam que, para utilizar projetos de código aberto na disciplina de engenharia de software, os projetos FLOSS devem ser mantidos por uma `Comunidade Ativa` de desenvolvedores de software que esteja envolvida no desenvolvimento contínuo do projeto. Ainda de acordo com Pinto et al. (2017), o *feedback* rápido dos mantenedores é crucial, já que a maioria dos cursos dura apenas um semestre. Esse *feedback* da comunidade também é apontado por Balali et al. (2018), como um auxílio para que os desenvolvedores encontrem uma tarefa que possam implementar e descubram como contribuir com o projeto (`Aceitação de Contribuidor`).

4.3.6 Tamanho do Projeto

Nome: `Tamanho do Projeto`.

Intenção: Retornar projetos FLOSS que estejam entre um intervalo especificado relacionado ao número de linhas de código.

Motivação: Considere que o professor deseja fazer um *fork* de um projeto FLOSS para que seus alunos desenvolvam habilidades de manutenção, em contato com projetos de grande porte, já que atividades de manutenção de software normalmente requerem mais esforço do que atividades de desenvolvimento (POSTEMA; MILLER; DICK, 2001). No entanto, ele está tendo dificuldade de escolher o projeto adequado, pois, de acordo com Jaccheri (JACCHERI; OSTERLIE, 2007), entrar em um projeto composto por milhares de arquivos fontes e linhas de código requer uma maior habilidade de programação. O problema desse professor poderá ser resolvido selecionando os projetos de acordo com o seu tamanho – número de linhas e de classes, por exemplo. Pode-se considerar que projetos de código aberto maiores tendem a ser mais difíceis de compreender, e o tamanho do projeto pode exigir maior ou menor tempo e conhecimento dos alunos (ELLIS; HISLOP; BURDGE, 2017).

Aplicabilidade: Use o critério `Tamanho do Projeto`, quando desejar obter projetos com base no número mínimo e/ou máximo de elementos do software propriamente dito (linhas de código, de comentários, de classes). Em geral, há correlação entre valores de diversas métricas de tamanho e a baseada no número de linhas de código (LOC). Desse modo, usar LOC pode ser a forma mais simples para lidar com o tamanho do projeto.

Consequências: O critério `Tamanho do Projeto` tem os seguintes benefícios e desvantagens: i) Ele retorna projetos com número de linhas de código em um intervalo especificado. Esse critério é utilizado como parâmetro de busca para localizar, dentre os disponíveis, aqueles que estejam no intervalo em número especificado, correspondente ao número de linhas. ii) Outros indicadores podem representar Tamanho do Projeto. No entanto, foi considerado apenas o número de linhas de código por ser representativo para aspectos relacionados a tamanho.

Operacionalização: Pode-se utilizar uma ferramenta de análise estática para contabilizar as linhas de código no projeto ou utilizar informações disponibilizadas pela plataforma que hospeda o software.

Implementação: A API da plataforma OpenHub pode ser utilizada para trazer informação relacionada ao número de linhas de código.

Usos Conhecidos: Rekha e Adinarayanan (2014) utilizaram `Tamanho do Projeto` para selecionar um projeto com mais de 3.000 e menos de 5.000 linhas de código. Morgan e Jensen (2014) observaram que o valor do `Tamanho do Projeto` considerado na seleção (no caso relatado, seleção do projeto Ubuntu) foi um obstáculo para os alunos, que não foram capazes de encontrar uma tarefa simples e interessante para trabalhar. Ellis, Purcell e Hislop (2012) consideram o critério `Tamanho do Projeto`, e utilizaram métricas

(LOC, KLOC, MLOC) para selecionar projetos FLOSS. Gokhale, Smith e McCartney (2013b), com base em sua experiência, recomendam que a seleção com base no `Tamanho do Projeto` considere um valor entre 5.500 e 10.500 linhas de código, de modo que os projetos sejam complexos, para permitir trabalhar habilidades de compreensão e aprimoramento, mas não tão complexos, para se adequar ao tempo e nível de conhecimento dos alunos.

4.3.7 Maturidade

Nome: `Maturidade`.

Intenção: Selecionar projetos FLOSS com base em seu número de lançamentos (*releases*).

Motivação: Um professor deseja colocar seus alunos de 3o. semestre em contato com projetos em evolução contínua que ofereçam oportunidades para desenvolver habilidades de comunicação trabalhar com técnicas e tópicos relacionados a manutenção de software. Com isso, ele planeja selecionar um projeto FLOSS hospedado no *GitHub* para que seus alunos realizem contribuições, mas ele está tendo dificuldade de escolher um projeto adequado.

Esse problema pode ser resolvido considerando a quantidade de *releases* do projeto disponíveis como medida para um projeto em evolução. Ellis, Hislop e Purcell (2013) relatam que para utilização em sala de aula, o projeto FLOSS deve ter pelo menos uma versão de produção estável. Projetos com poucos lançamentos podem não ter maturidade suficiente para apoiar a aprendizagem dos alunos. No entanto, para Jaccheri e Osterlie (2007), se um projeto FLOSS estiver maduro, bem testado e perto de um lançamento, muitas das tarefas restantes podem ser complexas. Se o objetivo é contribuir para um projeto, o professor deve, pelo menos, estar ciente destas possíveis forças e dificuldades.

Aplicabilidade: Use o critério `Maturidade`, quando: i) Possuir interesse em utilizar projetos com um histórico consistente de versões e *releases*, o que pode indicar que as tarefas restantes são complexas e exige um maior conhecimento dos alunos; ii) Desejar um projeto com um atividade de desenvolvimento constante, com várias *releases*, o que pode indicar que o projeto FLOSS oferece oportunidades de estudo e exemplos para serem analisados em sala de aula; iii) Desejar obter projetos FLOSS estáveis, mas com poucas *releases* para que alunos iniciantes contribuam com o projeto;

Consequências: O critério `Maturidade` tem os seguintes benefícios e desvantagens: i) A maturidade do projeto é interpretada com base na quantidade de *releases*, mas podem existir outros indicadores úteis e mais representativos. ii) O professor define sua noção de "Maturidade", informando a quantidade de *releases* de um projeto que ele julga desejável para uso na disciplina. iii) Os projetos retornados possuem apenas indícios de maturidade, mas a maturidade de fato, não pode ser garantida. iv) Ele retorna projetos

com uma quantidade de *releases* específica. Esse critério é utilizado como parâmetro de busca, para localizar projetos que possuem a quantidade de *releases* informadas.

Implementação: Para implementar o critério `Maturidade`, pode ser utilizada a API V3 do *GitHub* que possui o objeto Release. Dessa forma, é possível observar quantos lançamentos cada projeto possui, e extrair informações relevantes relacionadas a cada um deles.

Usos Conhecidos: Ellis, Purcell e Hislop (2012) apresenta `Maturidade` como um critério de seleção de projetos FLOSS. Jaccheri e Osterlie (2007) argumenta que selecionar um projeto FLOSS com pouca *Maturidade* pode ter a desvantagem de ser significativamente diferente de um projeto FLOSS que represente um exemplo do mundo real.

4.3.8 Domínio

Nome: `Domínio`.

Intenção: Selecionar projetos FLOSS de acordo com domínios especificados.

Motivação: Um professor dividiu a sua turma em 4 grupos, e ele deseja utilizar projetos FLOSS hospedados no *GitHub* para trabalhar tópicos relacionados à manutenção de software com seus alunos. Ele acredita que o domínio da aplicação, ou seja, o foco do projeto, pode ter um impacto na aprendizagem do aluno, sendo mais fácil a compreensão, por exemplo, de projetos relacionados a jogos do que projetos relacionados a física quântica. Ellis, Purcell e Hislop (2012) também afirmam que alguns projetos FLOSS requerem muito conhecimento de domínio por parte dos alunos. Sendo assim, é possível auxiliar o professor na seleção dos projetos de acordo o domínio do projeto que cada grupo possui mais afinidade.

Aplicabilidade: Use o critério `Domínio`, quando desejar obter projetos com domínio específico.

Consequências: O critério `Domínio` tem os seguintes benefícios e desvantagens: i) Ele retorna projetos com os domínios especificados, de acordo com o corpo da descrição de cada projeto. Se o domínio não estiver claro na descrição do projeto, este pode não ser retornado.

Implementação: Através da API v3 do *GitHub* é possível acessar o objeto Project. Esse objeto possui um campo que armazena a descrição do projeto. Com isso basta buscar todos os projetos que contêm, na descrição do projeto, o domínio especificado.

Usos Conhecidos: Rekha e Adinarayanan (2014) orientou seus alunos a usar o critério `Domínio` de modo que diferentes grupos fossem expostos a diferentes domínios. Buchta et al. (2006) deram atenção especial ao critério `Domínio` durante a seleção de projetos, para que projetos mais intuitivos e fáceis de entender fossem usados pelos alunos em tarefas de manutenção. Gokhale, Smith e McCartney (2013b) usaram 17 projetos Java de diferentes domínios (Jogos, Artes, Indexação, Cliente/Servidor, etc.) e observaram características de tornam os projetos adequados para ensino de manutenção, por exemplo, relevância do domínio, propósito claro e inclinação industrial.

4.3.9 Projeto Ativo

Nome: `Projeto Ativo`.

Intenção: Selecionar projetos FLOSS que possuem indícios de que estejam ativos, com contribuições recentes (por exemplo: contribuições realizadas nos últimos 30 dias).

Motivação: Um professor tem algumas dificuldades para ensinar manutenção de software, devido à falta de motivação dos seus alunos. Após tentar utilizar diversos métodos sem sucesso, ele pretende utilizar projetos FLOSS em suas aulas, mas como ele possui nenhuma ou pouca experiência com projetos FLOSS, ele não sabe como escolher o projeto adequado. Ele foi informado pelas pessoas que contribuem com projetos FLOSS que, se ele deseja que os alunos contribuam com o projeto hospedado no *GitHub*, o ideal é que o projeto esteja razoavelmente ativo. Esse problema pode ser resolvido selecionando os projetos com indícios de atividade, considerando a abordagem de Ellis, Hislop e Burdge (2017), que afirma que o número de *commits* pode ser usado como um indicador para atividade do projeto. Pouca ou nenhuma atividade ao longo de um ano, por exemplo, pode indicar que o projeto não está ativo.

Aplicabilidade: Use o critério `Projeto Ativo`, quando: Desejar obter projetos FLOSS com indícios de que tal projeto esteja ativo em termos de frequência de contribuições.

Consequências: O critério `Projeto Ativo` tem os seguintes benefícios e desvantagens: i) Ele retorna projetos que possuem indícios de atividade. Os indícios são baseados na frequência de contribuições, calculada como a quantidade de *commits* realizados nos últimos 30 dias. ii) Os projetos retornados possuem apenas indícios de atividade baseado nos *commits*, mas o conteúdo dos *commits* não serão avaliados. iii) Será utilizado o *commit* como indicador de projeto ativo por que acreditar que o mesmo é adequado para a identificação de projetos ativos, mas podem existir outros indicadores úteis. iv) De acordo com Ellis, Purcell e Hislop (2012), dez *commits* por mês são minimamente aceitáveis e 30 *commits* ou mais por mês, em média, seriam favoráveis ao uso de um projeto FLOSS na EES. No entanto, como essa quantidade pode ser relativa, dependendo da opinião de cada professor, ele poderá informar a quantidade de *commits* que ele julgar mais representativa.

Implementação: A API v3 do *GitHub* oferece o objeto *Commit*, sendo possível observar a data de criação de cada *commit*, e calcular o número de *commits* feitos nos últimos 30 dias.

Usos Conhecidos: Pinto et al. (2017) relatam que os professores que participaram de seu estudo recomendaram que os alunos buscassem projetos ativos para contribuir, ou seja, usassem o critério `Projeto Ativo`.

4.3.10 Principais Contribuidores

Nome: `Principais Contribuidores`.

Intenção: Permitir que professores ou alunos possam selecionar um projeto após listar e, eventualmente, contactar, seus principais contribuidores. Trata-se de uma consulta que pode apoiar o professor ou o aluno na seleção de um projeto FLOSS após identificar seus principais contribuidores e entrar em contato com ele(s).

Motivação: Segundo DuBois et al. (2002), a atribuição de mentores a novos membros de comunidades provou ser eficaz para ajudá-los a superar desafios. Logo, para aumentar a chance de apoio da comunidade às dúvidas de seus alunos, o professor que conhecer os principais contribuidores do projeto para que os alunos possam, eventualmente, entrar em contato com algum deles, ou até mesmo selecionar um projeto em que exista algum contribuidor conhecido.

Aplicabilidade: Este critério difere dos demais. Ele é aplicável quando o professor deseja obter projetos cujo nível de controle é "Nenhum Controle". Com isso o professor poderá verificar: i) Se dentre os principais contribuidores retornados, possui um contribuidor conhecido; ii) Se os principais contribuidores retornados, falam o mesmo idioma dos alunos.

Consequências: O critério `Principais Contribuidores` tem os seguintes benefícios e desvantagens: i) Ele retorna uma lista das pessoas que mais realizaram *commits* no projeto. Não será verificado o conteúdo do *commit*, apenas a quantidade. ii) Os contribuidores serão ordenados e exibidos apenas os 10 primeiros, com base na quantidade de *commits* desde a criação do projeto.

Operacionalização: Verificar os contribuidores do projeto e apresentar a lista de quem mais contribuiu com o projeto desde sua criação.

Usos Conhecidos: Um professor no estudo de Ferreira et al. (2018a) relatou que costuma fazer contato com pessoas-chave de alguns projetos, perguntando se os alunos da disciplina podem fazer contato.

4.4 CONSIDERAÇÕES FINAIS

Dez critérios usados por professores na seleção de projetos FLOSS foram identificados sistematicamente, classificados e catalogados em um formato estruturado, expandindo o conhecimento sobre tais critérios e seu uso.

Os professores de Engenharia de Software podem se beneficiar desses resultados usando-os para norteá-los no processo de seleção do(s) projeto(s) FLOSS mais adequado(s) para sua disciplina, o que pode trazer experiências positivas que estimulem o uso de projetos FLOSS no processo de ensino-aprendizagem de ES.

Embora os critérios tenham sido identificados a partir dos estudos primários filtrados no mapeamento sistemático, é possível que existam outros critérios que poderiam ser automatizados. É importante destacar também que a maioria dos trabalhos encontrados consistiu de relatórios de experiência que podem não ter capturado todo o espectro da seleção de projetos.

O conjunto de critérios de seleção proposto é uma versão preliminar, com critérios gerais que podem ser utilizados na seleção de projetos para uso pedagógico em diferentes sub-áreas de engenharia de software. Não há critérios específicos para uso em sub-áreas específicas. O catálogo de critérios de seleção de projetos FLOSS organiza e apresenta informações sobre os dez critérios identificados. Cada critério de seleção foi detalhado em termos de sua motivação, aplicabilidade, operacionalização, implementação e usos conhecidos, dentre outros aspectos. A versão do catálogo apresentada é certamente preliminar e espera-se que outros critérios de seleção de projetos FLOSS possam ser documentados e agregados no futuro.

Capítulo

5

FLOSSSEARCH.EDU: PLATAFORMA WEB DE SELEÇÃO DE PROJETOS FLOSS

Este Capítulo apresenta FlossSearch.Edu, uma ferramenta web projetada para apoiar professores e alunos na seleção baseada em critérios de projetos FLOSS em um repositório populado aleatoriamente com projetos clonados de GitHub. Os critérios implementados na ferramenta são aqueles documentados no catálogo de critérios de seleção apresentado no Capítulo 4.3.

5.1 VISÃO GERAL

A ferramenta FlossSearch.Edu é uma aplicação Web [1] open-source cujo código fonte está hospedado no GitHub[2] sob a Licença Pública Geral GNU (GNU General Public License - GPL).

São funcionalidades da ferramenta *FlossSearch.Edu*:

1. Filtrar os projetos FLOSS de acordo com até 10 critérios selecionados especificados pelo usuário.

2. Permitir que os usuários autenticados façam comentários sobre um determinado projeto, possibilitando o registro de possíveis experiências com aquele projeto. Esses comentários podem ser visualizados por qualquer usuário.

3. Permitir que usuários autenticados avaliem um projeto com 1 a 5 estrelas, que podem ser usadas para indicar o nível de satisfação em usar aquele projeto específico. A classificação média de cada projeto está disponível para visualização de qualquer usuário.

[1] Disponível online em: ⟨http://191.252.92.63/flosssearch/⟩
[2] Available at ⟨https://github.com/Moara/flosssearch⟩

4. Permitir que os usuários selecionem vários projetos e baixem seus metadados no formato *.csv*. Os metadados incluem nome do projeto, URL, número de linhas de código e descrição.

5. Permitir aos usuários visualizar o catálogo de critérios de seleção online.

O usuário pode alterar o nível desejado de controle do projeto de *Sem Controle* (padrão) para *Controle Interno*. Para cada nível de controle, um conjunto de critérios para ajudar os usuários na seleção do projeto é apresentado.

