UNIVERSIDADE FEDERAL DO PARÁ
INSTITUTO DE CIÊNCIAS EXATAS E NATURAIS
FACULDADE DE COMPUTAÇÃO
CURSO DE BACHARELADO EM CIÊNCIA DA COMPUTAÇÃO

CLARICE MORAES FERREIRA

UTILIZAÇÃO DE PROJETOS DE SOFTWARE LIVRE EM DISCIPLINAS DE ENGENHARIA DE SOFTWARE: A PERSPECTIVA DO ALUNO

BELÉM
2018

UNIVERSIDADE FEDERAL DO PARÁ
INSTITUTO DE CIÊNCIAS EXATAS E NATURAIS
FACULDADE DE COMPUTAÇÃO
CURSO DE BACHARELADO EM CIÊNCIA DA COMPUTAÇÃO

CLARICE MORAES FERREIRA

UTILIZAÇÃO DE PROJETOS DE SOFTWARE LIVRE EM DISCIPLINAS DE ENGENHARIA DE SOFTWARE: A PERSPECTIVA DO ALUNO

Trabalho de Conclusão de Curso apresentado como requisito parcial para obtenção de grau de Bacharel em Ciência da Computação, pela Universidade Federal do Pará.

Orientador: Prof. Dr. Gustavo Henrique Lima Pinto

BELÉM
2018

CLARICE MORAES FERREIRA

UTILIZAÇÃO DE PROJETOS DE SOFTWARE LIVRE EM DISCIPLINAS DE ENGENHARIA DE SOFTWARE: A PERSPECTIVA DO ALUNO

Trabalho de Conclusão de Curso apresentado como requisito parcial para obtenção de grau de Bacharel em Ciência da Computação, pela Universidade Federal do Pará.

Data de Aprovação:
Conceito:

Banca Examinadora

Prof. Dr. Gustavo Henrique Lima Pinto
Faculdade de Computação - UFPA
Orientador

Prof. Dr. Cleidson Ronald Botelho de Souza
Faculdade de Computação - UFPA
Membro da Banca

Prof. Dr. Filipe de Oliveira Saraiva
Faculdade de Computação - UFPA
Membro da Banca

BELÉM
2018

A todos que contribuíram para este trabalho.

“A grande sacada do software livre aliado à educação é a gente transpor essa filosofia de colaboração, troca de experiências, de se doar ao próximo. O software livre tem essa filosofia e, na educação, o importante é mostrar isso para o aluno. A partir do momento em que a gente mostra a possibilidade do protagonismo, isso é transformador". (Ronald da Costa)

RESUMO

Disciplinas tradicionais de Engenharia de Software comumente priorizam o ensino de metodologias e conceitos em pequenos e controlados ambientes. Essa decisão é em parte justificada pela dificuldade de se trazer para o contexto de uma sala de aula projetos de software reais. A onipresença de projetos de software livre, no entanto, contribui para a mitigação deste problema. Em particular, diversos professores já adotam tarefas de contribuição em projetos de software livre como parte do processo de ensino e avaliação em suas disciplinas. No entanto, pouco se sabe sobre qual a percepção dos alunos em ter que contribuir com sistemas de software livre no contexto de uma disciplina. Este trabalho tem como objetivo esclarecer os desafios, benefícios e percepção dos alunos. Para esse fim, foram conduzidas 14 entrevistas semi-estruturadas com alunos que cursaram tais disciplinas em cinco instituições brasileiras diferentes, resultando em diversos achados. Por exemplo, foi observado que embora os professores indiquem os projetos para os alunos trabalharem, os alunos (e também a comunidade dos projetos) participam do processo de escolha dos projetos e das tarefas (*issues*). Além disso, foi identificado também que as contribuições dos alunos variaram tanto em termos de quantidade de adições e deleções em *commits*, quanto pelo uso de diferentes linguagens de programação.

Palavras-chave: Software Livre. Engenharia de Software. Educação. Comunidades.

ABSTRACT

Traditional Software Engineering disciplines commonly prioritize the teaching of methodologies and concepts in small and controlled environments. This decision is partly justified by the difficulty of bringing real software projects into the classroom context. The omnipresence of free software projects, however, contributes to the mitigation of this problem. In particular, several professors already undertake tasks of contribution in free software projects as part of the teaching and evaluation process in their disciplines. However, not so much is known about the students' perception of having to contribute free software systems in the context of a discipline. This paper aims to clarify the challenges, benefits and perceptions of students. To this end, 14 semi-structured interviews were conducted with students who studied these subjects in five different Brazilian institutions, resulting in several findings not so well known. For example, observing that although teachers indicate the projects for students to work on, students (and also the project community) participate in the process of choosing projects and issues. In addition, we also identified that student contributions varied in terms of both the number of additions and deletions in commits and the use of different programming languages.

Keywords: Free software. Software Engineering. Education. Communities.

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

UFBA	Universidade Federal da Bahia
UFPE	Universidade Federal de Pernambuco
UnB	Universidade de Brasília
USP	Universidade de São Paulo
UTFPR	Universidade Tecnológica Federal do Paraná

SUMÁRIO

1 INTRODUÇÃO

A disciplina de Engenharia de Software (ES) é uma das mais desafiadoras para ensinar e aprender, Sedelmaier e Landes (SEDELMAIER; LANDES, 2013) citam como desafio a falta de um padrão curricular, e a dificuldade em descobrir quais tópicos específicos e competências devem ser aprendidos ou adquiridos por um grupo específico de estudantes. Existem os cursos tradicionais de engenharia de software, que enfatizam as metodologias e conceitos teóricos, mas não tem muito foco em ensinar os alunos a lidar com sistemas de software existentes e complexos (HOLMES; ALLEN; CRAIG, 2018; CAWLEY et al., 2014). Além disso, a área de desenvolvimento de software requer habilidades que vão além do conhecimento técnico, visto que a construção de um software é resultado de um esforço colaborativo, em que os aspectos humanos e sociais também são protagonistas, conforme constatado na pesquisa de Pinto *et al.* (2017). Ainda, a indústria de software depende de desenvolvedores de software que possam lidar com o código legado, mas tem pouco tempo disponível para treinar mão-de-obra, ainda que seja promissora.

Para alinhar com o atual panorama da área de desenvolvimento de software, o ensino de ES deve também incluir habilidades e atitudes que vão além de conceitos, métodos e técnicas (NASCIMENTO et al., 2013). Isso requer ir além das fronteiras do ensino tradicional, dando mais atenção à complexidade das interações e de como se dá a construção de software colaborativamente, em um ambiente real (PINTO et al., 2017).

Entretanto, não é fácil trazer projetos desenvolvidos na indústria para dentro da sala de aula, ou interagir com empresas no contexto educacional. Uma estratégia comum é, então, conduzir projetos colaborativos, em grupos formados com alunos da própria disciplina. Contudo, esses projetos são pequenos, em grande parte, *toy projects* desenvolvidos desde o início, e o software produzido, em geral, não é algo que se torne um produto para um usuário final. Conforme BUCHTA et al. (2006), desenvolver programas desde o início não corresponde com o que é práticado na indústria, em que desenvolvedores passam a maior parte do tempo evoluindo sistemas preexistentes de médio a grande porte.

A fim de preencher essa lacuna, uma abordagem que tem se tornado cada vez mais comum é promover a participação dos alunos em projetos de software livre (NASCIMENTO et al., 2013; PINTO et al., 2017). Projetos de software livre são ambientes de colaboração intensa, em que uma comunidade interage a fim de construir e evoluir software. A participação nesse tipo de projeto possibilita que os alunos interajam com sistemas reais, problemas reais e com uma equipe de desenvolvimento interessada em construir software de qualidade. Assim, os estudantes têm a oportunidade de aprender habilidades, atitudes e os desafios inerentes ao desenvolvimento de software no mundo real, o que pode aumentar a confiança dos estudantes quando forem iniciar suas atividades profissionais (CHAVEZ et al., 2011).

Conforme Pinto et al. (2017), em que investigou-se a perspectiva dos professores com

relação à utilização de projetos de software livre em cursos de Engenharia de Software, os alunos foram avaliados pelas contribuições realizadas nos projetos de software livre. Adotar esse tipo de prática melhora não apenas as habilidades técnicas, mas também desenvolve as habilidades sociais dos estudantes. Além disso, os professores ressaltaram que a possibilidade de ter contribuições em projetos de software livre é benéfico por compor um portfólio a ser utilizado em futuras oportunidades de emprego. Com relação aos desafios, os professores relatam que a restrição de tempo da disciplina pode dificultar o engajamento total dos alunos com os projetos, e torna o acompanhamento individual muito raso. No entanto, pouco se sabe sobre a percepção desses alunos em relação a esse tipo de abordagem de ensino no que tange as dificuldades enfretadas ao longo da disciplina, como o aluno vê a interação com as comunidades de software livre, quais os são os benefícios obtidos, e o que ele sugere como melhoria para essas disciplinas. Ter a visão do aluno si torna importante por eles serem os principais atores nessa troca de conhecimento entre professor, aluno e comunidade de software livre. Diante disso, este trabalho busca captar e analisar a percepção desses alunos, com intuito de verificar os benefícios, dificuldades e a dinâmica de aprendizagem ao cursar disciplinas com a abordagem de contribuir para um projeto de software livre.

1.1 Justificativa

A importância deste estudo permite entender o que os alunos entendem por benefícios e desafios em suas vivências nessas disciplinas que poderá servir de inspiração e recomendações para professores que desejem se beneficiar da participação de alunos em projetos de software livre. Além disso, os resultados obtidos podem aindar ser de interesse das comunidades de software livre que queiram se beneficiar das contribuições dos estudantes, em contrapartida identificar melhorias na recepção de novos contribuidores nos projetos de software livre.

Para direcionar este estudo foram elaboradas as seguintes questões de pesquisa:

QP1. Como é feita a escolha de projetos de software livre para os alunos contribuírem?

QP2. Como é feita a escolha da tarefa nos projetos para os alunos executarem?

QP3. Qual a natureza das tarefas realizadas pelos alunos?

QP4. Quais são as dificuldades em contribuir para um projeto de software livre no contexto de uma disciplina?

QP5. Quais foram os benefícios em fazer uma disciplina baseada em contribuições para projetos de software livre?

Cada questionamento citado acima parte do pressuposto investigativo em levantar respostas de alunos que cursaram disciplinas com enfoque em projetos de software livre. As perguntas de pesquisa foram, primariamente, respondidas com base nas entrevistas realizadas. Em específico, a QP3 teve sua análise também baseada nas contribuições realizadas pelos alunos nas

plataformas de desenvolvimento utilizadas, além das respostas das entrevistas.

1.2 Objetivos

1.2.1 Objetivo Geral

O objetivo geral deste trabalho é entender a percepção dos alunos que cursaram disciplinas que tinham como proposta a contribuição em projetos de software livre para o ensino de engenharia de software, com intuito de entender suas perspectivas sobre os benefícios, barreiras e a dinâmica de aprendizagem.

1.2.2 Objetivos Específicos

A seguir os objetivos espefíficos deste trabalho:

- Realizar entrevistas semi-estruturadas com os alunos que cursaram disciplinas em que teriam que contribuir com projetos de software livre.
- Realizar análise qualitativa das entrevistas.
- Realizar análise quantitativa das contribuições realizados pelos alunos nos projetos de software livre.

1.3 Estrutura do Trabalho

Este trabalho está organizado em 5 capítulos

- Capítulo 1 - apresenta a introdução deste trabalho, a justificativa, o objetivo e a estrutura do trabalho.
- Capítulo 2 - apresenta os trabalhos relacionados com este estudo.
- Capítulo 3 - apresenta a metodologia utilizada descrevendo como foi feita a seleção dos participantes, o projetos de software livre escolhidos, a condução e a análise das entrevistas.
- Capítulo 4 - apresenta os resultados obtidos após a análise das entrevistas e dos repositórios dos projetos de software livre que obtiveram contribuições dos alunos, assim como as implicações identificadas em outros achados.
- Capítulo 5 - apresenta as considerações finais e limitações deste trabalho, assim como os trabalhos futuros.

2 TRABALHOS RELACIONADOS

Diversos esforços envolveram alunos em projetos de software livre na sala de aula (MORGAN; JENSEN, 2014; SMITH et al., 2014; BUCHTA et al., 2006; SARMA et al., 2016). Smith e outros (SMITH et al., 2014) se concentraram em selecionar os projetos de software livre mais apropriados para os alunos trabalharem. Os autores argumentaram que, devido à curta duração de uma disciplina típica de Engenharia de Software (4-5 meses), os professores de ES não deveriam selecionar projetos de software livre grandes ou complexos, porque os alunos teriam dificuldade em contribuir significativamente durante o curso.

Da mesma forma, outros trabalhos sugerem que os projetos de software livre não devem ser muito pequenos ou muito simples, uma vez que os alunos não colocariam em prática muitos princípios de ES. Morgan e Jensen (MORGAN; JENSEN, 2014) detalharam a experiência de ministrar disciplinas de ES com o apoio de software livre e observaram que o tamanho do projeto (nesse caso, o Ubuntu) acabou sendo um obstáculo significativo para os alunos contribuírem. Da mesma forma, Buchta e seus colegas (BUCHTA et al., 2006) relataram experiências de ensinar evolução de software em disciplinas de ES. Os autores relataram tempo gasto de 60 horas, antes da disciplina, para selecionar os projetos e configurar o sistema de controle. Neste estudo, no entanto, os autores não discutiram os benefícios, implicações ou desafios da utilização desta metodologia para a execução de uma disciplina de ES.

Além disso, Holmes e colegas (HOLMES et al., 2014) relataram que em seu programa de Trabalhos de Conclusão de Curso em Projetos de Software Livre (UCOSP – do inglês *Undergraduate Capstone Open Source Projects*), os projetos devem manter um *issue tracker* ativo, usar um sistema de controle de versão e ter processo de revisão de código definido. Mais tarde, Holmes et al. (HOLMES; ALLEN; CRAIG, 2018) também analisaram a percepção dos alunos com relação à adoção de contribuição para projetos de sofwtare livre como Trabalho de Conclusão de Curso. Eles observaram que os alunos valorizam muito a oportunidade de aplicar seu conhecimento a tarefas reais, em projetos reais com uma comunidade de usuários, enquanto recebem orientação real de membros da equipe de desenvolvimento. Além disso, relatam que contribuir para projetos reais fornece maior compreensão da engenharia de software do que poderia ser obtido através de meios mais tradicionais. Os resultados de Holmes e colegas se alinham aos resultados obtidos por nós na presente pesquisa.

No estudo de Petrenko *et al.* (PETRENKO et al., 2007), os autores descreveram seu sistema de avaliação: os alunos são classificados com base em seu esforço individual, que inclui implementar, depurar e testar mudanças, mas também justificar suas decisões arquitetônicas. No trabalho de van Deursen *et al.* (DEURSEN et al., 2017), a avaliação é baseada em equipe (por exemplo, apresentações, análise de código, etc.) e entregas individuais (revisão do código, participação em aulas, etc).