A ferramenta foi elaborada com o objetivo de apoiar professores e alunos na seleção de projetos FLOSS para uso no SEE, mas pode ser usado no contexto de outros cursos que têm atividades baseadas em projetos ou requerem exemplos realistas. Os estudos encontrados em revisões de literatura recentes (NASCIMENTO; BITTENCOURT; CHAVEZ, 2015; BRITO et al., 2018) relatam que a seleção de projetos FLOSS usados em cursos de engenharia de software é realizada por alunos e professores – os usuários potenciais da *FlossSearch.Edu*.

A maioria dos estudos relata a seleção de projetos realizada pelos alunos. Eles podiam escolher projetos FLOSS quase que livremente, sujeitos a algumas restrições gerais definidas pelo professor, como linguagem de programação, tamanho do projeto, número de desenvolvedores, entre outros (ADINARAYANAN et al., 2014; PINTO et al., 2017).

Outros estudos relatam a seleção de projetos realizada por professores e assistentes de professores. Eles realizam a seleção manual para preparar e avaliar os projetos FLOSS para uso pedagógico (SMITH et al., 2014) e, posteriormente, fornecem aos alunos um projeto ou uma lista de projetos.

5.2 ARQUITETURA DA FERRAMENTA

A FlossSearch.Edu é uma aplicação web desenvolvida na linguagem de programação PHP e com o framework *CodeIgniter* no back-end e Locaweb Style no front-end. A aplicação executa solicitações assíncronas de JavaScript, de modo a evitar atualizações de página ao validar e transferir dados.

Juntamente com a aplicação web, foram desenvolvidos scripts PHP para coletar informações dos repositórios. Houve um foco nos projetos hospedados no GitHub, por se tratar de uma plataforma de hospedagem de projetos de software que tem crescido rapidamente, tornando-se um dos maiores repositórios de projetos de código aberto (CHATZIASIMIDIS; STAMELOS, 2015).

Cada projeto público no GitHub tem um repositório acessível publicamente, que permite acesso a todo o histórico de alterações do código-fonte. As informações públicas disponíveis nesses repositórios podem ser recuperadas por meio de uma Interface de Programação de Aplicativos(*Application Programming Interface* - API) fornecida por GitHub (BAUTISTA; FELIU, 2015). No entanto, essa API tem um limite de 5.000 solicitações por hora, incluindo solicitações de autenticação.

Tal limitação inviabilizou a busca por projetos em tempo real, diretamente no *GitHub*, pois haveria um problema de redução da taxa de coleta e a aplicação seria forçada a parar

até que o limite de solicitação fosse renovado. Para solucionar esse problema, a alternativa adotada foi a criação de um banco de dados local.

Como a API do GitHub não fornece informações sobre o número de linhas de código em um projeto software, optou-se por extrair essa informação do Black Duck Open Hub[3]), um site que coleta métricas sobre o desenvolvimento FLOSS, incluindo linhas de código. Essas informações são extraídas por meio de *web scraping*. Se um projeto indexado por FlossSearch.Edu não tem entrada correspondente no Open Hub, o número de linhas de código em nosso banco de dados é mantido em branco.

Para o processo de coleta de dados, usando a versão atual do FlossSearch.Edu, foram executados os scripts em uma amostra aleatória da base do projeto GitHub e armazenado os resultados em um banco de dados local. Assim, quando o usuário executa a ferramenta, ele está consultando informações previamente armazenadas em um banco de dados local, e não diretamente no GitHub em tempo real, já que o volume de projetos disponíveis no GitHub colocaria um atraso significativo na aquisição dos dados.

5.3 PROJETO DE INTERFACE E INTERAÇÃO COM A FERRAMENTA

A Tabela 5.1 apresenta as decisões do projeto de interface da ferramenta em relação à entrada de dados / valores dos critérios de seleção. A saída de informações sobre os principais contribuidores é mostrada na forma de uma lista com nome, afiliação, quantidade de *commits* e e-mail dos 10 contribuidores mais frequentes.

Ao acessar a ferramenta FlossSearch.Edu, a página principal mostra informações sobre a ferramenta e os critérios implementados (Figura 5.1). Ao escolher a opção *Start selection*, o professor é encaminhado para a tela de seleção de projetos. Um maior detalhamento das categorias e o conjunto de critérios utilizados pode ser visualizado na aba "About"da ferramenta (Figura 5.2 e Figura 5.3).

Ao iniciar a seleção, o usuário deve considerar as duas categorias relacionadas ao nível de controle, *Inside Control* e *No Control*. Em cada categoria, foi fornecido um conjunto de critérios para ajudar os usuários a selecionar o projeto de acordo com o nível de controle desejado sobre os projetos e atividades que serão realizados pelos alunos.

Em seguida, o usuário pode especificar os critérios de seleção desejados (Figura 5.4). Uma pesquisa no banco de dados local é executada e exibe os projetos que possuem as características definidas no painel à direita. Quando mais de um projeto é devolvido ao usuário, é possível selecionar um ou mais projetos e baixar suas informações de metadados em um arquivo *.csv* (Figura 5.5, retângulo verde à esquerda).

Ao clicar no nome de um projeto, são apresentadas informações adicionais, como a classificação do projeto em relação ao nível de satisfação com a utilização do projeto, e a indicação da existência de etiquetas de problemas como *help wanted* ou *good first issue*. Ao clicar no nome de uma etiqueta, o usuário é direcionado para a página de *issues* correspondente.

Outros detalhes do projeto também podem ser visualizados, como página inicial, linguagem de programação, licença do projeto, numero total de contribuidores, número de linhas de código, releases totais, existência de um wiki do projeto, a data de inserção do

[3]Anteriormente conhecido como Ohloh. Disponível em ⟨https://www.openhub.net/⟩

Tabela 5.1 Entrada de Dados

Critério	**Descrição**
Linguagem de Programação	Caixa de seleção (*checkbox*) múltipla, para que o professor escolha uma ou mais linguagens de programação.
Número de Contribuidores	*Range Slider* com controles deslizantes de intervalo personalizado, para que o professor informe o número mínimo e máximo de contribuidores no projeto.
Contribuição	*Radio Button* para que o professor escolha se deseja obter projetos que apresentam indícios que aceitam contribuições.
Issue Tracker	*Radio Button* para que o professor indique se deseja obter projetos que possuem *Issue Tracker* ou não.
Comunidade Ativa	*Input type date* para que o professor insira uma data que ele considere como comunidade ativa.
Tamanho (LOC)	*Range Sliders* com controle deslizante de intervalo personalizado, para que o professor informe o número mínimo e máximo de linhas de código no projeto.
Maturidade	*Ranger Slide*r com controles deslizantes de intervalo personalizado, para que o professor informe o número mínimo e máximo de *releases* do projeto.
Domínio	*Text box*, no qual o professor poderá informar os domínios desejados. O professor poderá informar 0 ou N domínios. Dessa forma é importante que tenha a opção para que seja inserido um novo domínio na busca, se necessário.
Projeto Ativo	Deverá ser disponibilizado um campo para que o professor insira a quantidade de *commits* que ele considera como projeto ativo, e realizar a verificação de quais projetos disponíveis possuem mais do que a quantidade de *commits* informada, considerando os últimos 30 dias.

projeto no banco de dados e a data quando a informação foi atualizada (Fig. 5.7) e a lista dos 10 principais contribuidores do projeto. (Figura 5.8).

A lista dos 10 principais contribuidores pode ser usada para identificar, por exemplo, a presença de um colaborador que fala o mesmo indioma dos alunos ou mesmo se existe um contribuidor conhecido que pode ajudar os alunos com possíveis dúvidas.

Por último, mas não menos importante, o professor também pode ler os comentários feitos por outros usuários sobre sua experiência no uso de um projeto FLOSS (Fig. 5.9).

Essas informações podem ter influência na decisão do professor em usar ou não um projeto em sala de aula.

5.4 CONSIDERAÇÕES FINAIS

Este Capítulo apresentou a ferramenta FlossSearch.Edu, desenvolvida para apoiar a seleção de projetos FLOSS hospedados no GitHub que atendam aos requisitos técnicos e

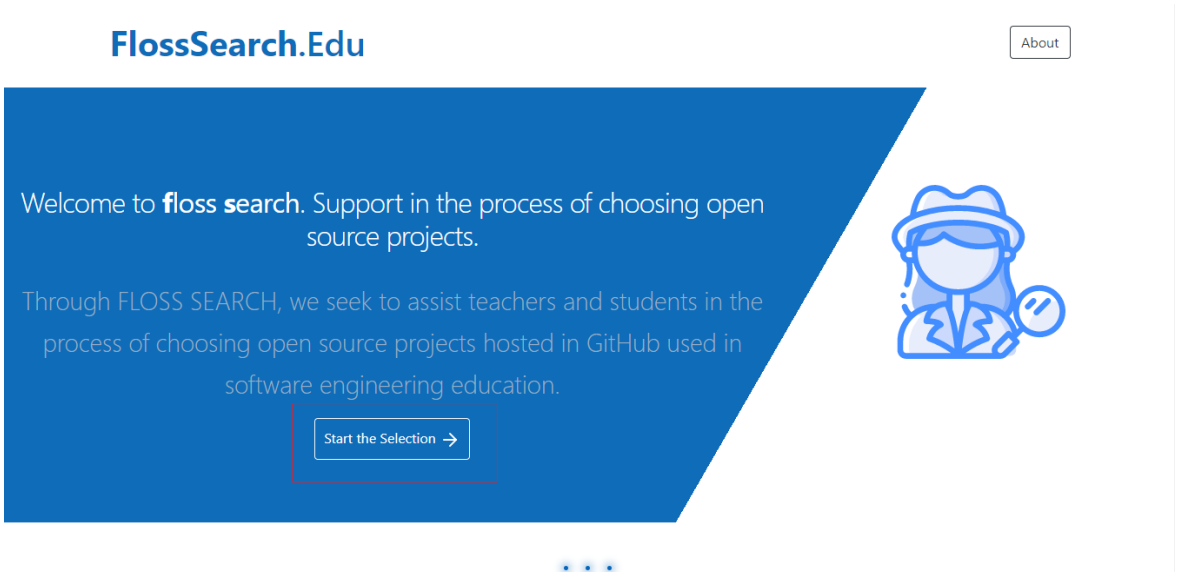

Figura 5.1 Informações Iniciais

sociais associados ao ensino-aprendizagem de tópicos de engenharia de software.

Atualmente, FlossSearch.Edu possui informações para um pequeno conjunto de projetos de software de código aberto. Para o futuro, planeja-se minerar e povoar automaticamente o repositório local com um maior número de projetos, implementados em outras linguagens de programação, e provenientes de outras plataformas, de modo a oferecer mais opções para professores e alunos.