Embora software livre já tenha uma presença formal no ensino de ES desde a última

década (HISLOP et al., 2009; SARMA et al., 2016), a maioria dos trabalhos publicados nessa área foram relatórios de experiência (*eg,* (MORGAN; JENSEN, 2014; SMITH et al., 2014; BUCHTA et al., 2006; HOLMES et al., 2014)). Esses relatórios não capturam todo o espectro de dinâmicas que podem surgir nessas disciplinas. Além disso, como esses estudos fazem diferentes perguntas de pesquisa, os resultados não são unificados e, dessa forma, são dificilmente comparáveis. O estudo conduzido com professores (PINTO et al., 2017) alinha-se com este presente estudo, pois mostra que as percepções dos professores são alinhadas às dos alunos ao que se refere aos benefícios e desafios. O presente trabalho complementa aquele (PINTO et al., 2017) por buscar entender a perspectiva dos alunos com relação à importância dessas atividades na formação acadêmica e como os alunos tem interagido/contribuído com as comunidades. Os trabalhos de Holmes *et al.* (HOLMES et al., 2014; HOLMES; ALLEN; CRAIG, 2018) são os que mais se aproximam deste trabalho. Pode-se dizer que os resultados são complementares, visto que foram analisadas diferentes abordagens de uso de Software Livre (sala de aula versus trabalho de conclusão de curso). Além disso, as questões de pesquisa aqui apresentadas visam cobrir diferentes aspectos, incluindo o entendimento da natureza das contribuições, bem como as especificidades da escolha de tarefas e dificuldades enfrentadas.

3 METODOLOGIA

A finalidade desta pesquisa é Aplicada ou Tecnológica. O seu objetivo é exploratório visto que busca conhecer os fatos e fenômenos que ocorreram nessas disciplinas sob a visão do aluno. O procedimento para a coleta de dados si deu por meio de entrevistas semiestruturadas realizadas por teleconferência. A amostragem é qualitativa ao si analisar as entrevistas e identificar categorias que respondem as perguntas de pesquisa. No entanto, ao si fazer a análise das contribuições nos repositórios oficiais dos projetos de software livre, a amostragem obtida passou a ser quantitativa pelo número de arquivos alterados, seja por adição ou remoção de código fonte.

3.1 Participantes

A seleção dos participantes da pesquisa foi realizada com base nos contatos estabelecidos com os professores que ministraram disciplinas em que o aluno deveria realizar contribuições em projetos de software livre. Ressaltando, que essas disciplinas ocorreram nos anos de 2012 e 2016, e foram cursados no período de um semestre, o que equivale a aproximadamente 5 meses. Os professores disponibilizam os contatos dos alunos que participaram dessas disciplinas. O contato foi realizado por e-mail com os potenciais participantes, de forma que o número de participantes de cada universidade não destoasse entre elas. Durante 5 semanas foram enviados 52 convites (aproximadamente 10 convites por semana), dos quais 20 alunos manifestaram interesse na pesquisa; e, ao final, somente 16 foram entrevistados. As duas primeiras entrevistas foram realizadas em formato piloto, para avaliar o roteiro da entrevista e a adequação das perguntas. Após as duas entrevistas-piloto, o roteiro foi revisado e algumas perguntas foram removidas, enquanto outras perguntas foram adicionadas. Os dois participantes das entrevistas piloto foram removidos e não fazem parte do grupo de participantes cujas entrevistas foram analisadas. Os dados demográficos dos 14 participantes entrevistados são apresentados na Tabela 1.

Tabela 1 – Dados demográficos dos participantes.

Participante	Idade	Gênero	Universidade	Nível da Disciplina	Ocupação	Experiência Prévia?
P1	27	Feminino	UFPE	Graduação	Engenheiro de Software	Não
P2	23	Feminino	UTFPR	Graduação	Engenheiro de Software	Não
P3	25	Feminino	UTFPR	Graduação	Estudante	Não
P4	24	Masculino	UnB	Graduação	Professor Substituto	Sim
P5	25	Masculino	UTFPR	Graduação	Estagiário	Sim
P6	26	Masculino	UnB	Graduação	Engenheiro de Software	Sim
P7	26	Masculino	UFPE	Graduação	Engenheiro de Software	Não
P8	24	Masculino	UFPE	Graduação	Engenheiro de Software	Não
P9	27	Masculino	UnB/ USP	Graduação / Pós-Graduação	Estudante	Não
P10	25	Feminino	UnB	Graduação	Consultor de desenvolvimento	Não
P11	30	Masculino	UnB	Graduação	Estudante	Não
P12	34	Masculino	UFBA	Pós-Graduação	Analista de Sistemas	Não
P13	33	Masculino	UFBA	Pós-Graduação	Empresário	Sim
P14	31	Masculino	UFPE	Graduação	Engenheiro de Software	Sim

Fonte: O Autor (2018)

A Tabela 1 apresenta um sumário demográfico dos alunos que participaram das entrevistas deste trabalho. Os 14 entrevistados estão identificados de P1 até P14 na primeira coluna, apresentando suas idades e seu gênero nas colunas seguintes. A média de idade é de 27 anos, na data das entrevistas. Dos 14 entrevistados, 4 eram mulheres. As universidades onde eles cursaram as disciplinas estão identificadas na coluna Universidade, que são ao total 5 (UFPE, UFBA, UnB, USP e UTFPR). Adicionalmente, a coluna nível da disciplina refere-se se é um curso de graduação ou pós-graduação. Por exemplo, o participante P9 cursou duas disciplinas (uma na graduação e outra na pós-graduação), em duas instituições diferentes. Nesse caso, a percepção desse participante foi dividida de acordo com a instituição que fez a disciplina. Em relação a ocupação, 12 dos participantes haviam concluído o curso em questão quando entrevistados, com a maioria deles atuando como Engenheiros de Software profissionalmente. Por fim, cinco participantes tinham experiência com projetos de software livre antes de participarem da disciplina em questão.

3.2 Projetos de Software Livre

A Tabela 2 apresenta os projetos de software livre que receberam contribuições dos alunos (P#), bem como os dados quantitativos sobre esses projetos: número de contribuições (#C), ou seja, *commits*; número de contribuidores diferentes (#CT); número de linhas de código (LOC), e linguagem de programação (LP) mais utilizada no projeto. O número de linhas de código foram calculadas através do ferramenta `cloc`, que considera linhas em branco e comentários.

Tabela 2 – Projetos de software livre que receberam contribuições dos alunos.

P#	Projeto	#C	# CT	# LOC	LP
P1	`Catch-The-Pigeon`	197	11	7K	Java
P2	`JabRef`	11K	143	138K	Java
P2	`Gnome Music`	2K	198	31K	Python
P3	`L.Office Impress`	1K<	39	18k	Objective-C
P4	`Noosfero`	15k	126	631K	JavaScript
P6, P9	`Mezuro/Prezento`	1.7K	35	13K	Ruby
P7	`Diaspora`	19K	338	151K	Ruby
P8	`Amadeus`	3K	13	114K	Python
P9	`Mezuro/Kalibro`	1.2K	12	7K	Ruby
P9	`Yosys`	3.8k	54	121K	C++
P10	`Radar Parlamentar`	2k	56	35K	Python
P10	`GestorPSI`	2k	14	103K	Python
P4, P11	`Analizo`	1.1k	18	7K	Perl
P13	`CakePHP`	35k	505	184K	PHP
P14	`Liferay-Portal`	264k	478	5,262K	Java

3.3 Entrevistas

Para a condução das entrevistas semi-estruturadas foi elaborado um roteiro com questões do tipo aberta (a versão final das questões da entrevista pode ser encontrada no Apêndice A). As entrevistas foram realizadas, via teleconferência online (Skype ou Hangout), e gravadas com a permissão dos participantes. As entrevistas foram realizadas no período de outubro a dezembro de 2017, e seus áudios foram transcritos. As 14 entrevistas tiveram a duração em média de 49 minutos (mínimo: 24 minutos; máximo: 77 minutos). As entrevistas foram baseadas nas questões de pesquisa **QP1–QP5** sendo compostas de cinco partes.

1. Perfil do Entrevistado: Perguntado aos participantes as informações demográficas como idade, gênero, escolaridade e ocupação. Além disso, foi solicitado que falassem sobre sua experiência com desenvolvimento de software.

2. Experiência anterior com software livre: Questionado aos entrevistados se eles já tinham contribuído com um projeto de software livre antes da disciplina.

3. Uso de software livre na disciplina: Colocado em discussão a escolha do projeto de software livre, como por exemplo, qual o projeto, a linguagem de programação utilizada, o conhecimentos dos alunos a respeito da linguagem de programação, como era feita as escolhas de tarefas, os tipos de atividades realizadas no projeto, a interação com os mantendores do software livre, bem como as dificuldades em interagir com a comunidade.

4. Específicas sobre a disciplina: Os alunos foram questionados sobre como eram as aulas, as avaliações. Além disso, foram perguntados sobre como avaliavam o seu desempenho, a duração da disciplina e o professor.

5. Encerramento: Ao final da entrevista foi perguntado aos participantes se teriam alguma informação adicional a acrescentar. Além disso, caso o entrevistado tenha realizado alguma contribuição foi solicitado o(s) link(s) da mesma no repositório oficial do projeto de software livre.

Através dos dados coletados e das análises das 14 entrevistas realizadas foram obtidos os insumos para responder as cinco questões de pesquisa.

3.4 Análise das Entrevistas

Para a análise das entrevistas foram realizadas as seguintes etapas:

Codificação inicial dos dados. Nesta etapa foram atribuídos códigos, ou seja, rótulos que dessem significado aos trechos das entrevistas que representassem ações e percepções relevantes dos entrevistados. Os códigos iniciais levantados são considerados provisórios e em alguns casos necessitam de refinamento. Os códigos foram identificados e refinados durante todo o processo (CHARMAZ, 2009).

Categorização dos códigos. Agrupados os códigos inicias, de acordo com características similares em categorias. Esse processo foi realizado durante toda a etapa de análise das entrevistas.

Revisão e definição das categorias. Durante este processo, foram revisadas as categorias, bem como os nomes dados a cada uma delas, para melhor representar os códigos ali agrupados.

Mapeamento das categorias versus quotes. Mapeamento em uma planilha eletrônica com as categorias identificadas na etapa de "Revisão e definição das categorias" com os trechos das entrevistas (quotes), em que partimos das categorias refinadas para os dados mais amplos. Com base nesse mapeamento, identificamos as respostas para a discussão das questões de pesquisa.

4 RESULTADOS

Neste capítulo serão apresentados os resultados agrupados pelas questões de pesquisa. Destacando-se a categoria identificada durante a codificação das entrevistas, a quantidade de ocorrências, e os trechos das entrevistas correspondentes. Com exceção, da QP3 que foi respondida com base no estudo de Hattori e Lanza (HATTORI; LANZA, 2008), além dos trechos das entrevistas.

QP1: Como é feita a escolha de projetos de software livre para os alunos contribuírem?

Dentre as estratégias utilizadas para escolher os projetos de software livre mencionadas pelos estudantes, destacam-se (i) a liberdade para escolha (6 ocorrências), (ii) indicação do professor (4 ocorrências) e (iii) conhecimento prévio sobre a comunidade de software livre (4 ocorrências). A seguir, apresentado essas estratégias.

Liberdade para escolha. De acordo com 6 alunos entrevistados, seus professores deram total liberdade na escolha dos projetos. O aluno P1, por exemplo, mencionou que: "*[...] ele deixou livre em relação a qual o projeto em si a gente queria contribuir*". O participante P3 também mencionou que "*o professor deixou livre a escolha*". Ainda, o estudante P11 afirmou que "*[...] eles apresentavam os conceitos e deixaram livres [...] pessoas escolhessem o projeto*". Adicionalmente, P8 descreveu que "*a gente poderia escolher qualquer projeto de software livre [. . .] nós escolhemos um dos projetos que inclusive havia saído de dentro da universidade, mais por questões de proximidade*". Identificamos outros casos em que o professor da disciplina oferecia liberdade parcial para escolha do projeto. Nesses casos, o professor fornecia algumas diretrizes que os alunos deveriam adotar ao procurarem seus projetos. Por exemplo, P5 mencionou que "*o critério para a escolha do projeto seria a simplicidade do projeto, o conhecimento sobre a linguagem e também a afinidade com o projeto*". Em outra instituição e com outro professor, P9 cita essa liberdade na disciplina de pós-graduação que cursou: "*Foi mais livre, basicamente a única exigência é que o projeto deveria estar ativo, inclusive podia ser correlacionado ao nosso mestrado*".

Indicação do professor. Em disciplinas de diferentes instituições houveram casos em que os projetos foram escolhidos majoritariamente por uma indicação do professor responsável. Quatro alunos relataram que essa indicação foi objetiva, como mencionado pelo participante P2: "*A primeira vez, o professor foi lá e escolheu para gente*". No entanto, houveram casos em que o professor indicava possíveis projetos para que os alunos fizessem uma escolha final. O participante P5 relatou que "*tinha que escolher um projeto do GNOME ou KDE*". A participação do professor em projetos de software livre também aparece como fator relevante para escolha de um projeto. P9 mencionou que "*o professor faz uma triagem com vários mantenedores de alguns projetos já consolidados, depois ele vê quem tem disponibilidade ou não desses mantenedores de dar um certa atenção aos alunos*". Finalmente, o participante P6 indicou uma forma colaborativa

para escolha dos projetos: "*O professor pega a lista de matriculados no início do semestre, e vai distribuindo o pessoal com base na experiência dos alunos com a linguagem de programação do projeto. Caso o aluno não tenha a experiência é colocado como desafio. Ele também pede ajuda para os alunos do laboratório. Todos os alunos dão opinião sobre os projetos, se teve disciplina com o colega ou não, etc. Por fim, o professor verifica o background do aluno. Se, por exemplo, o aluno tem background em C, então ele trabalharia com esse projeto aqui[...]* ". Nesse caso, a escolha do projeto aconteceu utilizando critérios estabelecidos pelo professor ao avaliar o perfil do aluno, que preenche um questionário no início do semestre. Nesse questionário, o aluno indica três opções de projetos, dentre aquelas previamente apresentadas pelo professor em sala de aula. O objetivo era ter equipes equilibradas em termos de conhecimento prévio dos alunos, de forma que o pareamento em sala de aula possa fazer o conhecimento circular entre os integrantes. Os alunos com mais conhecimentos e experiência prévia eram indicados como "coach" (papel definido no método ágil XP) das equipes.

Conhecimento prévio sobre a comunidade do projeto de software. Identificamos também alguns casos em que os projetos eram escolhidos com base no conhecimento prévio sobre a comunidade responsável pelo projeto de software livre em questão. Esse conhecimento prévio sobre o projeto e a comunidade pode ser tanto por parte do professor quanto por parte dos alunos. Por exemplo, um caso em que o aluno conhecia a comunidade foi relatado pelo participante P14 que mencionou: '*escolhi o projeto no qual já era familiarizado*". Além dele, o participante P1 mencionou que "*já conhecia os desenvolvedores que desenvolveram o jogo. Dessa forma, ficou mais fácil a escolha, pois a gente podia contactá-los diretamente*". Em dois casos, o professor tinha contato com a comunidade, um foi apresentado pelo participante P4: "*o professor escolhe quais projetos que a gente tem contato com os desenvolvedores [...] de várias linguagens e tecnologias.*." Outro caso foi relatado pelo participante P10: "*o próprio professor seleciona alguns projetos que ele acha interessante, ou que ele tem contato com alguma pessoa*".

QP2: Como é feita a escolha da tarefa nos projetos para os alunos executarem?

Após escolher o projeto, o próximo passo dos alunos é a seleção das tarefas a serem executadas nos projetos. Para a escolha da tarefa destacam-se (i) escolhendo tarefas simples (2 ocorrências), (ii) analisar issues cadastradas (14 ocorrências, com base que todos os participante mencionaram a análise das issues), (iii) label nas tarefas (2 ocorrências) e (iv) escolha colaborativa (3 ocorrências).

Escolhendo tarefas simples. P5 mencionou que é importante "*pegar uma tarefa periférica, e não tentar pegar uma tarefa da parte principal do projeto*". De forma similar, P8 relatou que "*procurava por issues abertas [...] pegava alguma issue, que achava que conseguiria resolver e que seria interessante para contribuir para o projeto*".