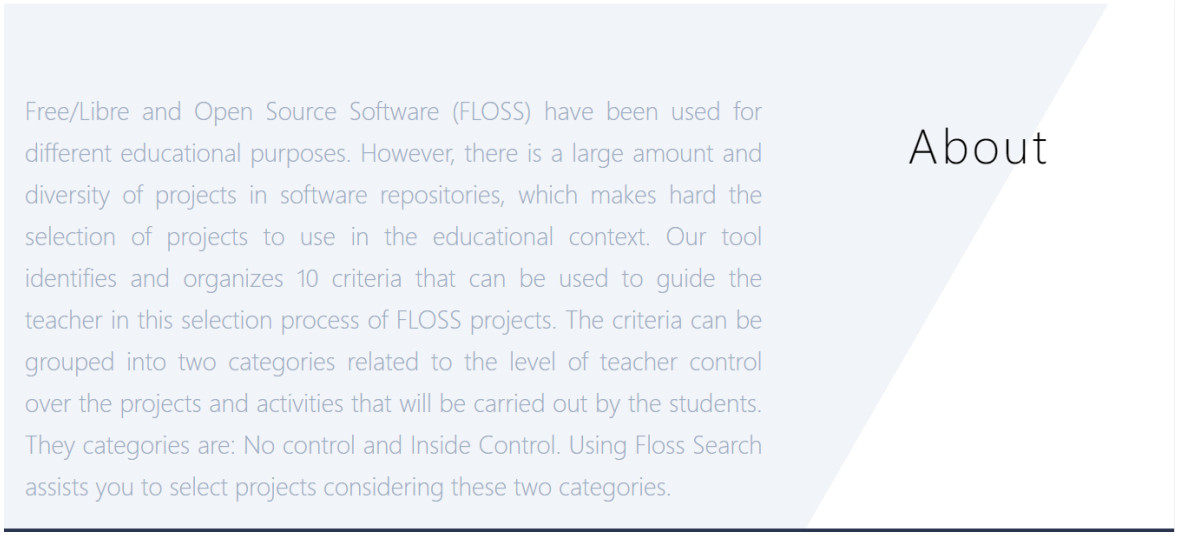

Figura 5.2 Informações sobre a ferramenta disponível na aba *About*

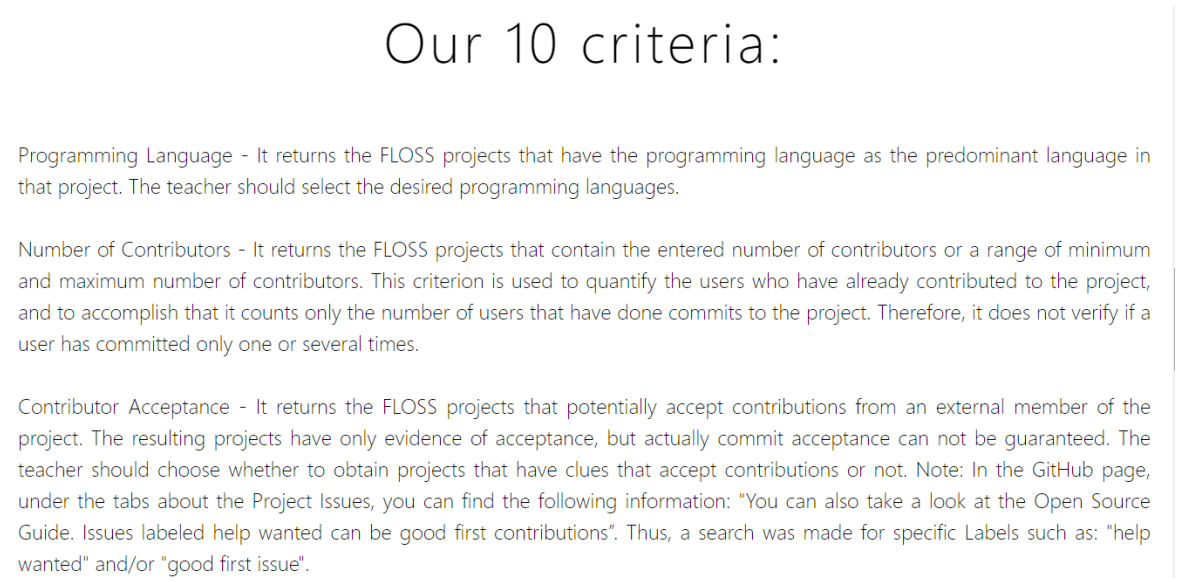

Figura 5.3 Informações sobre os critérios de seleção disponível na aba *About*

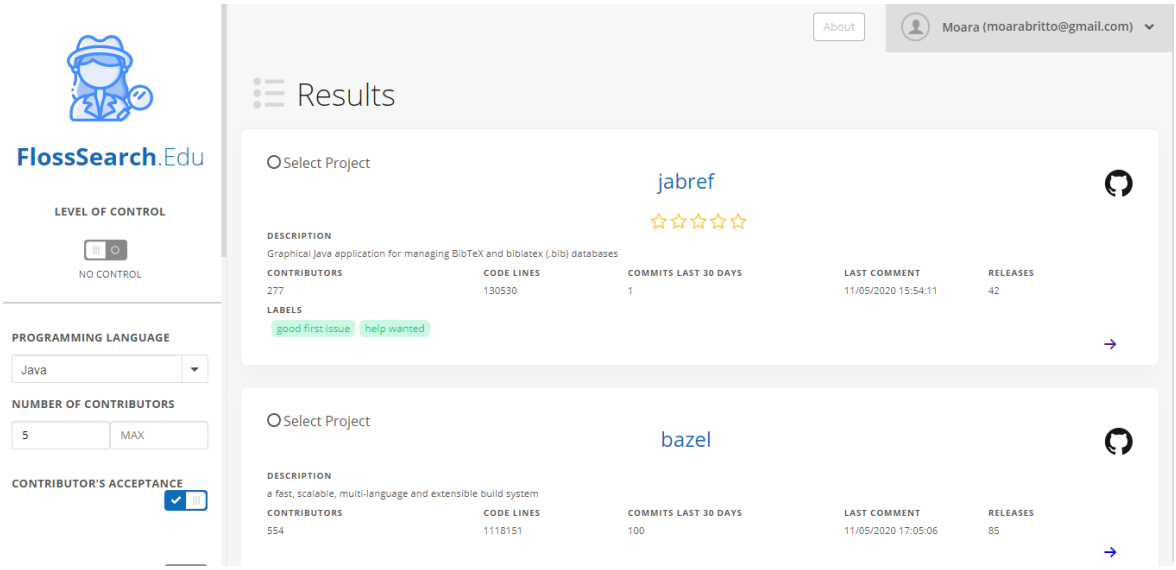

Figura 5.4 Interface de Seleção de Projetos

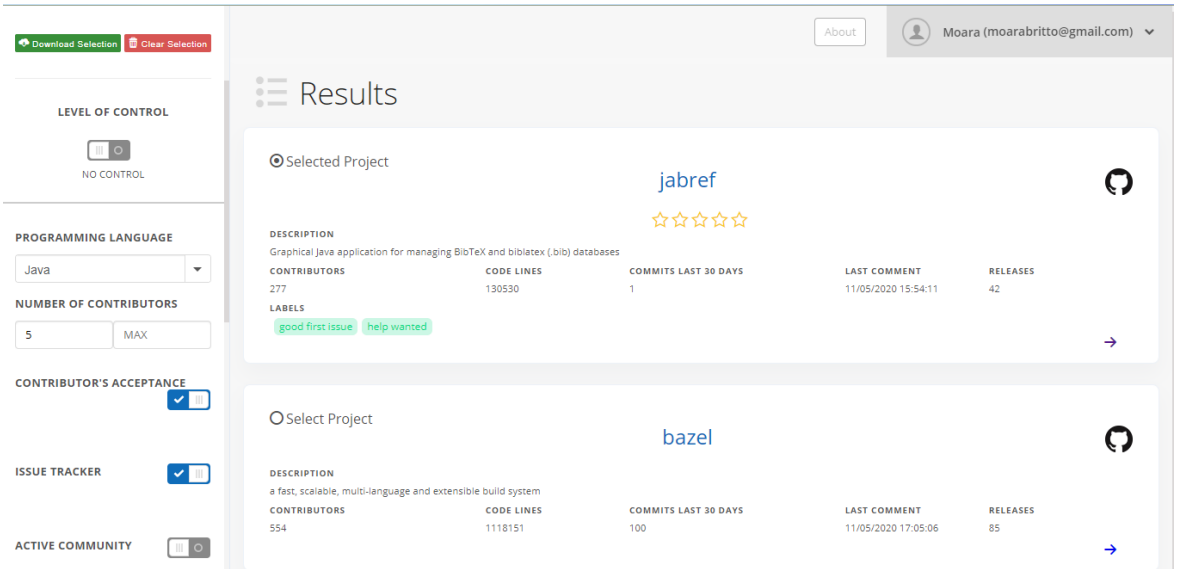

Figura 5.5 Interface para Baixar Projetos

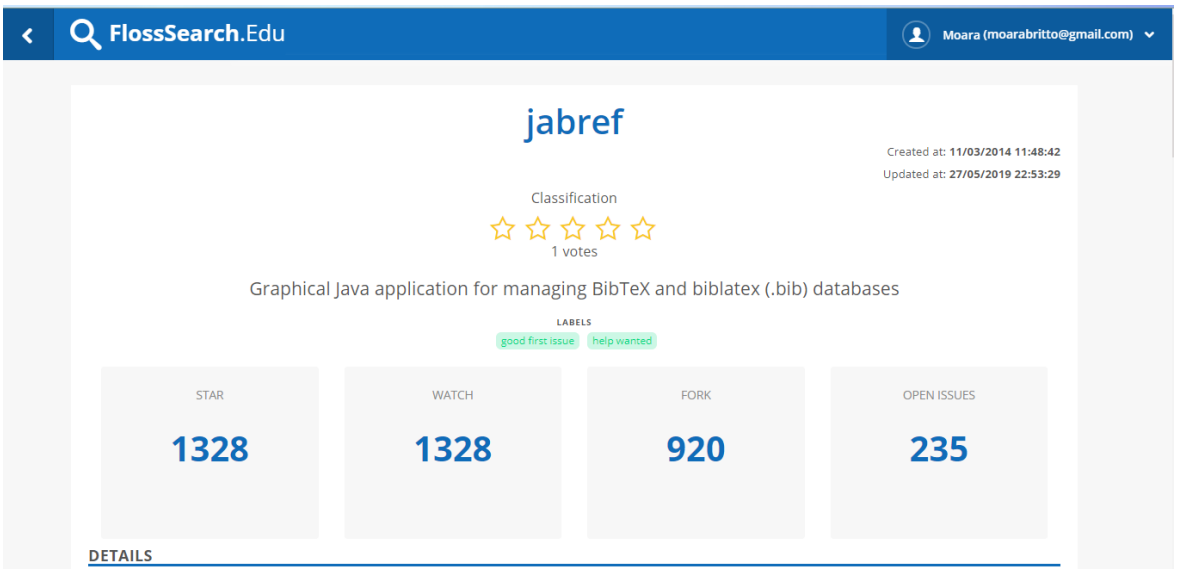

Figura 5.6 Detalhes sobre um projeto FLOSS

Figura 5.7 Detalhes sobre um projeto FLOSS (continuação)

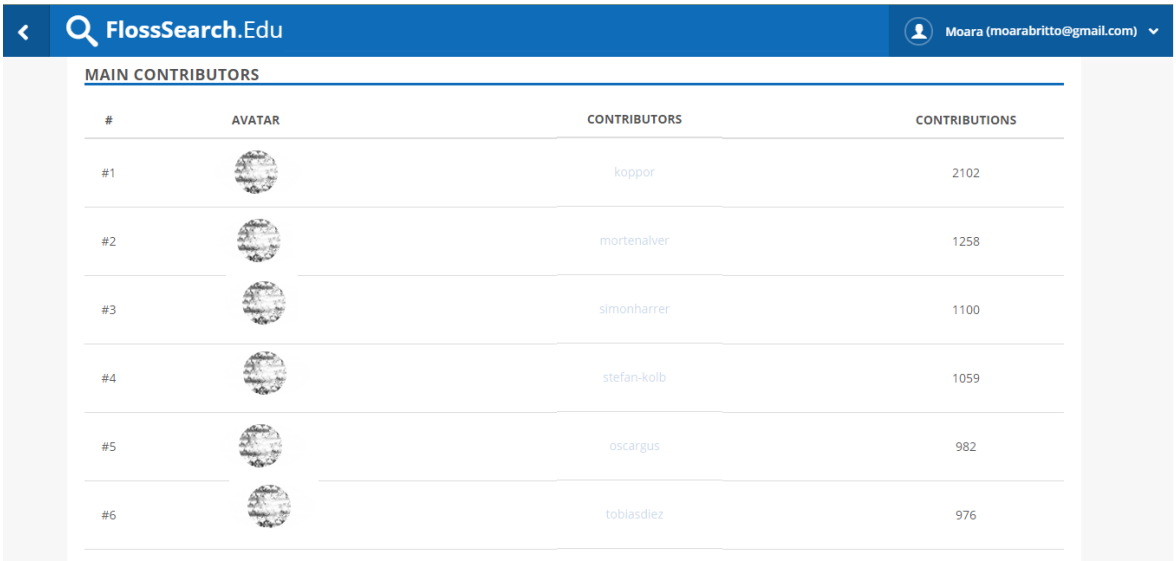

Figura 5.8 Os 10 principais contribuidores do projeto (anonimizado)

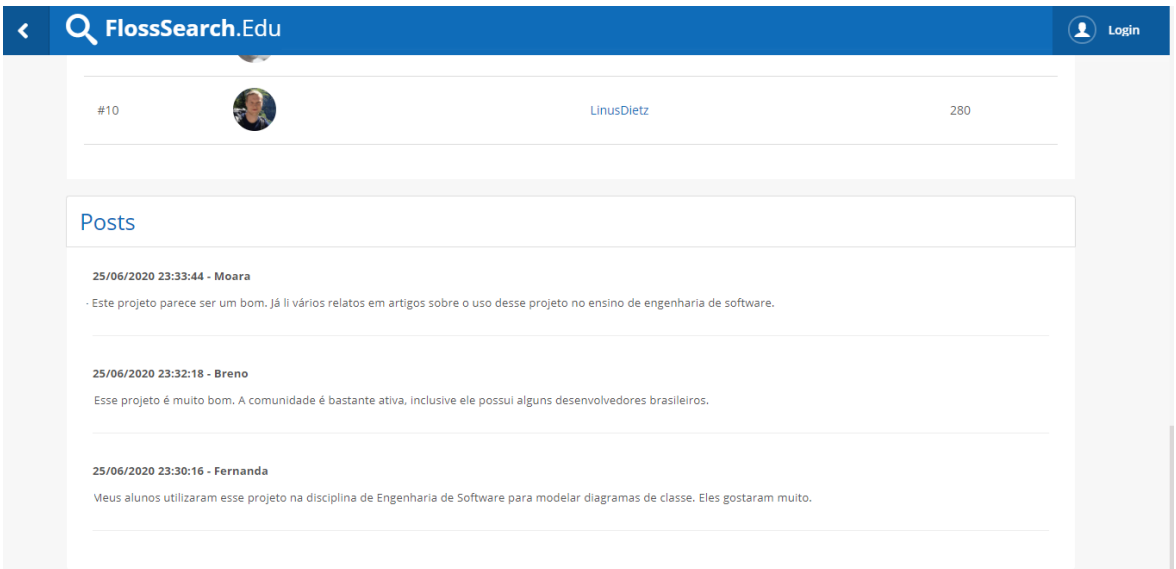

Figura 5.9 Comentários sobre um projeto FLOSS)

Capítulo

6

AVALIAÇÃO DA EXPERIÊNCIA DE UTILIZAÇÃO DA ABORDAGEM DESENVOLVIDA

A ferramenta FlossSearch.Edu implementa a abordagem de seleção de projeto FLOSS para uso no contexto da Educação em Engenharia de Software proposta neste trabalho. Assim, a avaliação da ferramenta FlossSearch.Edu pelos seus potenciais usuários é fundamental para responder às nossas questões de pesquisa.

6.1 EXPERIÊNCIA DE USO NA PERSPECTIVA DOS ALUNOS

6.1.1 Participantes

Participaram deste estudo alunos de graduação matriculados na disciplina de Engenharia de Software dos cursos de Ciência da Computação e Sistemas de Informação, em sua maioria concentrados nos 5º e 6º períodos do curso. Responderam ao questionário 24 alunos dos 30 que se matricularam na turma. Tais alunos foram agrupados em 5 equipes e convidados a selecionar projetos FLOSS para uso em atividades práticas de modelagem orientada a objetos relacionadas ao conteúdo diagramas de classes, sequência e atividades da UML. A maioria dos participantes não estavam familiarizados com os conceitos relacionados a projetos de software de código aberto, apenas lia notícias relacionadas a projetos FLOSS.

6.1.2 Instrumentação

De acordo com o planejamento, cada grupo de trabalho poderia selecionar um projeto FLOSS adequado aos seus interesses. Contudo, o uso de projetos FLOSS para fins educacionais requer a seleção de projetos uniformes em termos de tamanho e complexidade (SMITH et al., 2014). Sendo assim, o professor deveria adotar alguns critérios técnicos para o processo de seleção, por exemplo: linguagem de programação, número de lançamentos e tamanho do projeto. Portanto, nesse estudo o professor definiu as seguintes restrições: (i) o projeto FLOSS deve ser implementado na linguagem Java; (ii) o

projeto FLOSS deve ter mais de 10 *releases*; e (iii) o projeto FLOSS deve possuir mais de 5.000 linhas de código.

Para realizar a comparação entre a busca manual de projetos hospedados no GitHub com a busca realizada utilizando o Flosssearch.edu, solicitou-se que os alunos realizassem uma busca diretamente no repositório e outra busca utilizando a ferramenta FlossSearch.Edu. Os alunos tiveram acesso a um cenário para guiá-los na seleção do projeto FLOSS com características relacionadas aos critérios definidos pelo professor.

Após a experiência de busca de projetos FLOSS, foi aplicado o questionário com os alunos. Em seguida, foi verificada a confiabilidade do questionário aplicado. Analisou-se a confiabilidade para garantir a validade interna e consistência dos questionários utilizados para cada fator. De acordo com Streiner (2003) o valor mínimo aceitável para Alfa de Cronbach é 0,70. Abaixo desse valor, a consistência interna da escala utilizada é considerada baixa. Nesse estudo, os valores de Alfa foram 0,9223 para o item Utilidade Percebida e 0,8028 para Facilidade de Uso. Assim, os resultados demonstram que o questionário aplicado foi um instrumento confiável.

6.1.3 Impressões sobre o FlossSearch.Edu por meio do TAM

6.1.3.1 Utilidade percebida da ferramenta A Figura 6.1 apresenta as respostas dos participantes em relação à utilidade percebida do FlossSearch.Edu. É possível observar o número de participantes que concordaram com a utilidade da ferramenta.

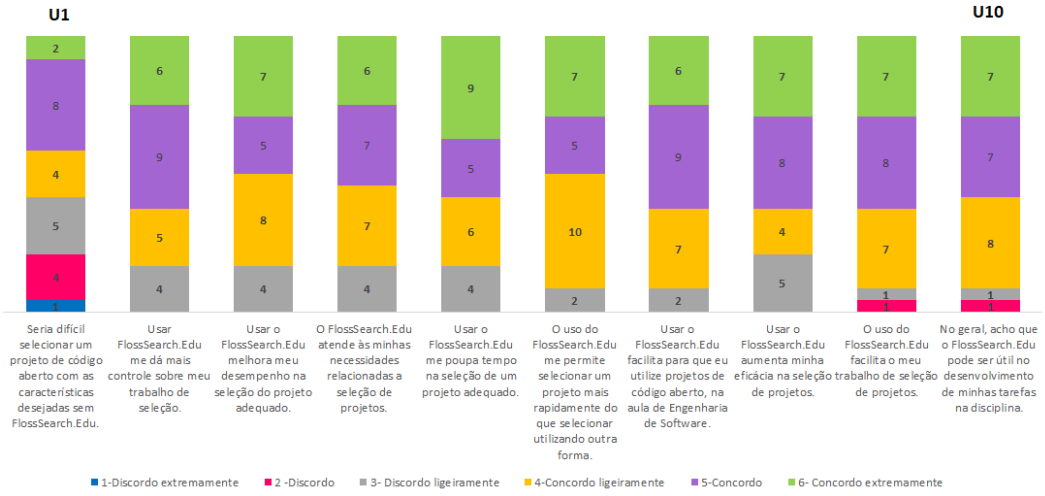

Figura 6.1 Utilidade percebida: um resumo das respostas.

O item `U1` teve o maior número de discordâncias, totalizando 10; no entanto, mais de 50% dos participantes (14) concordaram com a afirmação de que "seria difícil selecionar um projeto de código aberto com as características desejadas sem FlossSearch.Edu". O número de participantes que não concordaram com `U6`, `U7`, `U9` e `U10` foi no máximo dois. 37,5% concordam extremamente que o uso de FlossSearch.Edu economiza tempo

na seleção do projeto (U5). Além disso, apenas 16,6% dos participantes discordaram levemente de `U2`, `U3`, `U4` e `U5`.

Conforme mencionado na seção 3.3.1, foi coletado também a opinião dos alunos por meio questões abertas. Optou-se por utilizar esse método para possibilitar que os sujeitos expressassem sua opinião com mais liberdade. Nesse contexto, em relação a utilidade percebida do FlossSearch.Edu, os alunos foram questionados se conseguiram encontrar projetos que atendiam as restrições apontadas pelo professor. Além disso, foi pedido que os alunos dessem uma opinião, de maneira geral, sobre o uso da ferramenta.

As respostas que os alunos deram ao questionário TAM e as respostas dadas às questões abertas foram positivas. Por exemplo, 6 alunos enfatizaram que a ferramenta era boa. Outros alunos indicaram que a FlossSearch.Edu facilitou a pesquisa dos projetos, realizando de forma rápida, e que era bastante útil.

6.1.3.2 Facilidade de uso percebida da ferramenta A Figura 6.2 apresenta as respostas dos alunos sobre a facilidade de uso da ferramenta FlossSearch.Edu.

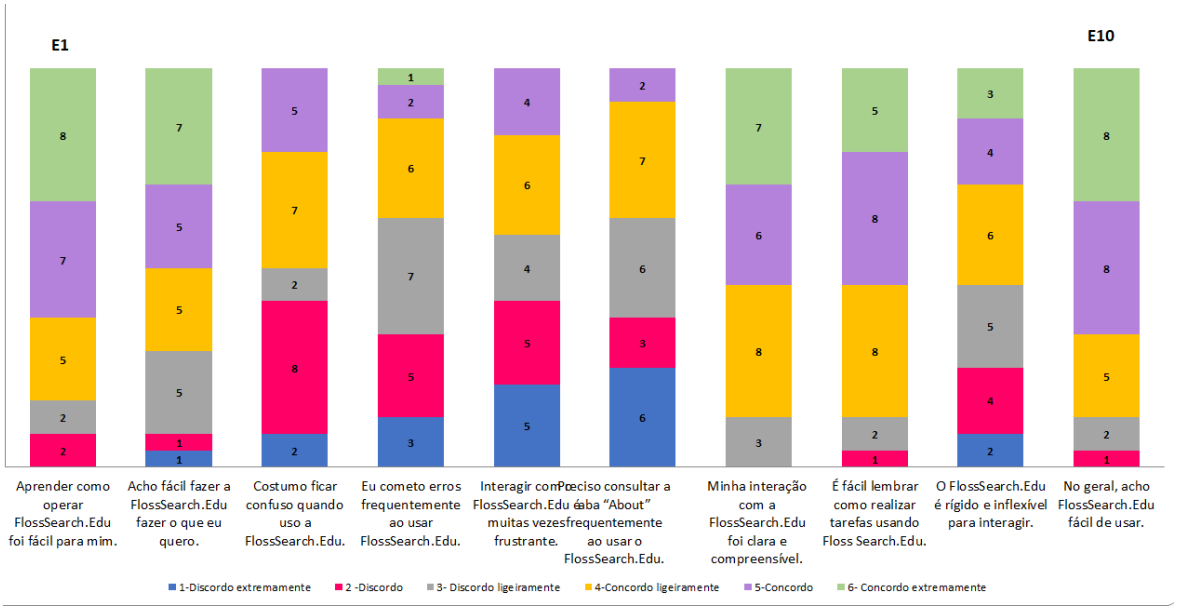

Figura 6.2 Facilidade de uso: resumo das respostas.