Analisar issues cadastradas. Como vários dos projetos envolvidos nas disciplinas estão hospedados em plataformas de desenvolvimento colaborativo como Github e Gitlab, P6 mencionou o fato de que existe "*issue tracker que tem uma lista de tarefas e problemas que o*

software tem". A busca por tarefas via as *issues* cadastradas no repositório do projeto foi também recorrentemente realizada pelos alunos em diferentes projetos. Por exemplo, P7 ressaltou que "*normalmente, os alunos iam atrás de tarefas no issue tracker*". Participante P5 chamou atenção para o fato de que, antes de procurar uma tarefa, é preciso "*estudar o projeto*". Em seguida, . Ainda, P5 indicou que uma possível tarefa para se familiarizar é "*escolher um bug e tentar reproduzir o bug*".

Labels nas tarefas. Com base nas *issues*, P6 comentou que um particular projeto criou um *label* (uma maneira de etiquetar as *issues*) para identificar "*as issues que os alunos deveriam trabalhar"*. P10 também indicou uso de *label* em *issues* para auxiliar na escolha da tarefa: "*o projeto tem issues que tinham labels que determinavam se era fácil ou tinha alguma tarefa. Os mantenedores desse projeto também auxiliavam na configuração do projeto e orientavam os alunos sobre quais issues eram fáceis. Os alunos determinavam quando iam pegar uma issue maior*".

Escolha colaborativa. Por fim, identificamos uma forma colaborativa para a escolha de tarefas, como relatado por P11: "*era uma junção do professor, com o mantenedor do projeto e os alunos. O mantenedor sugeria algumas coisas, os alunos escolhiam o que queriam fazer, e o professor falava se era viável para o escopo da disciplina.*". Adicionalmente, o participante P9 corroborou com a estratégia colaborativa, ao indicar que "*a definição das tarefas era bem democrática, bem métodos ágeis. Os alunos sentavam um dia da semana junto com o professor, abriam as issues e discutiam o que poderia ser feito*". P7 indicou que "*os alunos tentavam se comunicar com o time de desenvolvimento do projeto para checar se fazia sentido o trabalho previsto, ou se seria algo viável*". A abordagem colaborativa variava entre os projetos e também entre qual ciclo/etapa a equipe estava. Nos ciclos iniciais, geralmente, o professor participava das reuniões de planejamento (e de revisão ao final do ciclo de 15 dias). Portanto, o aluno logo no primeiro mês da disciplina teria a oportunidade de passar por um ciclo completo de interação com a comunidade em questão, o que os faziam melhor entender a dinâmica da disciplina e se motivarem para as contribuições ao projeto ao longo do semestre.

Umas das formas de iniciar a colaboração em um projeto de software livre é começar com uma tarefa mais fácil, que não faça parte do núcleo do software, e se possível indicada também pela própria comunidade. Procurar *issue* ou *bug* com indicações (geralmente, através de *label* nos repositórios) de tarefas que possam ser resolvidas por novatos é uma estratégia recomendada. A proximidade com a comunidade, participando das suas listas e canais de comunicação, inclusive via os comentários na *issues*, pode facilitar a decisão sobre qual tarefa trabalhar. Durante a disciplina, a intermediação do professor com a comunidade e a participação dele no planejamento, principalmente no início da disciplina, é importante para ajudar na escolha de tarefas que possam ser concluídas no curto prazo, o que motiva os alunos ao longo do semestre.

QP3: Qual a natureza das tarefas realizadas pelos alunos?

Neste trabalho, considerado uma contribuição um *commit* aceito no repositório oficial

do projeto. Como nem todos os alunos tiveram suas contribuições aceitas no repositório oficial, apesar de todos terem as enviado para as comunidades e terem sido avaliadas pelo professor independente do aceite, ou ainda colaboraram com a documentação do projeto, os participantes P5 e P12 não estão listados na Tabela 2, por não terem realizados contribuições de código fonte. Por outro lado, há projetos que receberam contribuições de mais de um dos entrevistados, como é o caso do Analizo. Também há participantes que colaboraram com mais de um projeto, na mesma disciplina, como é o caso do entrevistado P9, que colaborou com Prezento e o Kalibro , quando estava na graduação; e colaborou com o projeto Yosys enquanto pós-graduando.

Como é possível perceber na Tabela 2, os projetos são diferentes em termos de tamanho e linguagem de programação. O domínio dos projetos variou entre jogos para plataforma móveis, redes sociais, analisadores de código, gerenciador de clínicas, arcabouço de desenvolvimento web, dentre outros. Alguns dos projetos foram criados e são mantidos por contribuidores da própria universidade (e.g., o *Catch-The-Pigeon*); outros são populares e conhecidos pela ampla comunidade de desenvolvimento de software (e.g., o *CakePHP*).

Para definição do tipo de atividade realizada pelos alunos, foi utilizada a definição de Hattori e Lanza (HATTORI; LANZA, 2008), que categorizam contribuições em código em quatro grandes grupos. Essa classificação se baseia a partir dos comentários existentes nos commits no repositórios dos projetos de software livre. **Forward engineering** (adicionar novas funcionalidades); **Reengineering** (atividades de refatoração); **Corrective** (corrigir bugs), e **Management** (atualizar documentação).

Para cada participante, foram analisados os trechos das entrevistas em que eles explicam suas contribuições. O agrupamento dos tipos de atividades foi apoiado em uma prévia lista de palavras-chave também introduzidos por Hattori e Lanza (HATTORI; LANZA, 2008). Com base nas entrevistas, observou-se que a maioria das contribuições feitas foram do tipo **Forward engineering**; sete participantes relataram contribuições que adicionaram novas funcionalidades (P1, P3, P4, P8, P9, P10 e P11). Por exemplo, o participante P3 mencionou que o seu grupo fez "*uma contribuição no LibreOffice Impress, pra passar os slides com o celular. Não tinha essa funcionalidade, e a gente colocou*".

Atividades do tipo **Corrective** receberam contribuições de seis participantes diferentes (P2, P7, P9, P13 e P14). Por exemplo, o participante P2 observou um bug no "*processo de importação pra banco de dados. Funcionava apenas para o MySQL, mas não funcionava para o Postgres.*". O participante confirmou o bug e implementou a solução. Ademais, apenas três participantes mencionaram contribuições do tipo **Reengineering** (P6, P8 e P9). Dentre os exemplos, destaca-se a contribuição do participante P8, que fez uma *"refatoração de alguns layouts que não estavam muito adequados ao padrão Android*".

Por fim, apenas o participante P10 mencionou realizar contribuições do tipo **Management**. Este participante mencionou que relatava bugs ou adicionava *labels* em *issues* como parte das contribuições. Importante ressaltar que cinco participantes (P4, P6, P9, P10, P11) utilizaram

metodologias ágeis no processo de colaboração para os projetos durante das disciplinas, como destacado por P4: "*A gente fazia todo o processo do Scrum, tinha as sprints, estórias, pontuava as estórias, etc*". O pareamento era uma estratégia utilizada pelo professor para que o conhecimento circulasse entre os alunos e a escrita de testes automatizados era fundamental, pois, em geral, os projetos de software livre não aceitam contribuições sem seus respectivos testes automatizados.

Investigado também os *commits* realizados pelos alunos das disciplinas, daqueles entrevistados que encontraram o repositório *fork* usado na disciplina ou tiveram suas contribuições aceitas no repositório principal do projeto. A amostra analisada neste estudo é de 39 contribuições feitas em 11 projetos de software livre usados nas disciplinas.

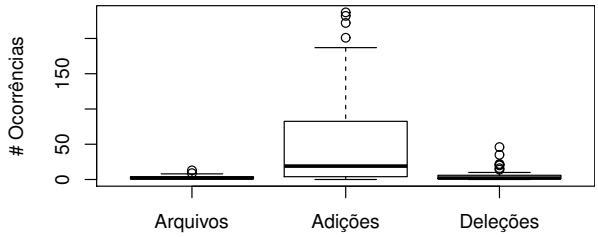

Figura 1 – Distribuição das contribuições feitas pelos alunos nos projetos de software livre.