A Figura 6.2 mostra que 87,5% dos participantes acharam que a interação com FlossSearch.Edu foi clara e compreensível (E7), que é fácil lembrar como realizar as tarefas (E8), e que, é fácil usar a ferramenta (E10).

Sobre a opinião dos alunos relacionado a facilidade de uso da ferramenta, o feedback também foi positivo. Eles indicaram principalmente que o FlossSearch.Edu era prático e fácil de usar. Ao comparar a ferramenta com o GitHub, os alunos declararam que o FlossSearch.Edu é mais intuitivo e apresenta buscas claras e objetivas.

6.1.3.3 Uso da Ferramenta no Futuro A Figura 6.3 apresenta o resultado sobre a indicação de uso futuro do FlossSearch.edu pelos alunos.

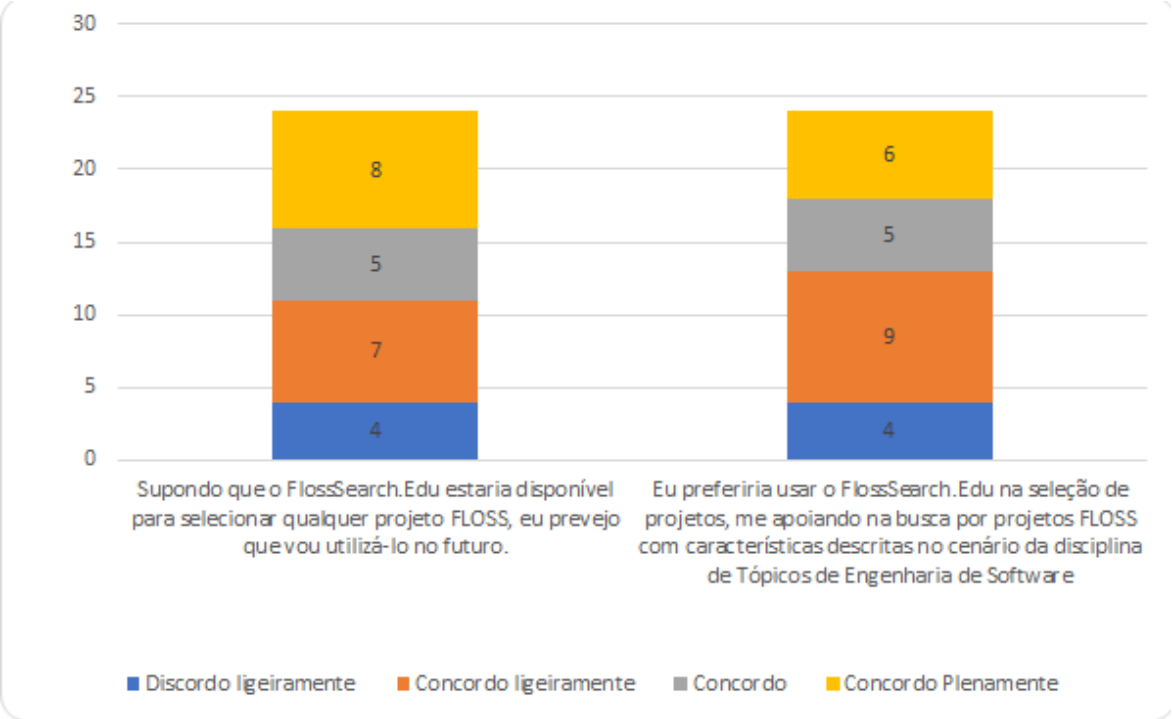

Figura 6.3 Uso futuro previsto: Resumo das respostas.

Em relação ao uso futuro esperado (Figura 6.3), 83,3% (20 participantes) concordaram que se FlossSearch.Edu estivesse disponível para selecionar qualquer projeto FLOSS, eles o usariam no futuro (S1). Apenas 4 participantes (16,6%) discordaram ligeiramente sobre a preferência de usar `FlossSearch.Edu` para selecionar projetos FLOSS com base nas características especificadas (S2).

6.2 EXPERIÊNCIA DE USO NA PERSPECTIVA DOS PROFESSORES

6.2.1 Participantes

A avaliação da abordagem de seleção de projetos FLOSS no contexto de EES foi realizada com professores que lecionam em diversas instituições brasileiras de ensino superior. Os professores foram convidados a participar considerando a sua experiência no ensino da disciplina de Engenharia de Software em cursos de graduação, e uso de projetos FLOSS como objeto de estudo em suas aulas. No total, 10 professores participaram desse processo de avaliação composto pela aplicação do modelo TAM e protocolo de TA.

6.2.2 Instrumentação

Foram realizadas sessões virtuais e individuais, em que foi explicado aos participantes que a avaliação incluiria uma sessão de videoconferência e seria utilizado o método TA.

Durante a avaliação, cada participante deveria compartilhar sua tela, utilizar o cenário apresentado para familiarização da ferramenta, e a medida que fosse navegando pelo FlossSearch.Edu iria "pensando em voz alta", apresentando suas impressões positivas e negativas. Por exemplo: "Gostei desse critério; Senti falta de realizar a pesquisa utilizando o critério y; Fiquei confusa quando tentei selecionar projetos em tal linguagem".

Antes de iniciar a sessão de videoconferência, os professores receberam as orientações e os seguintes cenários por e-mail:

"Os textos abaixo apresentam cenários opcionais para nortear sua familiarização (primeiro contato) com a ferramenta, selecionar alguns projetos FLOSS e para que possa entender melhor o que gostaríamos de avaliar. Os cenários descrevem situações na qual professores de Engenharia de Software precisam selecionar um ou mais projetos FLOSS para uso pedagógico, com base em algumas características destacadas em negrito. Sendo assim, gostaria que o senhor explorasse a ferramenta FlossSearch.Edu para buscar um ou vários projetos FLOSS que possivelmente atendesse a um mais cenários abaixo e/ou que o senhor pudesse utilizar em suas aulas de ES. A ferramenta FlossSearch.Edu está disponível em: http://191.252.92.63/flosssearch/."

Cenário 1. No contexto de uma disciplina ES oferecida no 5º semestre para alunos com conhecimento básico em Java e C, a Prof. Roberta deseja proporcionar aos seus alunos experiências do mundo real, relacionadas ao desenvolvimento e evolução de software de um projeto de software não trivial, hospedado no Github. O objetivo é que, ao longo da disciplina, os alunos trabalhem em diferentes alterações no código-fonte do projeto e outros artefatos, e enviem suas contribuições para o projeto, para que este seja revisado (e quiçá aceito) pela equipe/comunidade que mantém o software. Para isto, é necessário que o projeto aceite contribuição de membros externos e utilize algum mecanismo de registro e acompanhamento de tarefas (issues).

Cenário 2. Prof. João quer utilizar um projeto FLOSS em suas aulas sobre engenharia reversa para criação de diagramas de classe UML detalhados para seus alunos de 3º semestre. Ele deseja utilizar o código do projeto nas aulas, fazendo uma cópia (fork) para um repositório local, para uso apenas na disciplina, sem preocupação com a evolução do projeto real escolhido. Os alunos da turma possuem pouca experiência em programação, mas tiveram contato prévio com a linguagem de programação Java na disciplina do semestre anterior, sendo capazes de compreender programas pequenos, com até 50000 linhas de código e poucas classes. Um professor de outra universidade que utilizou FLOSS em suas aulas de engenharia reversa e refatoração, relatou ao Prof. João que o domínio do projeto utilizado pode influenciar a facilidade de compreensão e uso do projeto FLOSS e na aprendizagem dos alunos.

Cenário 3. Prof. Luis quer utilizar um projeto FLOSS em JAVA em suas aulas. Ele deseja proporcionar aos seus alunos experiências do mundo real, relacionadas ao desenvolvimento e evolução de software de um projeto de software. Prof. Luis não contribui com projetos FLOSS hospedados no github ou outra plataforma deste tipo, e fica receoso de não conseguir sanar dúvidas específicas de seus alunos sobre como contribuir com o projeto. Ela se recorda de um artigo que leu, que relatava que contribuidores podem ajudar na compreensão de informações sobre o projeto, sua cultura e normas. Assim, Prof. Luis decide buscar um projeto com uma comunidade de usuários ativos,

com base na lista de comentários do projeto, para dar suporte aos alunos, caso necessário.

Cenário 4. No contexto da disciplina de ES, Prof. Maria quer utilizar como objeto de estudo, projeto FLOSS em C, com seus alunos de 2º semestre . Prof. Maria acredita que projetos FLOSS com até 20 lançamentos (releases) e que possuem até 30.000 linhas de código serão suficientemente bons para trabalhar com os alunos. Ela também deseja guardar alguns dos projetos encontrados para analisar as suas características com mais calma em outro momento.

Para contextualizar o professor no início da videoconferência, foi apresentado o contexto geral dessa pesquisa e explicado que o objetivo do estudo seria conhecer a sua opinião sobre a seleção de projetos FLOSS para uso em contexto pedagógico de disciplinas de ES, por meio de uma seleção sistemática, estruturada, com base em critérios técnicos e sociais, pré-definidos no ambiente da ferramenta FlossSearch.Edu. Em seguida, o professor foi convidado a explorar a ferramenta e realizar a etapa de seleção de projetos FLOSS usando a FlossSearch.Edu, de acordo os cenários de uso estabelecidos.

Durante toda a interação do professor com a ferramenta, foi realizada a gravação de tela e em seguida, foi disponibilizado o link para o preenchimento do questionário baseado no TAM, apenas com questões fechadas. A duração de cada avaliação teve uma média de 30 minutos.

6.2.3 Impressões sobre o FlossSearch.Edu por meio do TAM

6.2.3.1 Utilidade percebida da ferramenta A Figura 6.4 mostra as respostas dos professores sobre a utilidade percebida da ferramenta.

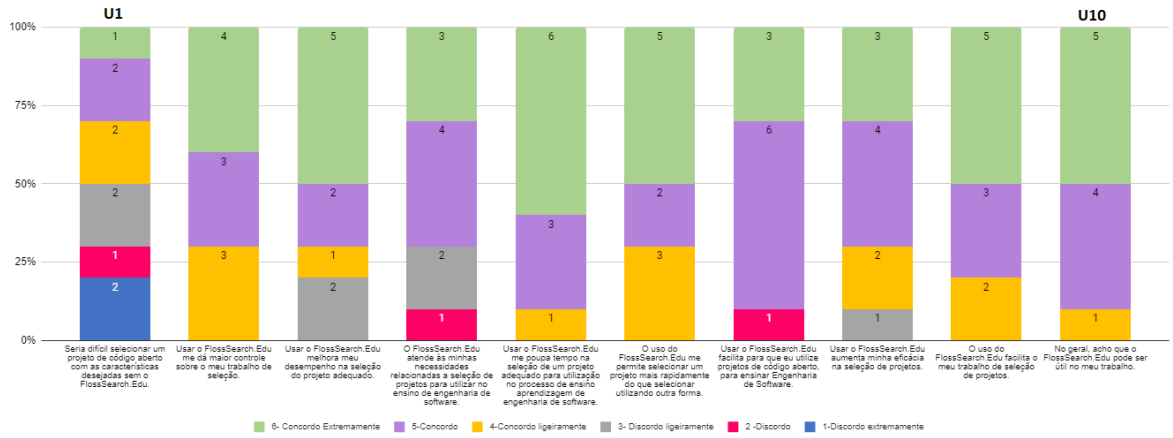

Figura 6.4 Utilidade percebida: um resumo das respostas.

O item U1 teve o maior número de discordância. 50% dos professores afirmaram que conseguiriam selecionar projetos com as características desejadas sem utilizar o FlossSearch.Edu, e 2 professores discordaram extremamente dessa afirmação. Não houve discordância com U2, U5, U6, U9 e U10. 60% concordaram extremamente que utilizar FlossSearch.Edu poupa tempo na seleção de um projeto adequado para utilização no processo de ensino e aprendizagem de ES (U5). Além disso, apenas 1 professor discordou de

U4, U7 e U8, e 2 professores discordaram levemente da afirmação sobre o FlossSearch.Edu melhorar o desempenho na seleção do projeto adequado (U3).

6.2.3.2 Facilidade de uso percebida da ferramenta A Figura 6.5 mostra que nenhum professor discordou que o FlossSearch.Edu seja fácil de usar (E10) e fácil de lembrar como executar tarefas (E8). Apenas 1 professor discordou ligeiramente sobre a facilidade em aprender a operar o FlossSearch.Edu (E1). 80% dos participantes acharam que a interação com a ferramenta foi clara e compreensível (E7). 40% afirmaram que muitas vezes se confundiram ao utilizar o FlossSearch.Edu (E3) e precisaram consultar a aba "About" (E6), e 3 professores relataram ter cometido erros frequentemente (E4).

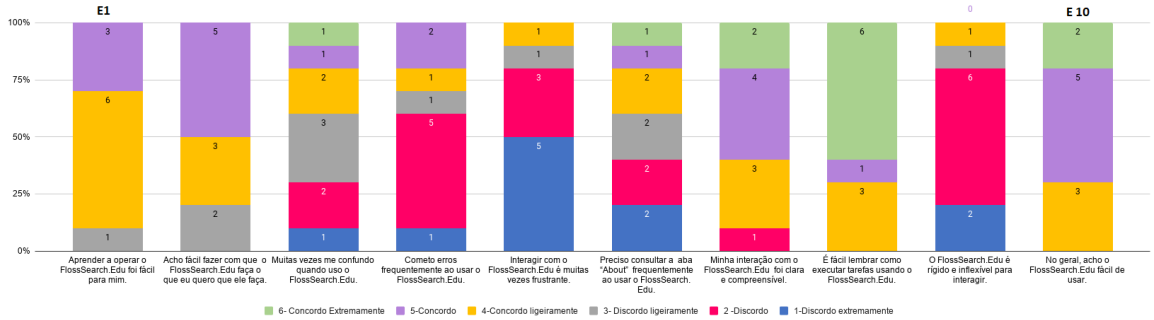

Figura 6.5 Facilidade de uso: resumo das respostas.

6.2.3.3 Uso da ferramenta no futuro Em relação a um possível uso futuro da ferramenta, como mostrado na Figura 6.6, todos os professores que participaram concordaram que se o FlossSearch.Edu estivesse disponível para selecionar qualquer projeto FLOSS, eles utilizariam no futuro (S1). Todos os professores indicaram que prefeririam usar o FlossSearch.Edu na seleção de projetos, apoiando-os na busca por projetos FLOSS com características desejadas para o uso no processo de ensino aprendizagem de Engenharia de Software (S2).

6.2.4 Impressões sobre o FlossSearch.Edu por meio do Think Aloud

A percepção dos professores sobre a ferramenta FlossSearch.Edu, coletada por meio do Think Aloud, foi registrada em arquivos de áudios extraídos das gravações das videoconferências realizadas individualmente. Em seguida, foram realizadas as transcrições para arquivos de texto e a análise desse conteúdo, a fim de agrupar as frases e comportamentos em um núcleo comum de sentido. Com a junção dessas informações em grupos maiores que partilham conteúdo foi possível agrupar as sentenças de acordo com sua relevância para avaliação dos critérios de seleção utilizados, da usabilidade da ferramenta, sugestões de melhorias e uso futuro pretendido do FlossSearch.Edu. Relatos dos professores que não diziam respeito a estes aspectos foram descartados.

No início da videoconferência, apenas dois participantes demonstraram tentar entender o funcionamento geral da ferramenta antes de realizar algum procedimento referente

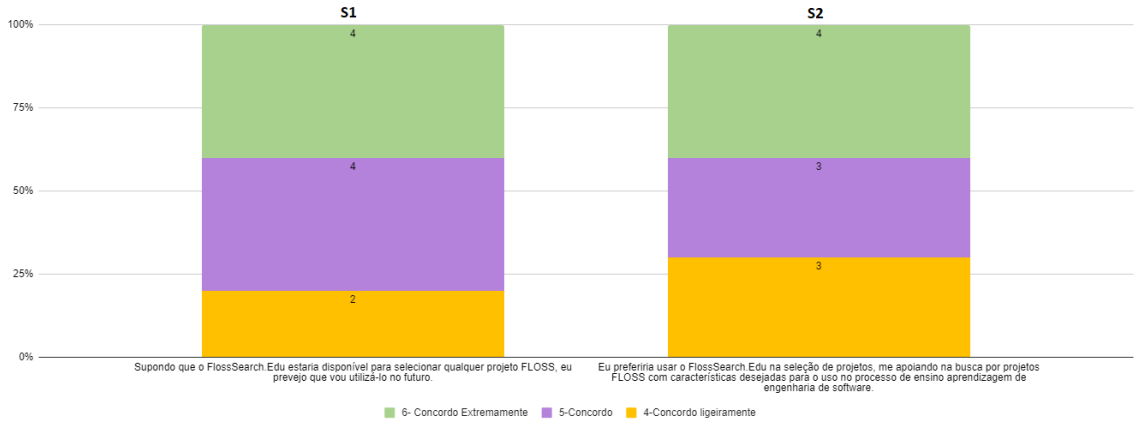

Figura 6.6 Uso futuro previsto: resumo das respostas.

aos cenários disponibilizados. Esse comportamento pode ser observado a partir das primeiras falas relatadas a seguir:

"Eu vou dar uma olhada rápida nessas informações que tem aqui no início. Aqui provavelmente já começa, mas eu vou ler as informações abaixo antes de começar".

"A primeira coisa que eu estou fazendo aqui é dando uma movida de olho na ferramenta... Eu tento olhar tudo, mas eu não vou muito a fundo. Eu quero ver tudo que eu consigo obter de informação sobre a ferramenta. Então eu estou rolando para cima e para baixo para eu entender o que a ferramenta faz e como é que ela faz. Eu entendi que ela busca algumas coisas baseado nas features que eu quero integrar ao meu curso e que ela pesquisa em projetos do GitHub... Como eu não tenho medo de mexer, de fuçar, eu prontamente cliquei aqui.. vim aqui e cliquei e descobri que algumas coisas desaparecem aqui quando eu clico nisso".

Foi possível perceber que somente um participante iniciou a avaliação da ferramenta realizando a leitura do *about* com foco em conhecer os critérios e o nível de controle para selecionar os projetos FLOSS. Os demais professores utilizaram a estratégia de partir inicialmente para a leitura da tela inicial da ferramenta e dos cenários disponibilizados, mesmo sem a visão sistêmica da abordagem desenvolvida.

6.2.4.1 Utilidade dos Critérios e Funcionalidades da Ferramenta Os professores avaliaram a utilidade dos critérios e funcionalidades da FlossSearch.Edu.

Maturidade

Na avaliação houveram relatos indicando que a utilização do critério de maturidade é interessante, bem como a relação existente entre a maturidade e a quantidade de *releases* de um projeto.

O professor que achou interessante ter utilizado o critério maturidade relatou o seguinte:

"Mas eu achei interessante, e acho que é válido não somente para Engenharia de Software, como também para outras disciplinas."

Quatro professores conseguiram identificar a relação de maturidade com *release*, sendo que um deles ainda apresentou sugestão de melhorias em sua operacionalização e outro apresentou objeção quanto à essa relação, conforme descrito a seguir:

"Aqui eu consigo pegar pela maturidade do projeto, por meio das releases. Quanto mais releases, provavelmente mais commits, mas se eu quiser pegar projetos que tem mais de x commits eu não consigo. Poderia fazer da mesma forma que você fez para contribuidores, fazer para commit."

"Nível de maturidade e release, estou entendendo aqui como se fosse uma aplicação com menos problemas para resolver, uma coisa mais estável."

"Se eu escolher com pelo menos 20 releases deve ser aqui em maturidade, a quantidade de release."

"Em maturidade parece ser quantidade de releases, mas eu não vejo relação de maturidade com release, elo menos não cabe na minha cabeça."

Linguagem de Programação

Durante a avaliação, dois professores relataram estratégias para escolha do critério de linguagem de programação na seleção de projetos, conforme descrito a seguir:

"Como meus alunos já estão acostumados a trabalhar com Java, então vou escolher a linguagem java."

"Escolha da linguagem, geralmente a gente seleciona, pelo menos na disciplina de Engenharia de Software I, a gente seleciona Java."

Apenas um professor apontou que em sua experiência de seleção de projetos, não costumava levar em consideração a linguagem de programação:

"Basicamente eu não restringia uma linguagem de programação, deixava em aberto, porque as vezes tem aluno que se interessa a trabalhar para um projeto com uma linguagem que ele nunca trabalhou."

Issue Tracker

Dois professores apresentaram em seus relatos a importância de utilizar o critério de Issue Tracker.

"Na minha turma o que a gente faz é procurar por issues na Issue Tracker para que os alunos possam contribuir com isso aí".

"Issue Tracker eu acho interessante e uso muito isso".

A abordagem desenvolvida considera apenas as issues abertas dos projetos selecionados, sendo que um professor apresentou interesse em acessar as issues fechadas do projeto durante o uso da ferramenta.

"Eu também não consigo acessar as issues do projeto. Eu só consigo ver as que estão abertas".

Aceitação de Contribuidor

Dentre os professores que participaram do estudo, quatro deles compreenderam o critério de aceitação de contribuidor, sua importância, bem como as *labels* que são verificadas, já que conseguiram identificar os projetos que estão pedindo auxílio.

"Acho que esse Contributors Acceptance significa que aceita contribuição externa".

"Sobre quiçá aceito, eu marcaria aceitação de contribuidor com certeza, poque é importante para minha disciplina que seja aceito."

"Eu acho legal ter essas label aqui (good firs, help wanted), pois nelas possuem issues que ajudam os novatos. Então considerando que os alunos são provavelmente novatos eu iria dizer isso para eles, para eles procurarem projetos que tenham essas issues e que tentem começar por essas issues".

"Esse campo de contribuidores foi um dos que mais usei, linguagens e contribuidores".

"Ao clicar no rótulo "good first issue", um professor chamou atenção para uma informação que até então desconhecíamos."

" Aí uma dica aqui pra você se eu não estou enganado, a tag "help wanted" ela é padrão e talvez você ache em todos os projetos. Não necessariamente eles usam. Ela vem como padrão quando você cria um projeto."

Comunidade Ativa

Dentre os avaliadores, apenas um professor fez uma observação sobre a importância de utilizar o critério de comunidade ativa:

" Para que seja aceito é implícito que tenha que ter uma comunidade ativa, pois se o projeto está morto não adianta nada."

Tamanho do Projeto

Alguns professores ficaram confusos em relação ao tamanho do projeto. Para eles não ficou claro que a métrica utilizada para indicar o tamanho foi a quantidade de linhas de código.

" Em Project Size poderia ficar claro que é linhas de código, pois pode ser número de classe por exemplo."

Outros professores disseram não se importar com linha de código.

" Eu não me importaria muito com tamanho do projeto."

" Linha de código é uma métrica muito incipiente, só ela só não ajuda a você determinar a complexidade."

Domínio

um professor achou interessante o critério de Domínio, enquanto que outro professor ficou confuso sobre sua utilização:

" Esse domain achei interessante."

" Ficou confuso em relação ao campo Dominio. Achou que poderia ter relação com o campo de Labels."

Projeto Ativo

Em relação a estratégia utilizada de verificar os *commits* dos últimos 30 dias, um professor comentou:

" Eu nao sei se commits, 30 dias. Vocês se basearam em alguma literatura para fazer isso? Porque eu já vi métricas que falam de até 6 meses de commits para considerar um projeto ativo. Claro que isso pode variar de pesquisa, de projeto, enfim."

Principais Contribuidores

Três professores gostaram da decisão de exibir os principais contribuidores. Um professor ainda relatou que no processo de seleção de projetos para utilizar em sala de aula, ele faz contato prévio com os principais contribuidores brasileiros.

" Legal aqui que tem uma lista de contribuidores."

" Aqui tem os principais contribuidores do projeto. Clicando aqui vai pro portfólio de cada um. Legal."

" Para fazer o contato prévio eu geralmente procuro por projetos que tenha brasileiros."

Um professor chamou atenção para uma limitação do critério.

" O problema dessa lista de principais contribuidores é que ela mostra todos os contribuidores desde o início do projeto. Não significa necessariamente que esses caras estão ativos né. Seria legal de alguma forma, saber nesse momento que são os revisores do projeto, que estão revisando os pull request e que estão mais ativos. Ordenar por exemplo os contribuidores mais ativos nos últimos 30 dias. Como tem professores que não são muito experientes, então vai precisar da ajuda dos principais contribuidores para ajudar a guiar os alunos."

Informações dos Projetos

Por meio dos relatos dos professores sobre as informações disponíveis dos projetos, alguns demonstraram interesse, sugestões de melhorias e novas descobertas a partir do exposto:

" Nossa aqui tem label para novato Help wanted."

" Clicando aqui eu posso olhar as informações desse projeto. Esse projeto foi utilizado com sucesso em diversas turmas (ele leu um comentário feito) opa! Isso eu não tinha visto. isso aqui foi interessante. Achei bem legal."

" Tem um post aqui embaixo legal."

" Outra coisa que notei e não falei foram as estrelinhas. Eu notei que esses projetos tem bastante estrela. Talvez ele seja um projeto legal de trabalhar porque um dia alguém já deve ter votado. Se a gente entra, observa que ele teve um voto."

" Legal. Isso é um indicativo de alguém usou e aprovou o uso desse projeto, interessante."

" (leu as informações relacionadas ao projeto) (chamou atenção a informação de possuir wiki). (Conseguiu identificar as informações sociais e técnicas)

" Aqui na página com detalhes resumidos do projeto não mostra a linguagem."

" As informações que traz dos projetos é legal. Mostrar a licença do projeto é importante. A minha maior dificuldade em encontrar projetos é que as informações mais interessantes não vêm. Por exemplo linhas de código."

" O que eu senti falta foi de uma descrição mais detalhada dos projetos."

" Essa parte da licença é importante... Acho que a licença deveria vir na página principal. A licença não é muito fácil de identificar. Não sei como está pegando essa licença aqui, é sempre complicado pegar licença."

" Ele também dá como resposta a descrição do projeto, número de contribuidores, número de linhas de código, a quantidade de commits nos últimos 30 dias, o último comentário e o número de releases."

" Quando eu entro em algum projeto ele vai mostrar o que? Tem esse label Help Wanted em um dos projetos... Ele traz a informação da quantidade de forks, quantidade de issues abertas."

" Aqui tem o nome completo, a página, linguagem, tipo de licença, total de contribuidores, número de linhas de código, número de releases, comentários, principais contribuidores, posts."

" Legal ter algumas informações resumidas aqui."

" Acho importante ter essa licença do projeto aqui."

" Seria útil aqui também nos resultados ter um esquema de ordenação. Ordenar por numero de contribuição, número de código, release, ultimo comentário. Senão fica difícil para o usuário navegar aqui."

Download das informações dos projetos

A funcionalidade que permite o download das informações dos projetos foi bastante elogiada pela sua utilidade:

" E na hora que eu seleciono os projetos achei legal. É possível baixar. Legal! Ele me dar o detalhe básico, o resumo de cada projeto".

" Projeto selecionado. Posso fazer o download. Ok."

" Agora eu posso fazer o download desses projetos. Ele vai me da uma listinha com algumas informações para olhar depois. Nessa listinha tem o link para os repositórios no GitHub. Ok."

" Eu não tinha visto essa opção de Select Project. Vou selecionar, mas depois que selecionar o que eu faço com eles? Ah sim, Download selection. Está aqui em cima."

" Aqui eu já posso baixar direto? Baixo uma planilha."

" O que é o donwload? Ah ele vai me dar uma planilha...ah ta!"

" Para guardar os projetos, o que eu precisaria fazer? Será que é necessário ter login? Ah achei. Ah legal.. esse tipo de coisa que baixa aqui é bem legal."

Para aprimorar a funcionalidade de download das informações dos projetos, um professor apresentou a seguinte sugestão de melhoria:

" Outra opção seria eu conseguir baixar todos sem precisar eu clicar em todos um a um. Aqui por exemplo retornou 86 projetos. Se eu tiver q clicar em todos os 86 eu vou desistir."

6.2.4.2 Usabilidade da Ferramenta A avaliação dos participantes foi agrupada em 7 dos 10 princípios gerais para o design da interface do usuário (NIELSEN, 2005). São eles: i) intuitividade da interface gráfica; ii) consistência e padrões; iii) visibilidade do status do sistema; iv) reconhecimento em vez de recordar; v) ajuda e documentação; vi) controle e liberdade do usuário; e vii) localização dos critérios.

Intuitividade da Interface Gráfica

A medida que os professores foram realizando a seleção baseada nos cenários disponibilizados, fizeram comentários e sugestões de melhorias sobre a intuitividade da FlossSeach.Edu.

" No geral os filtros estão legais. A questão de usabilidade para mim está tranquilo."

" Talvez um ícone de informação aqui facilite a minha vida."

" Talvez se você colocasse alguma dica mais explicita. Pode colocar algo do tipo: quer saber mais sobre os paramentos e linkar para uma página que tem isso aí talvez seja mais fácil das pessoas clicarem."

" Uma coisa que podia ter aqui era um link pro projeto. Os outros eu consegui acessar porque tinha as labels que eles usam. Mas esse aqui, por lugar nenhum eu consigo chegar na página do projeto."

" Foi fácil fazer criar um usuário"

" No geral achei bem fácil de usar"

" Achei bom que aqui na página inicial deixa claro que é projetos do GitHub, por que aqui em cima fala que é Open Source e aqui é legal que da descrito da fonte."

" Uma coisa que eu gostaria de ter aqui é clicar no projeto, e ele abrir direto no GitHub para eu dar uma olhada (Na pagina de detalhes de informações). Seria legar tem um botão: Abrir no Git."

" Poderia ter também um lugar que se deixasse o mouse em cima ele mostrasse que esse critério diz, por que senão vou ter que voltar lá no início pra ver de novo."

" Poderia ter informações de propaganda: Tem mais de x projetos indexados; já tiveram mais de x acessos; já foram realizadas mais de x buscas; mais de x projetos

já foram selecionados. Poderia ter um indicativo que tem mais informações aqui pra baixo nessa tela inicial."

" Acho que para o cenário um está bem fácil de achar os projetos com esses critérios por que são caixas aqui no canto esquerdo e é simplesmente marcando."

" Está ótimo assim. Bem fácil de navegar, bem fácil de encontrar. Bem tranquilo."

Além disso, foi possível notar também uma série de falas pelos professores que expressaram a falta de intuitividade da interface. São elas:

" E aqui a gente tem uns links que não leva a lugar nenhum (em detalhes), porque eu clico e eles não funcionam."

" [Ao lembrar do botão About]. Eu mesmo nunca clicaria ali. Porque quando eu vejo alguma coisa de About fico pensando que é sobre o produto...Não espere que muitas pessoas cliquem no About."

" Esse botão do About não está adequado aqui."

" Agora que eu vi ali que o About tava habilitado, porque do jeito que tava o About ali eu pensei que tava desabilitado, é que eu vi habilitado."

" Eu já li critérios em vários pontos do site, mas eu não vi nenhum link direto para poder ver quais são esses critérios."

" A primeira coisa que achei estranho nessa tela inicial é que você tem esses 3 pontinhos aqui. Normalmente, quando tem isso aqui fica parecendo que tem aquele carrossel que fica passando as coisas. E esses pontinhos é pra indicar em que ponto do carrossel você está."

" Essa barra aqui acho que está um pouco confusa [barra superior da página que apresenta os resultados]. Porque deveria vir classificado, mas está classificado e não classificado junto, a busca que não deixasse selecionar os dois."

" Não sei, esses rádios aqui [Select project] poderiam passar despercebidos, não sei se está uma boa posição para ele aqui."

" Não ficou muito intuitivo eu ter que marcar isso aqui para ele fazer o download. A ideia é legal, mas eu encontrei isso aqui puramente no achometro."

Consistência e Padrões

De acordo com a consistência e padrões, ao acessar uma ferramenta os usuários não devem se perguntar se palavras, situações ou ações diferentes significam a mesma coisa (NIELSEN, 2005). Com isso, foram apresentadas situações que infringiam esse princípio, conforme as seguintes falas:

" Aqui eu não tenho mais o hover de novo (maturity, active Project). Então eu tinha acostumado que eu tinha finalmente descoberto que tinha um esquema de informação sobre cada critério."

" Eu não sei o que quer dizer maturity, release, então ter uma informação é legal."

“ No primeiro momento eu confundi essas estrelas aqui com as estrelas do Git. Talvez poderia ter um outro esquema de dar notas aqui. Acho que estrelas pode remeter as estrelas do Git e confundir as pessoas. Estava imaginando que essas estrelas era popularidade dos projetos no Git. E acho que não, é a popularidade deles aqui no Catálogo né? Seria bom colocar uma diferente nomenclatura para evitar.”

“ Outra dúvida eu fiquei das estrelas né, eu fiquei na dúvida se as estrelas aqui eram aquelas estrelas do github... fiquei na dúvida aqui, apareceu aqui na barra aí depois eu entendi.”

“ Talvez poderia mover active Project para próximo de active Community. Achei um pouco confuso essas duas. Qual a diferença de um para o outro?”

“ acho que deveria adotar um padrão nas máscaras. Em Inserção na Base está um formato diferente dos demais.”

“ acho que deveria adotar um padrão nas máscaras. Em Inserção na Base está um formato diferente dos demais.”