Ao analisar a quantidade de adições e deleções dessas contribuições, foi observado a presença de vários *outliers*. Por exemplo, uma única contribuição introduziu 2.711 adições e 2.589 deleções em um único commit[1]. Esta contribuição modificou o layout de um jogo, logo foi necessário realizar modificações em 39 arquivos diferentes, possivelmente por se tratar de uma interface de um aplicativo Android modificando também os arquivos XML gerado. Sem considerar os *outliers*, em média, as contribuições modificaram 12,5 arquivos diferentes, introduzindo 56 linhas de código e removendo 7 linhas de código.

A Figura 1 apresenta a distribuição do número de arquivos modificados, adições e remoções realizadas. Por um lado, contribuições pequenas foram também frequentes, ficando de acordo com o apresentado em estudos recentes que sugerem que em projetos de software livre as contribuições tendem a ser pequenas (PINTO; DIAS; STEINMACHER., 2018; GOUSIOS; PINZGER; DEURSEN, 2014; GOUSIOS et al., 2015). Na Figura 1, o primeiro quartil de contribuições inseriu 4 linhas de código e removeu 1 linha de código. Por exemplo, a contribuição[2] feita no projeto `Mezuro/Prezento` foi de uma única adição: trocou uma imagem em uma página web. Por outro lado, em geral, nossa amostra indica uma uma média alta de arquivos modificados e de linhas de códigos alteradas por contribuição (*commit*). A recomendação, e sendo algo que evolui conforme a experiência do desenvolvedor, é que se tenha *commits* pequenos e auto-contidos (por exemplo, com seus testes automatizados), o que facilita a revisão das novas contribuições por parte dos membros das comunidades dos projetos. Dessa forma, contribuições com um certo

[1] <https://github.com/rodolfoasantos/catch-the-pigeon/commit/51607083>
[2] <https://github.com/mezuro/prezento/pull/114/files>

volume de modificações sinalizam a possível falta de experiência dos estudantes no processo colaborativo de desenvolvimento de software livre, o que é o esperado nas disciplinas, uma vez que maioria não tinha experiência prévia de colaboração com projetos de software livre. Portanto, este fato corrobora a importância do aluno passar por este tipo de experiência prática em uma disciplina durante sua formação.

QP4: Quais são as dificuldades em contribuir para um projeto de software livre no contexto de uma disciplina?

A partir das análises dos relatos dos participantes, selecionados os principais obstáculos para contribuir para os projetos de software livre durante as disciplinas: (i) a estrutura do código (8 ocorrências); (ii) a interação com a comunidade (8 ocorrências); (iii) entender e configurar o ambiente de desenvolvimento do projeto (5 ocorrências), e (iv) a duração da disciplina ser insuficiente (2 ocorrências).

Estrutura do código. O estudante P2 relatou que "*[...] a minha dificuldade foi com a estrutura do código [...]configurar o ambiente. Achar um bug em si é difícil[...]. Achar o problema era muito difícil também [...]*". O participante P3 disse " *[...] a gente tinha que analisar o projeto inteiro [...]até você adaptar com esse projeto também é difícil*". O participante P4 falou que "*[...] sempre tem a dificuldade de entender a arquitetura, o código do projeto*". O entrevistado P5 afirmou que "*a dificuldade que eu tive foi com o código, pois não seguia uma convenção [...] parecia um monte de cartas embaralhadas, que a gente não conseguia colocar em ordem*". O aluno P8 disse "*[...]tentar entender como ela estava estruturada [...] e também si adequar as normas que eram especificadas para cada projeto*". Ainda, o entrevistado P7 relatou que a principal dificuldade "*[...] foi mais essa familiarização com o projeto*". Por fim, P9 cita uma questão de curva de aprendizado por conta da "*[...] complexidade do projeto, por ter que lidar com hardware*", assim como P11 relata que "*[...] era difícil por causa da linguagem, a questão da complexidade do projeto*".

Entender e configurar o ambiente do projeto. De acordo com o participante P2 o obstáculo encontrado foi "*[...]configurar o ambiente*". O estudante P1 disse que "*a minha dupla já tinha um conhecimento maior com o ambiente Linux, e eu não*". P5 relatou que "*[...] o sistema de versionamento utilizado era desconhecido*". O aluno P7 afirmou que " *[...] configurar o projeto, às vezes pode dar um pouquinho de trabalho*". Concluindo, o participante P9 disse que foi "*[...] a curva de aprendizado da linguagem, aprender o Git, e entender como a comunidade funciona [...] tudo ao mesmo tempo.*"

Estas dificuldades, do ponto de vista técnico, corroboram o quão real foram as experiências dos alunos em lidar com projetos de software livre. Por exemplo, para mitigar este tipo de problema quanto à curva de aprendizado do projeto e código, um dos professores acompanhava mais de perto o projeto e organizava as duplas de forma que quem tinha mais dificuldade tivesse a ajuda de alguém mais confortável com o código do projeto. O professor entrava em contato diretamente com mantenedores do projeto para agendar vídeo-conferências para sanar as dúvidas.

Interação com a comunidade. De acordo com os relatos, durante o desenvolvimento das atividades, os alunos muitas vezes interagiam diretamente com as comunidades. É algo necessário e positivo para a experiência de colaboração, mas, de um lado, essa comunicação nem sempre é fácil. O estudante P3 relata que "*interagir nas listas de discussões é difícil, por que você não sabe quem é que vai te responder [...]alguém do nosso grupo mandou mensagem na lista de discussão, e um cara foi bem ríspido*". O estudante P4 cita a dificuldade em entender "*[...] como é a organização da comunidade em si [...]*". O P5 disse que "*[...] existem revisores [...] corrigir aquilo que você fez, por um lado é bom, mas por outro a rigorosidade do processo inibe que novos contribuidores se engaje.*". O aluno P9 afirmou que a dificuldade estava em "*[...] entender como a comunidade funciona[...]*". Já P11 cita como obstáculo "*[...] você conseguir o primeiro contato com alguém da comunidade[...]*", o que impacta em algo enfatizado por P13 com principal dificuldade: "*[...] foi escolher o que fazer*".

Por outro lado, por ser a primeira experiência de muitos em uma projeto real, lidando com desenvolvedores acostumados a colocar software em produção, parte das dificuldades tinham origem em um receio em como interagir com a comunidade do projeto. P10 cita que teve "*[...] medo de contribuir [...] dificuldade inicial de entender como funciona, de ter coragem de submeter um pull-request, porque um commiter irá avaliar isso, medo de fazer uma besteira e quebrar tudo"*. P9 fala em "*[...] não saber o que fazer para o meu nível*". P11 completa: "*[...] acho que começar é mais difícil, perder o medo de mexer no código, mas depois que você quebra esse medo as coisas começam a andar um pouco melhor*". Por isso, o recomendado pelos professores que passaram por estas experiências em sala de aula (PINTO et al., 2017) para mitigar essas dificuldades inclui encorajar os alunos, passar confiança e, quando necessário, especialmente no início da disciplina, fazer diretamente a comunicação com a comunidade, também estando presente nas listas de discussão e nos canais de comunicação.

Além disso, o atraso na avaliação das contribuições por parte das comunidades impacta nessa interação com os estudantes. P12 cita como dificuldade '*[...] entrar em contato com o pessoal, a gente tinha algumas dúvidas sobre como fazer algumas coisas, a gente teve uma certa dificuldade em alguns momentos conseguir obter o retorno num prazo satisfatório*". Por exemplo, P9 mencionou que fez quatro contribuições para um projeto, no entanto, *"até agora, dois dos meus quatro pull-requests estão abertos [...] eles até hoje não responderam*". P1 também percebeu comportamento similar em trabalhos de colegas: "*alguns colegas tiveram feedback rápido e conseguiram contribuir efetivamente [...] Tiveram outros colegas que submeteram, mas não tiveram a contribuição avaliada durante a disciplina. Até hoje não sei se foi aceita ou não*". Comunidades de software livre tendem a demorar mais para processar contribuições quando o autor não é um integrante ativo da comunidade (PINTO; DIAS; STEINMACHER., 2018). Essa assertiva foi também observada pelos participantes deste estudo, e os professores devem ficar atendo para essa demora não desmotivar os alunos. O contato do professor com os membros das comunidades, antes do início da disciplina, explicando a metodologia em sala de aula e as necessidades dos alunos, é importante para mitigar a falta de retorno nas avaliações das

contribuições enviadas (PINTO et al., 2017).