Visibilidade do Status do Sistema

O sistema deve sempre manter os usuários informados sobre o que está acontecendo, por meio de feedback apropriado dentro de um prazo razoável (NIELSEN, 2005). Em alguns momentos da avaliação os professores se queixaram de não terem sido informados sobre o que está acontecendo nos momentos de processamento das buscas pela ferramenta:

“ Não sei se é porque minha internet está ruim. Deixa eu diminuir aqui o domínio... ele ficou o tempo todo como resultado não encontrado. Eu estava visualizando que ele estava pesquisando ainda, e fiquei na dúvida se era internet ou não.”

“ Vou clicar aqui no meio [da página] para vê se aparece algum resultado.”

“ Imaginei que a cada interação fosse atualizar automaticamente, mas acho que esse no project found não atualizou.”

“ Começando aqui com essa imagem do no project found no começo me deu a entender que ele já estava fazendo uma busca, mas depois eu entendi que ele estava esperando uma ação para fazer a busca depois.”

“ Ao abrir aqui pela primeira vez e no project found fica um pouco estranho, então talvez não ter nada ou aguardar uma ação, aguardando uma busca, porque nenhum projeto encontrado dá para entender que realizou uma busca e não encontrou nada.”

“ Aperto no botão e não muda nada, nenhum projeto encontrado.”

Reconhecimento em vez de Recordar

O princípio do reconhecimento em vez de recordar aponta que a carga de memória do usuário deve ser minimizada, a partir da visibilidade de objetos, ações e opções. O usuário não deve ter que se lembrar de informações de uma parte do diálogo

para outra. As instruções de uso do sistema devem ser visíveis ou facilmente recuperáveis quando apropriado (NIELSEN, 2005). No entanto, foram detectados no FlossSearch.Edu problemas que ferem esse princípio conforme relatos a seguir:

" A primeira coisa que eu enxergo é que tenho um level control que eu não tenho a mínima ideia para que serve."

" Eu não sei o que quer dizer maturidade e release."

" Maturidade. O que é maturidade? É o percentual."

" Esse nível de controle que eu me lembre é o que?"

" Eu não entendi muito bem esse nível de controle."

" O que são essas coisas? Quantidade de fork, de star, watch. Não consegui entender."

" Não fica claro o que são projetos ativos."

" Eu fiquei na dúvida o que seria esse número de contribuidores aqui."

" Active project não sei se aqui é percentual..."

" Tamanho do projeto não sei qual a medida de tamanho, acho que seja linha de código. Esse maturidade e release também fiquei confuso porque não vi relação, quanto mais release mais maduro? Aqui também não entendi muito bem [projeto ativo], será que é percentual? "

" O tamanho do projeto aqui por conta do e-mail eu sei que é linhas de código, mas não ficou claro assim de mesmo com e-mail o que que ela tá dizendo, as linhas de código não deixa claro o que é linhas de código. "

" eu não entendi o que é aceitar contribuições. Como é que definiu esse aceitar contribuições? A mesma coisa é da comunidade ativa né."

" Para mim não ficou claro o que definiu esse aceitar contribuições."

" Não tá tão óbvio assim de entender essa questão de aceitar contribuidores, de comunidade ativa."

" Project Size também poderia ser mais claro. É size do que? Megabytes do projeto? Acho que poderia explicitar um pouco mais."

ajuda e documentação

Mesmo que seja melhor se o sistema puder ser usado sem documentação, pode ser necessário fornecer ajuda e documentação. Essas informações devem ser fáceis de pesquisar, focadas na tarefa do usuário, listar etapas concretas a serem realizadas e não ser muito extensas (NIELSEN, 2005). No entanto, mesmo fornecendo uma documentação sobre os critérios, houve relatos de que essa documentação não foi acessada ou localizada.

"eu não cliquei no about. Tinha que dá uma lida nos critérios né? "

"Ah, tem um hover aqui.."/ Ele identificou que parando o mouse no nome do critério aparece a informação sobre ele.

"Acho que eu tinha que dá uma lida aqui nos critérios antes né? "

Controle e Liberdade do Usuário

Os usuários geralmente escolhem as funções do sistema por engano e precisarão de uma "saída de emergência" claramente marcada para deixar o estado indesejado sem ter que passar por um diálogo extenso. Suporte para desfazer e refazer(NIELSEN, 2005). Considerando esse princípio, destacamos os seguintes relatos dos professores:

" Na realidade acho que todos esses dados que aparecem na página principal poderia ser pesquisáveis (filtros)."

" Poderia poder escolher duas linguagens. Mas só posso escolher uma."

" Queria selecionar todos esses projetos. Tem a opção de selecionar? Parece que não.. então vou selecionar um a um."

" Eu não consigo combinar duas linguagens."

" Poderia ter um botão de limpar tudo (Limpar a busca)"

" Uma coisa que seria legal talvez é ter essas informações que estão nesses quadrados para poder ordenar. Se eu quisesse ordenar por estrelas, por número de contribuidores. Então selecionaria aqueles critérios e selecionava os projetos com base na ordenação."

" Eu posso selecionar mais de uma linguagem? Uma coisa aqui que seria interessante é que tem muitos projetos multilinguagens né?"

Localização dos Critérios

Dois professores destacaram a necessidade de mudança na localização do domínio e principais contribuidores, conforme relatos:

" Esse domain achei interessante mas ficou desprivilegiado muito no finalzinho, talvez nem use ele né?"

" Essa parte dos principais contribuidores eu não colocaria aqui, é melhor ter um ambiente para as pessoas comentarem sobre o uso desse projeto em sala de aula do que ter informações como essa aqui."

6.2.4.3 Utilização Futura Dentre os participantes, 70% dos avaliadores apresentaram relatos positivos sobre a abordagem implementada por meio do FlossSearch.Edu:

" Gostei bastante do que vocês estão fazendo. É um problema que acho mega relevante estudar. Uma coisa que falta é a gente ter essa facilidade de encontrar projetos e isso afastou muitos professores. Eu não sou exemplo, porque eu já estou nessa há muito tempo Eu acho que um catálogo como esse aqui poderia facilitar e muito. Achei bem interessante."

" Eu gostei da ideia de ter os critérios para identificar esses possíveis sistemas para serem usados em algo. Não conhecia o trabalho mas achei legal."

" Eu gostei bastante do propósito da ferramenta e vejo que ela tem uma boa contribuição como ferramenta de apoio, quando a gente quiser pesquisar projetos para Engenharia de Software ou áreas correlatas. Acho bem interessante termos alguma coisa desse tipo."

" A ferramenta é interessante. Eu só definiria mais algumas coisas, como a linguagem de programação, se desse para selecionar mais de uma e definir o tamanho de projeto. Porque eu posso ter o tamanho do projeto por função, mas você definiu por linha de código."

" Eu acho que de maneira geral é legal."

" Acho que tem várias coisas que podem adicionar, mas acho que a ferramenta já está numa fase bem legal. No estágio que ela está já é bem útil."

6.2.4.4 Potenciais melhorias para a ferramenta Diante do posicionamento da maioria dos avaliadores, que no geral avaliaram positivamente a ferramenta FlossSearch.Edu, destacamos os relatos de três professores com sugestões de melhorias:

" eu diria que ter mais informações aqui seria bem adequado (informações sobre os critérios de seleção na página de seleção), isso seria mega interessante para eu saber se eu clico ou não."

" Seria interessante você escolher um projeto e querer analisar ele. Então ter um botão do tipo "analisar qualidade" ai ele baixava o projeto e fazia um chek-out. Uma vez que baixou você rodaria algumas métricas de qualidade, complexidade, manutenibilidade e trazer esses resultados, e dizer por exemplo se ele é complexo ou não é complexo."

" Poderia também trazer algum feedback se não encontrar nada. Por exemplo: Não encontramos nada, mas se você marcar tal coisa; ele já inferir isso."

6.3 AMEAÇAS A VALIDADE

Como a abordagem foi avaliada apenas por 24 alunos de uma instituição, e 10 professores de diferentes instituições brasileiras, pode haver viés de amostragem em nossos respondentes do TAM. No entanto, podemos argumentar que, como a ferramenta foi pensada para utilização no contexto acadêmico, para selecionar projetos para utilização no processo de ensino aprendizagem, qualquer aluno de graduação da área de computação, e professores de áreas relacionadas a engenharia de software poderiam atuar como usuários.

Por fim, não é possível generalizar os resultados a partir da percepção de apenas 24 alunos de uma turma e 10 professores escolhidos por conveniência; com certeza, o estudo deverá ser realizado com outros professores com e sem experiência de utilização de FLOSS em sala de aula, e em outras turmas de SE de diferentes instituições.

Capítulo 7

DISCUSSÃO

Os resultados apresentados na revisão sistemática (BRITO et al., 2018) fornecem *insights* sobre como selecionar projetos apropriados para adoção de FLOSS em cursos de engenharia de software. Foi identificado que, na maioria dos casos relatados na literatura, os projetos FLOSS utilizados na EES devem atender às restrições especificadas pelo professor, representadas pelo que é chamado de critérios para a seleção de projetos FLOSS, podendo ser realizada a escolha do(s) projeto(s) pelos alunos do curso. Em outros estudos, foi descoberto que o professor realiza trabalho manual para preparar e avaliar projetos (SMITH et al., 2014) e, posteriormente, fornece aos alunos um projeto ou uma lista de projetos. Os estudos citados reforçam a percepção de que, para selecionar projetos FLOSS, é necessário utilizar critérios para orientar dois potenciais usuários: alunos e professores.

O professor faz a seleção de projetos tipicamente em uma fase de planejamento da disciplina, que antecede sua execução, para escolher um ou mais projetos que possam ser utilizados em atividades práticas, ou como exemplo os realistas para apresentação teórica conceitos. A seleção feita pelos alunos pode ocorrer, por exemplo, em cenários em que os mesmos irão contribuir para um projeto com o qual tenham maior afinidade, ou onde o professor não queira impor um projeto aos seus alunos e delegue a escolha para eles, mas definindo antes alguns critérios de seleção.

Nesse estudo, em relação a perspectiva dos alunos, eles relataram que conseguiram encontrar projetos que atendiam às características desejadas, uma vez que a ferramenta FlossSearch.Edu fornece filtros relacionados aos critérios definidos pelo professor. Também foi possível observar que os alunos que tiveram dificuldade em encontrar manualmente os projetos adequados, relataram ter encontrado facilmente os projetos por meio da flosssearch.edu. Cada equipe de alunos selecionou um projeto FLOSS[1] e analisaram os artefatos dos projetos selecionados e desenvolveram as atividades práticas planejadas pelo instrutor, incluindo a construção de diagramas de classe, sequência e atividades da UML a partir do código-fonte do projeto. Os 5 projetos selecionados foram: `Atom`, `BeTheHero`,

[1] ⟨https://tinyurl.com/projetosselecionados⟩

`WebbAPP`, `JAdventure` e `NewPipe`. Os alunos utilizaram apenas a última versão do código disponível no GitHub para a criação dos diagramas UML[2] e não foram discutidos detalhes sobre as modificações ocorridas em diferentes versões do projeto estudado.

Os alunos indicaram que para selecionar os projetos manualmente no GitHub, era necessário acessar cada repositório de projetos para realizar a verificação manual das características desejadas. Em relação a quantidade de linhas de código do projeto, a maioria dos alunos relataram que não conseguiram identificar essa informação em projetos hospedados no GitHub. Apenas 2 alunos conseguiram localizar essa informação. Para isso, eles utilizaram a extensão Gloc[3]. Quanto ao critério referente à comunidade ativa, todos os alunos declararam não saber identificar o que caracteriza uma comunidade ativa. Em suma, vários alunos apontaram que não possuíam informações suficientes para encontrar projetos adequados de acordo as restrições apontadas pelo professor, seja porque o gitHub não possui opção das buscas desejadas, ou por eles não possuírem conhecimento suficiente sobre o GitHub para usar a busca avançada.

Em relação ao FlossSearch.Edu, questionamos aos alunos se eles sugeriam a inclusão de outros critérios de seleção. Apenas 5 alunos sugeriram inserir outros critérios. As sugestões foram: quantidade de *releases*, pesquisar a partir do tempo em que o projeto esta sendo desenvolvido, perfil do projeto, número de *Pull Requests* e País de Origem do projeto. Vale ressaltar que já é utilizada a quantidade de *releases* nessa ferramenta para indicar a maturidade do projeto. Em relação a avaliação do professor, não foram identificados relatos negativos sobre o critério de Maturidade, no entanto houve discordância quanto à métrica utilizada para indicar a maturidade do projeto. A decisão de utilizar *releases* para indicar Maturidade foi tomada no momento da operacionalização dos critérios, levando em consideração o que é mencionado na literatura. Por exemplo, Ellis, Hislop e Purcell (2013) relatam que para utilização em sala de aula, o projeto deve ter pelo menos uma versão de produção estável. Os projetos que não têm isso podem não ter maturidade suficiente para apoiar a aprendizagem dos alunos. Para Jaccheri e Osterlie (2007), se um projeto FLOSS estiver maduro, bem testado e perto de um lançamento, muitas das tarefas restantes podem ser complexas. Se o objetivo é contribuir para um projeto, o professor deve, pelo menos, estar ciente destas possíveis dificuldades.

Analisando os resultados das avaliações dos professores, chamou-se atenção para um percentual expressivo dos que afirmaram que conseguem selecionar projetos com as características desejadas de forma manual. Possivelmente isso se deve ao fato de que todos os participantes são professores especialistas no uso de projetos FLOSS na EES com experiência em contribuição com comunidades de código aberto. Ainda assim, foi unânime o posicionamento dos professores quanto aos benefícios no uso da ferramenta FlossSearch.Edu relacionados ao tempo de seleção de projetos FLOSS para uso no contexto de EES e a facilidade de uso.

Percebe-se que as informações dos projetos foi a funcionalidade mais comentada pelos professores e a maioria dos relatos sobre problemas na consistência e padrões estão relacionadas às estrelas de classificação de projetos disponibilizadas na ferramenta. Foi

[2] ⟨https://tinyurl.com/diagramascriados⟩

[3] https://chrome.google.com/webstore/detail/github-gloc/kaodcnpebhdbpaeeemkiobcokcnegdki

observado que a maioria dos comentários sobre o princípio do reconhecimento em vez de recordar foram realizados por professores que não optaram pela leitura completa do *about* para se familiarizar com os termos utilizados na ferramenta. Entretanto, os participantes argumentaram que a localização do botão "About"não estava intuitiva.

O nível de controle e número de contribuidores não foram mencionados por qualquer um dos 10 professores que usaram o FlossSearch.Edu. Foi possível observar que a maioria das funcionalidades que os professores não encontraram na ferramenta, já estavam disponível e não foram utilizadas por falta de intuitividade. O princípio da intuitividade foi bastante explorado na avaliação, no entanto os professores não são especialista em usabilidade. Eles deram apenas a opinião a partir de sua experiência de uso.

Quanto à decisão de considerar apenas as *issues* abertas na abordagem implementada, mesmo tendo um professor com interesse em identificar as *issues* fechadas do projeto selecionado, não consideramos interessante a liberação de tal acesso.

A fim de aprimorar a abordagem apresentada nesse estudo, foram realizadas sugestões por professores e alunos. Um estudante sugeriu inserir mais opções de linguagem de programação e mais opções de projetos. No momento da realização desse estudo, havia 193 projetos na base de dados; porém, essa base pode ser facilmente atualizado e ampliada a qualquer momento, com o apoio de scripts desenvolvidos por pesquisadores para esse fim. Foi sugerido também, inserir um botão para realizar a busca, pois a exibição dos projetos é realizada automaticamente de acordo com a inserção dos filtros. Sendo assim, o aluno apontou que não se sabe ao certo se ocorreu ou não a pesquisa com com os filtros desejados. Acatamos a sugestão e realizamos os ajustes necessários. E por fim, foi sugerido uma versão em português da ferramenta. No entanto, optamos por desenvolver a ferramenta em inglês para facilitar seu uso por pesquisadores de diversos países.

Alguns outros participantes sugeriram algumas mudanças e novos recursos relacionados a inserção de outras métricas, tais como qualidade, complexidade e manutenibilidade e utilização de técnicas de recomendação. Foi possível perceber que, embora os professores tenham apresentado muitas sugestões de melhorias sobre a usabilidade da ferramenta na avaliação por meio do think aloud, eles avaliaram positivamente no TAM, sendo unânime a informação de que o FlossSearch.Edu é fácil de usar e fácil de lembrar como executar tarefas. Como esses fatores se correlacionam fortemente com a intenção de uso futuro, acredita-se que o FlossSearch.Edu será usado no futuro, já que em geral foi avaliado como uma ferramenta útil e de fácil usar.

Capítulo

8

CONCLUSÕES

Por muitos anos, a disciplina de Engenharia de Software foi implementada na forma de aulas expositivas combinadas com o desenvolvimento de projetos de pequeno porte ("*toy* examples") que não representavam conhecimentos e habilidades alinhados com padrão da indústria de software. Aos poucos, evidenciou-se a necessidade de refinar currículos, conteúdos, habilidades, objetivos de aprendizagem acadêmica, e utilização de uma abordagem de ensino-aprendizagem mais prática e realista do que apenas aulas expositivas tradicionais, para promover o alinhamento com demandas da indústria de software.

Nesse contexto, esse trabalho investigar o problema da seleção de projetos de código aberto para uso pedagógico na educação em engenharia de software e prover uma solução para apoiar o professor, de forma sistemática, na tarefa de selecionar projetos para uso em disciplinas de engenharia de software.

Vários critérios podem ser usados para encontrar um projeto FLOSS para uso educacional. Nesta dissertação, foi apresentada uma abordagem de seleção de projetos FLOSS para EES que inclui um conjunto de critérios catalogados, coletados após uma cuidadosa revisão da literatura, uma ferramenta da web que apóia a busca por projetos FLOSS adequados com relação a tais critérios. Além disso, foram apresentado os resultados de uma avaliação dos professores com base Think Aloud, e com base na avaliação por meio do TAM realizada por alunos de graduação e professores.

Os resultados indicaram que a abordagem apresentada neste trabalho desempenhou um papel importante no processo de seleção dos projetos FLOSS. A maioria dos participantes (alunos e professores) que lidaram com FlossSearch.Edu relataram que a ferramenta era útil, fácil de usar e que pretendiam usá-la no futuro. Espera-se que professores e alunos possam se beneficiar desta abordagem, utilizando-a para orientá-los no processo de escolha de um ou mais projetos FLOSS com as características desejadas. Embora não seja possível generalizar os resultados a partir das respostas de uma pequena amostra de 24 alunos e 10 professores que participaram desse estudo, acredita-se que, o número de participantes não é inexprimível e esses resultados preliminares podem ser somados ao estado da arte, e promover a adoção de projetos FLOSS no contexto da Educação em Engenharia de Software.

8.1 TRABALHOS FUTUROS

A abordagem de seleção proposta e implementada a ferramenta *FlossSearch.Edu*, em seu estado atual, possibilita filtrar os projetos FLOSS de acordo com um conjunto preliminar de critérios que podem ser utilizados de forma genérica, contudo, como existem diversas áreas da Engenharia de Software em que os projetos FLOSS podem ser utilizados, há uma necessidade de indicação por meio de técnicas de recomendação, em relação as áreas específicas que esses projetos podem ser utilizados (Ex. Testes, Manutenção, Requisitos). Além disso, no meu conhecimento não há estudos sobre quais são os critérios que podem ser utilizados para selecionar um projeto para essas áreas especificas. Sendo assim, uma investigação nesse âmbito pode colaborar com o aperfeiçoamento da pesquisas nessa área.

Os dez critérios de seleção identificados foram documentados em um formato de catálogo para evidenciar informações relevantes e promover o seu reuso. Este catálogo pode crescer à medida em que outros critérios forem identificados ou propostos.

Além disso, dado o possível elevado número de projetos que atendem aos critérios que são indicados pelo professor, pretende-se apoiar a priorização de projetos com base em um ou mais dos critérios, de acordo com diferentes técnicas (por exemplo, processo de hierarquia analítica). A seleção de projetos também pode se basear na experiência documentada por outros professores ao utilizá-los em seus cursos. A eficácia um projeto de código aberto selecionado para uso em uma disciplina envolve diversos aspectos, sendo desejável que uma ou mais de suas características possam ser relacionadas a dimensões de ensino-aprendizagem.

Embora a ferramenta utilize a própria API do GitHub, juntamente com a API do *Black Duck Open Hub* para obter informações sobre os projetos, também seria interessante que a abordagem seja aperfeiçoada, a fim de permitir a seleção de projetos com base em critérios definidos sobre atributos de qualidade do projeto, tais como complexidade e manutenibilidade, usando métricas conhecidas.

REFERÊNCIAS BIBLIOGRÁFICAS

ABDALLAH, F.; TOFFOLON, C.; WARIN, B. Models transformation to implement a project-based collaborative learning (PBCL) scenario: Moodle case study. In: *2008 Eighth IEEE International Conference on Advanced Learning Technologies*. IEEE, 2008. Disponível em: ⟨https://doi.org/10.1109/icalt.2008.174⟩.

ADCOCK, R. et al. *Curriculum Guidelines for Graduate Degree Programs in Software Engineering*. New York, NY, USA, 2009.

ADINARAYANAN, V. et al. An open source approach to enhance industry preparedness of students. In: IEEE. *2014 International Conference on Advances in Computing, Communications and Informatics (ICACCI)*. [S.l.], 2014. p. 194–200.

AMAN, H. et al. A health index of open source projects focusing on pareto distribution of developer's contribution. In: IEEE. *Empirical Software Engineering in Practice (IWESEP), 2017 8th International Workshop on*. [S.l.], 2017. p. 29–34.

AMORIM, S. et al. Understanding the effects of practices on kde ecosystem health. Springer International Publishing, 2017.

ANDRADE, R. M. de C. et al. Retrospective for the last 10 years of teaching software engineering in ufc's computer department. In: ACM. *Proceedings of the 31st Brazilian Symposium on Software Engineering*. [S.l.], 2017. p. 358–367.

ANUYAHONG, B. Effects of communicative competency in english by using project based learning of undergraduate students in thailand. In: *2018 5th International Conference on Business and Industrial Research (ICBIR)*. IEEE, 2018. Disponível em: ⟨https://doi.org/10.1109/icbir.2018.8391263⟩.

BALALI, S. et al. Newcomers' barriers... is that all? an analysis of mentors' and newcomers' barriers in oss projects. *Computer Supported Cooperative Work (CSCW)*, Springer, p. 1–36, 2018.

BANAKHR, F. A.; IQBAL, M. J.; SHAUKAT, N. Active project based learning pedagogies: Learning hardware, software design and wireless sensor instrumentation. In: *2018 IEEE Global Engineering Education Conference (EDUCON)*. IEEE, 2018. Disponível em: ⟨https://doi.org/10.1109/educon.2018.8363463⟩.

BAUTISTA, A. M.; FELIU, T. S. A process to mining issues of software repositories. In: IEEE. *Information Systems and Technologies (CISTI), 2015 10th Iberian Conference on*. [S.l.], 2015. p. 1–6.

BELLOMO, S. et al. Got technical debt?: surfacing elusive technical debt in issue trackers. In: ACM. *Proceedings of the 13th International Conference on Mining Software Repositories*. [S.l.], 2016. p. 327–338.

BENNANI, S. et al. Online project based learning driven by competencies: A systematic strategy proposal for assessment. In: IEEE. *Proceedings of 2012 International Conference on Interactive Mobile and Computer Aided Learning (IMCL)*. [S.l.], 2012. p. 92–97.

BISHOP, J. et al. How to use open source software in education. In: ACM. *Proceedings of the 47th ACM Technical Symposium on Computing Science Education*. [S.l.], 2016. p. 321–322.

BISSYANDÉ, T. F. et al. Got issues? who cares about it? a large scale investigation of issue trackers from github. In: IEEE. *2013 IEEE 24th international symposium on software reliability engineering (ISSRE)*. [S.l.], 2013. p. 188–197.

BOEHM, B. Some future software engineering opportunities and challenges. In: *The future of software engineering*. [S.l.]: Springer, 2011. p. 1–32.

BOLINGER, J. et al. From student to teacher: Transforming industry sponsored student projects into relevant, engaging, and practical curricular materials. In: *2010 IEEE Transforming Engineering Education: Creating Interdisciplinary Skills for Complex Global Environments*. IEEE, 2010. Disponível em: ⟨https://doi.org/10.1109/tee.2010.5508872⟩.

BOURQUE, P.; FAIRLEY, R. E. (Ed.). *SWEBOK: Guide to the Software Engineering Body of Knowledge*. Version 3.0. Los Alamitos, CA: IEEE Computer Society, 2014. ISBN 978-0-7695-5166-1. Disponível em: ⟨http://www.swebok.org/⟩.

BOWRING, J.; BURKE, Q. Shaping software engineering curricula using open source communities. *Journal of Interactive Learning Research*, Assoc. for the Advancement of Computing in Education (AACE), v. 27, n. 1, p. 5–26, 2016.

BRITO, M. S. et al. Floss in software engineering education: an update of a systematic mapping study. In: *Proceedings of the XXXII Brazilian Symposium on Software Engineering*. [S.l.: s.n.], 2018. p. 250–259.

BUCHTA, J. et al. Teaching evolution of open-source projects in software engineering courses. In: IEEE. *null*. [S.l.], 2006. p. 136–144.

BUFFARDI, K. Localized open source collaboration in software engineering education. In: IEEE. *Frontiers in Education Conference (FIE), IEEE*. [S.l.], 2015. p. 1–5.

BUFFARDI, K. Localized open source software projects: Exploring realism and motivation. In: IEEE. *Computer Science & Education (ICCSE), 2016 11th International Conference on*. [S.l.], 2016. p. 382–387.

CASTELLUCCIA, D.; VISAGGIO, G. Teaching evidence-based software engineering: learning by a collaborative mapping study of open source software. *ACM SIGSOFT Software Engineering Notes*, ACM, v. 38, n. 6, p. 1–4, 2013.

CHATZIASIMIDIS, F.; STAMELOS, I. Data collection and analysis of github repositories and users. In: IEEE. *Information, Intelligence, Systems and Applications (IISA), 2015 6th International Conference on*. [S.l.], 2015. p. 1–6.

CHAVEZ, C. et al. Free/libre/open source software development in software engineering education: Opportunities and experiences. *Fórum de Educação em Engenharia de Software (CBSoft'11-SBES-FEES)*, 2011.

CHAVEZ, C. et al. Using free/libre/open source for software engineering education. *CBSoft 2011-SBES-FEES*, 2011.

CHEN, Z.; MEMON, A.; LUO, B. Combining research and education of software testing: a preliminary study. In: ACM. *Proc. of the 29th Annual ACM Symposium on Applied Computing*. [S.l.], 2014. p. 1179–1180.

CICIRELLO, V. Student developed computer science educational tools as software engineering course projects. *Journal of Computing Sciences in Colleges*, Consortium for Computing Sciences in Colleges, v. 32, n. 3, p. 55–61, 2017.

CLEAR, T. et al. Developments in global software engineering education. In: IEEE. *2016 IEEE Frontiers in Education Conference (FIE)*. [S.l.], 2016. p. 1–4.

The 29th IEEE Conference on Software Engineering Education and Training. Disponível em: ⟨http://paris.utdallas.edu/cseet16/⟩.

CURRICULA, T. J. T. F. on C. *Software Engineering 2004: Curriculum Guidelines for Undergraduate Degree Programs in Software Engineering*. New York, NY, USA, 2004. Disponível em: ⟨https://www.acm.org/binaries/content/assets/education/se2004-volume.pdf⟩.

CURRICULA, T. J. T. F. on C. *Curriculum Guidelines for Undergraduate Degree Programs in Software Engineering*. New York, NY, USA, 2015.

DAUN, M. et al. Project-based learning with examples from industry in university courses: An experience report from an undergraduate requirements engineering course. In: *2016 IEEE 29th International Conference on Software Engineering Education and Training (CSEET)*. IEEE, 2016. Disponível em: ⟨https://doi.org/10.1109/cseet.2016.15⟩.

DAVIS, F. D. Perceived usefulness, perceived ease of use, and user acceptance of information technology. *MIS quarterly*, JSTOR, p. 319–340, 1989.

DEURSEN, A. V. et al. A collaborative approach to teaching software architecture. In: ACM. *ACM SIGCSE Technical Symposium on Computer Science Education*. [S.l.], 2017. p. 591–596.

DING, Y. et al. Application of software visualization in programming teaching. In: IEEE. *Computer Science & Education (ICCSE), 9th Intern. Conf. on*. [S.l.], 2014. p. 803–806.

DINIZ, G. C. et al. Using gamification to orient and motivate students to contribute to oss projects. In: IEEE PRESS. *Proceedings of the 10th International Workshop on Cooperative and Human Aspects of Software Engineering*. [S.l.], 2017. p. 36–42.

DORODCHI, M.; DEHBOZORGI, N. Utilizing open source software in teaching practice-based software engineering courses. In: IEEE. *Frontiers in Education Conference, FIE 2016*. [S.l.], 2016. p. 1–5.

DUBOIS, D. L. et al. Effectiveness of mentoring programs for youth: A meta-analytic review. *American journal of community psychology*, Wiley Online Library, v. 30, n. 2, p. 157–197, 2002.

ELLIS, H. et al. Team project experiences in humanitarian free and open source software (hfoss). *ACM Transactions on Computing Education (TOCE)*, ACM, v. 15, n. 4, p. 18, 2015.

ELLIS, H. et al. Learning within a professional environment: shared ownership of an hfoss project. In: ACM. *15th Annual Conference on Information Technology Education*. [S.l.], 2014. p. 95–100.

ELLIS, H. J. et al. Towards a model of faculty development for foss in education. In: IEEE. *Software Engineering Education and Training (CSEE&T), 2013 IEEE 26th Conference on*. [S.l.], 2013. p. 269–273.

ELLIS, H. J.; HISLOP, G. W.; BURDGE, D. Courseware: Hfoss project evaluation. In: ACM. *Proceedings of the 2017 ACM Conference on Innovation and Technology in Computer Science Education*. [S.l.], 2017. p. 90–91.

ELLIS, H. J. et al. How to involve students in foss projects. In: IEEE. *Frontiers in Education Conference (FIE), 2011*. [S.l.], 2011. p. T1H–1.

ELLIS, H. J.; HISLOP, G. W.; PURCELL, M. Project selection for student involvement in humanitarian foss. In: IEEE. *Software Engineering Education and Training (CSEE&T), 2013 IEEE 26th Conference on*. [S.l.], 2013. p. 359–361.

ELLIS, H. J. et al. Holistic software engineering education based on a humanitarian open source project. In: IEEE. *20th Conference on Software Engineering Education & Training, CSEET'07*. [S.l.], 2007. p. 327–335.

ELLIS, H. J.; PURCELL, M.; HISLOP, G. W. An approach for evaluating foss projects for student participation. In: ACM. *Proceedings of the 43rd ACM technical symposium on Computer Science Education*. [S.l.], 2012. p. 415–420.

ERICSSON, K. A.; SIMON, H. A. *Protocol analysis: Verbal reports as data*. [S.l.]: the MIT Press, 1993.

FERNANDES, S.; BARBOSA, L. Collaborative environments in software engineering teaching: A floss approach. In: ACADEMIC CONF. INTERNATIONAL LIMITED. *European Conf. on e-Learning*. [S.l.], 2016. p. 201.

FERNANDES, S. et al. Integrating formal and informal learning through a floss-based innovative approach. In: SPRINGER. *Intern. Conf. on Collaboration and Technology*. [S.l.], 2013. p. 208–214.

FERREIRA, C. et al. When students become contributors: leveraging oss contributions in software engineering courses. In: ACM. *Proceedings of the XXXII Brazilian Symposium on Software Engineering*. [S.l.], 2018. p. 260–269.

FERREIRA, T. et al. Identifying emerging topics and difficulties in software engineering education in brazil. In: ACM. *Proceedings of the XXXII Brazilian Symposium on Software Engineering*. [S.l.], 2018. p. 230–239.

GAMMA, E. *Design patterns: elements of reusable object-oriented software*. [S.l.]: Pearson Education India, 1995.

GAROUSI, V. et al. Closing the gap between software engineering education and industrial needs. *IEEE Software*, IEEE, v. 37, n. 2, p. 68–77, 2019.

GAROUSI, V. et al. Closing the gap between software engineering education and industrial needs. *IEEE Software*, Institute of Electrical and Electronics Engineers (IEEE), p. 1–1, 2019. Disponível em: ⟨https://doi.org/10.1109/ms.2018.2880823⟩.

GOKHALE, S.; SMITH, T.; MCCARTNEY, R. Teaching software engineering from a maintenance-centric view using open-source software. *Journal of Computing Sciences in Colleges*, Consortium for Computing Sciences in Colleges, v. 28, n. 6, p. 189–191, 2013.

GOKHALE, S.; SMITH, T.; MCCARTNEY, R. Teaching software maintenance with open source software: Experiences and lessons. In: IEEE. *2013 IEEE Frontiers in Education Conference (FIE)*. [S.l.], 2013. p. 1664–1670.

GOKHALE, S. S.; SMITH, T.; MCCARTNEY, R. Integrating open source software into software engineering curriculum: Challenges in selecting projects. In: IEEE PRESS. *Proceedings of the First International Workshop on Software Engineering Education Based on Real-World Experiences*. [S.l.], 2012. p. 9–12.

GUTICA, M. Improving students' engagement with large-team software development projects. In: ACM. *Proceedings of the 23rd Annual ACM Conference on Innovation and Technology in Computer Science Education*. [S.l.], 2018. p. 356–357.

HECKMAN, S.; KING, J.; WINTERS, M. Automating software engineering best practices using an open source continuous integration framework. In: ACM. *Proceedings of the 46th ACM Technical Symposium on Computer Science Education*. [S.l.], 2015. p. 677–677.

HISLOP, G.; ELLIS, H. Humanitarian open source software in computing education. *Computer*, IEEE, v. 50, n. 10, p. 98–101, 2017.

HISLOP, G. W. et al. A multi-institutional study of learning via student involvement in humanitarian free and open source software projects. In: ACM. *Proceedings of the eleventh annual International Conference on International Computing Education Research*. [S.l.], 2015. p. 199–206.

HOLMES, R. et al. Lessons learned managing distributed software engineering courses. In: ACM. *Companion Proceedings of the 36th International Conference on Software Engineering*. [S.l.], 2014. p. 321–324.