Duração da disciplina insuficiente. Quando questionados sobre sugestões para as disciplinas que adotam software livre como alvo de contribuições, dado a dificuldade de se ambientar com o projeto, alguns participantes mencionaram que a duração da disciplina foi insuficiente. O participante P4 mencionou que "*[...] essa disciplina poderia, inclusive, ser divida em duas: a primeira, com foco em como se ambientar na comunidade e a segunda, o qual os alunos de fato contribuiriam com o projeto de software livre*". De forma similar, o participante P1 mencionou que a disciplina poderia ser "*[...]dividida em dois semestres*". O participante P11 foi mais extremo e mencionou que "*[...]toda disciplina de graduação poderia utilizar software livre. Sempre que um professor precisar mostrar código aos alunos, o professor poderia mostrar a implementação em um projeto software livre*". Esta dificuldade gerada pela duração da disciplina, uma vez que a contribuição dos alunos para projetos de software livre é uma iniciativa individual de alguns professores em seus cursos, foi citada no estudo que tratou a perspectiva dos professores (PINTO et al., 2017), e é corroborada aqui na visão dos estudantes.

Entender a estrutura do código, interagir com a comunidade, entender e configurar o ambiente do projeto, bem como o tempo insuficiente da disciplina, foram as principais dificuldades que os estudantes ressaltaram nas entrevistas. Em conformidade com os relatos, para atenuar o problema sobre interagir com a comunidade, o professor indicava os projetos que tinha contato com mantenedores. Para facilitar o entendimento da estrutura do código, segundo relatos de entrevistados, uma estratégia é a escolha de um projeto que o aluno já tenha conhecimento sobre a linguagem de programação usada. De toda forma, as dificuldades relatadas pelos entrevistados serviram, de acordo com as informações fornecidas, para obterem uma maturidade na área de desenvolvimento de software. De uma outra perspectiva, se este tipo de iniciativa com projetos de software livre fosse adotada em mais de uma disciplina, conforme sugerido por um dos alunos, a questão do tempo insuficiente da disciplina para uma experiência ainda mais completa seria mitigada.

QP5: Quais foram os benefícios em fazer uma disciplina baseada em contribuições para projetos de software livre?

Questionado aos estudantes sobre os benefícios de cursar uma disciplina com ênfase em contribuições para projetos de software livre. Baseado nos relatos, foram identificados três benefícios: (i) participar de um projeto real (7 ocorrências), (ii) aperfeiçoar o portfólio (11 ocorrências) e (iii) continuar contribuindo após a disciplina (8 ocorrências).

Participar de um projeto real. Os alunos se sentiram motivados e se mostraram satisfeitos por terem a chance de participar de um projeto real. O aluno P2, por exemplo, cita que " *[...] te dá um pouco mais de confiança, que você colabora, e aceita, um negócio real, que tá no ar, qualquer pessoa pode utilizar [...] é o mais próximo da realidade que a gente vai ter, a experiência mais próxima do mundo real*". P4 adicionou que "*[...] é uma grande oportunidade para o aluno. Primeiro, para ter essa parte de contribuir com um software grande que já existe*

[...] e também dá a oportunidade para ele conhecer que existem projetos de software livre e que não é um bicho-de-sete-cabeças contribuir com eles". Já P14 citou que o "*[...] conhecimento filosófico, da importância de você contribuir e colaborar com um repositório de software livre*". E, de maneira interessante, P13 conclui que "*[...] não era um bicho-de-sete-cabeças colaborar com código*".

Do ponto de vista pessoal em relação a trabalhar com um projeto real, P7 compartilha que "*[...] você fica bem mais contente de saber que você aprendeu, e além de ter aprendido, você contribuiu um pouquinho com um projeto que está sendo usado*". O entrevistado P12 também aborda que "*[...] você poder ver como é que o ecossistema do projeto sobrevive, como é que se consegue trabalhar com uma rede formada basicamente de desenvolvedores*". Nesse contexto, P9 completa: "*[...] a visão que software não é feito por uma pessoa, mas é feito colaborativamente, por várias pessoas, seja livre ou não*".

Aperfeiçoar o portfólio. De acordo com o participante P8, a experiência na disciplina "*[...] foi o ponto de partida pra ingressar nesse mundo de software livre*". O estudante P6 relatou que "*[...] hoje na empresa onde eu trabalho, eu sou referência no versionamento (controle de versão e repositório de código), exatamente por conta dessa experiência[...] um reconhecimento e destaque pessoal, que é chegar em uma empresa e falar que tinha tido contribuições em projetos de software livre na entrevista. Isso é um benefício imensurável*". P10 adicionou que foi uma oportunidade para "*[...] trazer um portfólio, que são contribuições que estão no nosso GitHub, e a gente consegue levar em uma entrevista de emprego, e isso é bem visto[...] então muitas pessoas que gostaram e que contribuíram fortemente dentro das disciplinas conseguiram mestrados, empregos em grandes empresas no exterior, e muito por causa do portfólio que a gente vai fazendo enquanto estuda*".

Continuar contribuindo. Com relação à continuidade nas contribuições para projetos de software livre após a disciplina, oito participantes mencionaram que continuaram a contribuir. O relato mais rico foi feito por P11: "*depois da disciplina eu passei bastante tempo procurando me envolver em algum projeto. Eu comecei a mandar pull-request em projetos do GitHub que achava interessante. Até encontrar um projeto específico e contribuo pro projeto já tem um tempo*". O entrevistado P11 completou seu relato destacando: "*eu contribuo com software livre, viajo para conferências, trabalho com software livre. Então, fez bastante diferença na minha vida. A gente resolveu criar uma ferramenta, o Kiskadee [...] a gente colocou essa ferramenta no Google Summer of Code deste ano, debaixo do projeto Fedora (de grande reconhecimento internacional, liderado pela RedHat). Então, eu fui mentor de um aluno [...] E depois que o programa acabou, a gente colocou a ferramenta na disciplina, a mesma que eu fiz no passado [...]*". Além disso, o participante P9 (que não tinha experiência prévia antes da disciplina) mencionou que se tornou contribuidor de projetos de software, incluindo distribuições Linux: "*atualmente, eu estou bem focado no Linux [...] Eu também contribuo com o Debian, só que não é implementação, é com empacotamento; eu mantenho dois pacotes lá, e estou preparando*

outros três".

Colaborar e contribuir com o desenvolvimento de software livre geram benefícios imediatos durante a disciplina, e consequências positivas depois dela. Através da experiência com software livre na disciplina, os entrevistados relatam oportunidades profissionais de alto nível, devido ao portfólio de colaboração em projetos de software livre, muitas vezes iniciado na disciplina. Outro benefício, bastante relevante colhido das entrevistas foi o relato da continuação nas contribuições após a disciplina (e ainda depois da conclusão do curso de graduação, por exemplo), uma vez que alguns dos entrevistados tornaram-se desenvolvedores oficiais de projetos de software profissionalmente. Com exemplo, um dos entrevistados colabora com *Kernel* do Linux e outro participou como mentor do *Google Summer of Code* por conta de uma ferramenta livre que ele mesmo desenvolveu.

4.1 Implicações

Nesta seção são apresentados implicações para os professores, os alunos e as comunidades de software livre, a partir dos achados deste estudo.