HORSTMANN, C. Challenges and opportunities in an open source software development course. In: *Free and Open Source Software (FOSS) Symposium*. [S.l.: s.n.], 2009. p. 1–3. Http://www.hfoss.org/symposium09/.

HSIEH, G. et al. Welcome!: social and psychological predictors of volunteer socializers in online communities. In: ACM. *Proceedings of the 2013 conference on Computer supported cooperative work*. [S.l.], 2013. p. 827–838.

IEEE; ACM. *Computer Science Curricula 2013: Curriculum Guidelines for Undergraduate Degree Programs in Computer Science*. 2013. Disponível em: ⟨http://www.acm.org/binaries/content/assets/education/cs2013_web_final.pdf⟩.

JACCHERI, L.; OSTERLIE, T. Open source software: A source of possibilities for software engineering education and empirical software engineering. In: IEEE. *Emerging Trends in FLOSS Research and Development, 2007. FLOSS'07. First International Workshop on*. [S.l.], 2007. p. 5–5.

JEREMIC, Z.; JOVANOVIC, J.; GASEVIC, D. Semantically-enabled project-based collaborative learning of software patterns. In: IEEE. *2009 Ninth IEEE International Conference on Advanced Learning Technologies*. [S.l.], 2009. p. 569–571.

KALLIAMVAKOU, E. et al. The promises and perils of mining github. In: ACM. *Proceedings of the 11th working conference on mining software repositories*. [S.l.], 2014. p. 92–101.

KITCHENHAM, B. A.; DYBA, T.; JORGENSEN, M. Evidence-based software engineering. In: IEEE COMPUTER SOCIETY. *Proceedings of the 26th international conference on software engineering*. [S.l.], 2004. p. 273–281.

KNUDSON, D.; RADERMACHER, A. Software engineering and project management in cs projects vs."real-world" projects: A case study. In: *Proceedings of the IASTED International Conference*. [S.l.: s.n.], 2009. v. 669, n. 094, p. 38.

KOLMOS, A. Reflections on project work and problem-based learning. *European Journal of Engineering Education*, v. 21, n. 2, p. 141–148, 1996.

KON, F. et al. Free and open source software development and research: Opportunities for software engineering. In: IEEE. *Software Engineering (SBES), 2011 25th Brazilian Symposium on*. [S.l.], 2011. p. 82–91.

KRISHNAMOORTHY, V.; APPASAMY, B.; SCAFFIDI, C. Using intelligent tutors to teach students how apis are used for software engineering in practice. *IEEE Transactions on Education*, IEEE, v. 56, n. 3, p. 355–363, 2013.

KRUTZ, D.; MALACHOWSKY, S.; REICHLMAYR, T. Using a real world project in a software testing course. In: ACM. *45th ACM Technical Symposium on Computer Science Education*. [S.l.], 2014. p. 49–54.

KUSSMAUL, C. Experience report: Guiding faculty & students to participate in humanitarian foss communities. In: IEEE. *Technology for Education (T4E), 2016 IEEE Eighth International Conference on*. [S.l.], 2016. p. 224–227.

KUSSMAUL, C. et al. Board# 77: Helping faculty & students to participate in humanitarian free & open source software: The openfe & openpath projects. In: *2017 ASEE Annual Conference & Exposition*. [S.l.: s.n.], 2017.

LAITENBERGER, O.; DREYER, H. M. Evaluating the usefulness and the ease of use of a web-based inspection data collection tool. In: IEEE. *Proceedings Fifth International Software Metrics Symposium. Metrics (Cat. No. 98TB100262)*. [S.l.], 1998. p. 122–132.

LELLI, V. et al. Gamification in remote teaching of se courses: Experience report. In: *Proceedings of the 34th Brazilian Symposium on Software Engineering*. [S.l.: s.n.], 2020. p. 844–853.

LIAO, Z. et al. Exploring the characteristics of issue-related behaviors in github using visualization techniques. *IEEE Access*, IEEE, v. 6, p. 24003–24015, 2018.

MACKELLAR, B. K.; SABIN, M.; TUCKER, A. B. Bridging the academia-industry gap in software engineering: a client-oriented open source software projects course. In: *Open Source Technology: Concepts, Methodologies, Tools, and Applications*. [S.l.]: IGI Global, 2015. p. 1927–1950.

MARQUES, M. R.; QUISPE, A.; OCHOA, S. F. A systematic mapping study on practical approaches to teaching software engineering. In: IEEE. *Frontiers in Education Conference (FIE), 2014 IEEE*. [S.l.], 2014. p. 1–8.

MCIVER, J.; CARMINES, E. G. *Unidimensional scaling*. [S.l.]: Sage, 1981.

MEC. *Diretrizes Curriculares - Cursos de Graduação em Engengaria da Computação*. 2016. Disponível em: ⟨http://portal.mec.gov.br/index.php?option=com_docman&view=download&alias=52101-rces005-16-pdf&category_slug=novembro-2016-pdf&Itemid=30192⟩.

MENEELY, A.; WILLIAMS, L.; GEHRINGER, E. F. Rose: a repository of education-friendly open-source projects. In: ACM. *ACM SIGCSE Bulletin*. [S.l.], 2008. v. 40, n. 3, p. 7–11.

MEYER, B. Software engineering in the academy. *Computer*, IEEE, v. 34, n. 5, p. 28–35, 2001.

MISHRA, A.; CAGILTAY, N. E.; KILIC, O. Software Engineering Education: Some Important Dimensions. *European Journal of Engineering Education*, Taylor & Francis, v. 32, n. 3, p. 349–361, 2007. Disponível em: ⟨https://doi.org/10.1080/03043790701278607⟩.

MISHRA, D.; HACALOGLU, T.; MISHRA, A. Teaching software verification and validation course: A case study. *International Journal of Engineering Education*, v. 30, p. 1476–1485, 2014.

MOORE, M.; POTTS, C. Learning by doing: Goals and experiences of two software engineering project courses. *Software engineering education*, Springer, p. 151–164, 1994.

MORGAN, B.; JENSEN, C. Lessons learned from teaching open source software development. In: SPRINGER. *IFIP International Conference on Open Source Systems*. [S.l.], 2014. p. 133–142.

NANDIGAM, J.; GUDIVADA, V. N.; HAMOU-LHADJ, A. Learning software engineering principles using open source software. In: IEEE. *Frontiers in Education Conference, FIE 2008*. [S.l.], 2008. p. S3H–18.

NASCIMENTO, D. M.; BITTENCOURT, R. A.; CHAVEZ, C. Open source projects in software engineering education: a mapping study. *Computer Science Education*, Taylor & Francis, v. 25, n. 1, p. 67–114, 2015.

NASCIMENTO, D. M. C. *Educacão em engenharia de software com a adoção de projetos de código aberto: uma análise detalhada*. Tese (Doutorado) — Universidade Federal da Bahia, 2017.

NASCIMENTO, D. M. C.; CHAVEZ, C. F. G.; BITTENCOURT, R. A. The Adoption of Open Source Projects in Engineering Education: A Real Software Development Experience. In: *2018 IEEE Frontiers in Education Conference (FIE)*. San Jose, CA, USA: IEEE, 2018. p. 1–9. ISBN 978-1-5386-1174-6. Disponível em: ⟨https://ieeexplore.ieee.org/document/8658908/⟩.

NASCIMENTO, D. M. C.; CHAVEZ, C. v. F. G.; BITTENCOURT, R. A. Does FLOSS in Software Engineering Education Narrow the Theory-Practice Gap? A Study Grounded on Students' Perception. In: *Open Source Systems - 15th IFIP WG 2.13 International Conference, OSS 2019, Montreal, QC, Canada, May 26-27, 2019, Proceedings*. [s.n.], 2019. p. 153–164. Disponível em: ⟨https://doi.org/10.1007/978-3-030-20883-7_14⟩.

NATTASSHA, R.; AZIZAH, F. N. Database analysis and design learning tool based on problem/project-based learning. In: *2015 International Conference on Data and Software Engineering (ICoDSE)*. IEEE, 2015. Disponível em: ⟨https://doi.org/10.1109/icodse.2015.7436973⟩.

NAUMAN, M.; UZAIR, M. Se and cs collaboration: Training students for engineering large, complex systems. In: IEEE. *20th Conference on Software Engineering Education & Training (CSEET'07)*. [S.l.], 2007. p. 167–174.

NIELSEN, J. *Heuristic Evaluation*. 2005. Disponível em: ⟨http://www.useit.com/papers/heuristic/⟩.

OLMSTED-HAWALA, E. L. et al. Think-aloud protocols: Analyzing three different think-aloud protocols with counts of verbalized frustrations in a usability study of an information-rich web site. In: IEEE. *2010 IEEE International Professional Comunication Conference*. [S.l.], 2010. p. 60–66.

PAPADOPOULOS, P.; STAMELOS, I.; MEISZNER, A. Enhancing software engineering education through open source projects: Four years of students' perspectives. *Education and Information Technologies*, Springer, v. 18, n. 2, p. 381–397, 2013.

PARNAS, D. L. Software engineering programs are not computer science programs. *IEEE software*, IEEE, v. 16, n. 6, p. 19–30, 1999.

PETERSEN, K. et al. Systematic mapping studies in software engineering. In: *Ease*. [S.l.: s.n.], 2008. v. 8, p. 68–77.

PINTO, G. H. L. et al. Training software engineers using open-source software: the professors' perspective. In: IEEE. *Software Engineering Education and Training (CSEE&T), 2017 IEEE 30th Conference on*. [S.l.], 2017. p. 117–121.

POSTEMA, M.; MILLER, J.; DICK, M. Including practical software evolution in software engineering education. In: IEEE. *Proceedings 14th Conference on Software Engineering Education and Training.'In search of a software engineering profession'(Cat. No. PR01059)*. [S.l.], 2001. p. 127–135.

PRINCE, M. J.; FELDER, R. M. Inductive teaching and learning methods: Definitions, comparisons, and research bases. *Journal of engineering education*, Wiley Online Library, v. 95, n. 2, p. 123–138, 2006.

QUEK, C. L. Developing an asynchronous computer mediated communication tool for project-based learning. In: IEEE. *2010 4th International Conference on Distance Learning and Education*. [S.l.], 2010. p. 222–225.

RADERMACHER, A.; WALIA, G. Gaps between industry expectations and the abilities of graduates. In: ACM. *Proceeding of the 44th ACM Technical Symposium on Computer Science Education*. [S.l.], 2013. p. 525–530. Investiga quais as habilidades reconhecidas como mais importantes para os profissionais pela indústria de software, quais as áreas em que há maiores deficiências nos recém-formados, e qual a importância de soft skills.

RAJLICH, V. Teaching undergraduate software engineering. In: IEEE. *Software Maintenance (ICSM), 2010 IEEE International Conference on*. [S.l.], 2010. p. 1–2.

RAZALI, R.; CHITSAZ, M. Cases development for teaching software engineering. In: IEEE. *Education Technology and Computer (ICETC), 2010 2nd International Conference on*. [S.l.], 2010. v. 2, p. V2–121.

REICHLMAY, T. J. Collaborating with industry: strategies for an undergraduate software engineering program. In: *Proceedings of the 2006 international workshop on Summit on software engineering education*. [S.l.: s.n.], 2006. p. 13–16.

REKHA, S.; ADINARAYANAN, V. An open source approach to enhance industry preparedness of students. In: IEEE. *Advances in Computing, Communications and Informatics (ICACCI), Intern. Conf. on*. [S.l.], 2014. p. 194–200.

REUTER, R.; BESLMEISL, M.; MOTTOK, J. Work in progress: Teaching-obstacles in higher software engineering education. In: *2017 IEEE Global Engineering Education Conference (EDUCON)*. IEEE, 2017. Disponível em: ⟨https://doi.org/10.1109/educon.2017.7943067⟩.

ROGOZAN, D.; HOTTE, R.; ABDULRAB, H. Modélisation d'un espace dynamique dédié à la réalisation de projets d'apprentissage à distance. In: INSTITUT NATIONAL DES SCIENCES APPLIQUÉES DE LYON. *Technologies de l'Information et de la Communication dans les Enseignements d'ingénieurs et dans l'industrie*. [S.l.], 2002. p. 245–252.

SBC, S. B. da C. *Currículo de Referência para Cursos de Graduação em Bacharelado em Ciência da Computação e Engenharia da Computação*. 2005. Disponível em: ⟨http://portal.mec.gov.br/index.php?option=com_docman&view=download&alias=52101-rces005-16-pdf&category_slug=novembro-2016-pdf&Itemid=30192⟩.

SEO, H. et al. Programmers' build errors: a case study (at google). In: ACM. *Proceedings of the 36th International Conference on Software Engineering*. [S.l.], 2014. p. 724–734.

SHACKELFORD, R. et al. Computing curricula 2005: The overview report. *SIGCSE Bull.*, Association for Computing Machinery, New York, NY, USA, v. 38, n. 1, p. 456–457, mar. 2006. ISSN 0097-8418. Disponível em: ⟨https://doi.org/10.1145/1124706.1121482⟩.

SHIBUYA, B.; TAMAI, T. Understanding the process of participating in open source communities. In: IEEE COMPUTER SOCIETY. *Proceedings of the 2009 ICSE Workshop on Emerging Trends in Free/Libre/Open Source Software Research and Development*. [S.l.], 2009. p. 1–6.

SILVA, F. G. et al. FLOSS in Software Engineering Education: Supporting the Instructor in the Quest for Providing Real Experience for Students. In: *Proceedings of the XXXIII Brazilian Symposium on Software Engineering - SBES 2019*. Salvador, Brazil: ACM Press, 2019. p. 234–243. ISBN 978-1-4503-7651-8. Disponível em: ⟨http://dl.acm.org/citation.cfm?doid=3350768.3353815⟩.

SMITH, T.; GOKHALE, S.; MCCARTNEY, R. Understanding students' preferences of software engineering projects. In: ACM. *Conference on Innovation & Technology in Computer Science Education*. [S.l.], 2014. p. 135–140.

SMITH, T. M. et al. Selecting open source software projects to teach software engineering. In: ACM. *Proceedings of the 45th ACM technical symposium on Computer science education*. [S.l.], 2014. p. 397–402.

SRINIVASA, K.; SOWMYA, B. J. Project based learning for internet of things and data analytics: Experience report of learning from ET601x. In: *2016 IEEE Eighth International Conference on Technology for Education (T4E)*. IEEE, 2016. Disponível em: ⟨https://doi.org/10.1109/t4e.2016.070⟩.

STATISTICS, U. O. L. *Occupational Outlook Handbook - Computer and Information Technology - Software Developers*. 2019. Disponível em: ⟨https://www.bls.gov/ooh/computer-and-information-technology/software-developers.htm⟩.

STEGLICH, C. et al. Hackathons as a pedagogical strategy to engage students to learn and to adopt software engineering practices. In: *Proceedings of the 34th Brazilian Symposium on Software Engineering*. [S.l.: s.n.], 2020. p. 670–679.

STEINMACHER, I. et al. Overcoming open source project entry barriers with a portal for newcomers. In: ACM. *Proceedings of the 38th International Conference on Software Engineering*. [S.l.], 2016. p. 273–284.

STEINMACHER, I. et al. Why do newcomers abandon open source software projects? In: IEEE. *2013 6th International Workshop on Cooperative and Human Aspects of Software Engineering (CHASE)*. [S.l.], 2013. p. 25–32.

STREINER, D. L. Being inconsistent about consistency: When coefficient alpha does and doesn't matter. *Journal of personality assessment*, Taylor & Francis, v. 80, n. 3, p. 217–222, 2003.

SUN, Y. The challenge and practice of creating software engineering curriculum. In: *2011 24th IEEE-CS Conference on Software Engineering Education and Training (CSEE&T)*. IEEE, 2011. Disponível em: ⟨https://doi.org/10.1109/cseet.2011.5876132⟩.

TERCEIRO, A.; SOUZA, R.; CHAVEZ, C. Patterns for engagement in free software projects. In: *Proceedings of the 9th Latin-American Conference on Pattern Languages of Programming*. New York, NY, USA: Association for Computing Machinery, 2012. (SugarLoafPLoP '12). ISBN 9781450327879. Disponível em: ⟨https://doi.org/10.1145/2591028.2600812⟩.

TERCEIRO, A. S. de A. *Caracterização da complexidade estrutural em sistemas de software livre*. Tese (Doutorado) — Universidade Federal da Bahia, 2012.

TVEDT, J. D.; TESORIERO, R.; GARY, K. A. The software factory: An undergraduate computer science curriculum. *Computer Science Education*, Taylor & Francis, v. 12, n. 1-2, p. 91–117, 2002.

VILLARRUBIA, A.; KIM, H. Building a community system to teach collaborative software development. In: IEEE. *Computer Science & Education (ICCSE), 10th International Conference on*. [S.l.], 2015. p. 829–833.

WANG, H. et al. Linking issue tracker with q&a sites for knowledge sharing across communities. *IEEE Transactions on Services Computing*, IEEE, 2015.

WARREN, I. Migrating to a teaching style that facilitates active learning. *CiLTHE Stage*, v. 1, 2002.

WEBSTER, J.; WATSON, R. T. Analyzing the past to prepare for the future: Writing a literature review. *MIS quarterly*, JSTOR, p. xiii–xxiii, 2002.