Para os professores de Engenharia de Software: dado a dinâmica entre professor, alunos e comunidades, os participantes mencionaram que o envolvimento do professor em atividades relacionadas aos projetos de software livre é essencial para a boa condução da disciplina. Em particular, P3 indicou que esse envolvimento "*era essencial, uma vez que o professor tinha mais experiência. Então, muitas das vezes em que os aluno desanimavam ou não sabiam muito bem o que fazer, o professor conseguia muito bem direcioná-los*". Ademais, o participante P5 indicou que, através de pesquisas acadêmicas conduzidas pelo professor, os alunos já sabiam que tipos de dificuldades eles enfrentariam:"*o professor estuda formas pra fazer com que as pessoas desistam menos de contribuir para projetos de software livre, e a gente ali teve que enfrentar essas dificuldades*". Além disso, P10 ressaltou o fato de que "*se o professor não está preparado, não tem como essa dinâmica ocorrer. Tem que ser uma pessoa que entenda e seja apaixonado pelo software livre.*". Por fim, P5 indicou também que além do engajamento com a comunidade de projetos de software livre, é também necessário "*engajar os alunos para ir pelo melhor caminho*". Isso significa que mais professores podem se motivar e fomentar a utilização de projetos de software livre em suas disciplinas, ao conhecerem melhor as dinâmicas de colaboração com as comunidades de software livre.

Para os alunos de Engenharia de Software: a experiência prévia com software livre não é requisito obrigatório. Nove dos 14 participantes entrevistados não tinham experiência prévia com projetos de software livre. E, de maneira geral, os participantes relataram um bom desempenho nas disciplinas. Ademais, além de linguagens como C, C++, Java, Python e Ruby, foram observados a utilização de linguagens de programação não comumente encontradas em currículos de curso de graduação em computação, como é o caso de Perl, Objective-C e

JavaScript. Esses pontos implicam na possibilidade de expandir o aprendizado para além de uma disciplina convencional de Engenharia de Software, também do ponto de vista da experiência com diferentes linguagens de programação.

Para os mantenedores de projetos de Software Livre: Uma vez que mais e mais professores introduzem esse tipo de metodologia em sala de aula, é esperado que outros alunos participem e se envolvam ainda mais com as comunidades. Mantenedores de projetos de software livre poderiam se inspirar em alguns dos achados deste trabalho e, por exemplo, propor *issues* que sejam propícias para serem resolvidas por alunos num curto período de tempo. Mantenedores poderiam também realçar na documentação e nos repositórios dos projetos quais os canais de comunicação (listas, IRC, entre outros) estão disponíveis para sanar dúvidas de alunos e facilitar o seu envolvimento no projeto, estando cientes que, enquanto estudantes, eles estão inseguros e com receios, não sabendo algumas vezes fazer a abordagem correta para interagir com os membros das comunidades. Nesse contexto, alguns entrevistados mencionaram algumas práticas que, se adotadas por comunidades de software livre, poderiam facilitar o processo de inserção de desenvolvedores externos às comunidades, em particular, alunos. Por exemplo, o participante P5 indicou que "*[...] tem a documentação do que o projeto faz, mas não tem uma documentação voltada para o desenvolvedor*". P13 também corroborou com a importância de boa documentação nos projetos: "*a gente estudou por que um projeto faz mais sucesso que outro, e geralmente isso se atribui a uma documentação muito bem feita*". Um outro ponto também mencionado por P5 é "*a ausência de label nas issues pra identificar se é uma tarefa para iniciante, intermediário, etc*". De maneira geral, esse participante foi além e mencionou que o projeto poderia, inclusive, ser dividido de forma que pessoas inexperientes possam começar contribuindo com partes para iniciantes.

5 CONSIDERAÇÕES FINAIS

Com o advento das plataformas de desenvolvimento colaborativo e codificação social, como o Github e o GitLab, vários projetos de software livre se tornaram mais acessíveis, ajudando na disseminação do processo colaborativo de software já usado pelas comunidades de software livre. Recentemente, professores de Engenharia de Software tem mostrado interesse em utilizar projetos de software livre como parte das atividades e da avaliação das suas disciplinas. Na pesquisa de Pinto et al (2017) foram discutidas algumas das percepções desses professores, que forneceram lições sobre a dinâmica da disciplina, bem como alguns possíveis benefícios e desafios encontrados ao colaborarem com projetos de software livre em suas disciplinas.

No entanto, pouco se sabe da percepção dos alunos que participaram de disciplinas com esse contexto. Neste trabalho, foram entrevistados 14 alunos que participaram de disciplinas que utilizavam contribuições em projetos de software livre como parte do processo de ensino-aprendizado. A partir das entrevistas foi possível compreender a percepção desses alunos sobre essa experiência. As questões de pesquisa foram respondidas com base na metodologia empregada. A utilização de projetos de software livre em disciplinas é algo promissor visto que possibilita ao aluno a vivência prática e aperfeiçoamento do seu portfólio. Em contrapartida, possuem desafios inerentes a duração da disciplina e a complexidade dos projetos. No entanto, vários alunos relataram uma satisfação em ter cursado as disciplinas, bem como avaliaram positivamente suas próprias participações nas mesmas.

5.1 Limitações

Os achados desse trabalho são limitados às 14 entrevistas realizadas com os alunos das cinco instituições em que os participantes cursaram as disciplinas investigadas. De forma relacionada, este trabalho se limita a estudar disciplinas de Engenharia de Software que utilizam contribuições em projetos de software livre como parte do processo de ensino-aprendizado. A utilização de contribuições em projetos de software livre em outras disciplinas — que pode levar a diferentes achados — está fora do escopo deste trabalho. Além disso, as disciplinas foram cursadas nos anos de 2012 a 2016, e as entrevistas foram conduzidas no 2º semestre de 2017, o que limita o estudo a memória dos partipantes. Porém, foi possível perceber os benefícios dos alunos a longo prazo como foi o caso do participante P9 que cursou uma disciplina com esta abordagem na graduação e mestrado, e também o participante P11 que si tornou mantenedor de um projeto de software livre. Por fim, a condução do estudo foi primariamente baseado nas entrevistas (embora enriquecido por uma análise dos dados das contribuições feitas nos projetos de software livre), dessa forma, está limitado ao entendimento subjetivo da análise das entrevistas.

5.2 Trabalhos Futuros

Como trabalhos futuros, ampliar o escopo do trabalho realizando mais entrevistas com outros alunos, e realizar um estudo de larga escala em repositórios de software livre de forma a identificar e categorizar contribuições feitas por alunos em contextos de disciplinas. Por fim, conduzir entrevistas com mantenedores e líderes de projetos de software livre de forma a triangular os achados deste estudo, bem como com resultados de estudos com análise da perspectiva do professor, de forma a confirmar ou refutar resultados conhecidos na literatura.

REFERÊNCIAS

BUCHTA, J. et al. Teaching evolution of open-source projects in software engineering courses. In: **22nd IEEE International Conference on Software Maintenance**. [S.l.: s.n.], 2006. p. 136–144.

CAWLEY, O. et al. Incorporating a self-directed learning pedagogy in the computing classroom: Problem-based learning as a means to improving software engineering learning outcomes. In: ____. [S.l.]: IGI Global, 2014. p. 348–371.

CHARMAZ, K. **A Construção da Teoria Fundamentada - Guia Prático para Análise Qualitativa**. 1ª edição. ed. [S.l.]: Artmed, 2009. ISBN 978-85-36319995.

CHAVEZ, C. et al. Free/libre/open source software development in software engineering education: Opportunities and experiences. In: **4th Software Engineering Education Forum, FEES@SBES@CBSoft, São Paulo, Brazil**. [S.l.: s.n.], 2011.

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

GOUSIOS, G.; PINZGER, M.; DEURSEN, A. van. An exploratory study of the pull-based software development model. In: **36th International Conference on Software Engineering, ICSE '14, Hyderabad, India - May 31 - June 07, 2014**. [S.l.: s.n.], 2014. p. 345–355.

GOUSIOS, G. et al. Work practices and challenges in pull-based development: The integrator's perspective. In: **37th IEEE/ACM International Conference on Software Engineering, ICSE 2015, Florence, Italy, May 16-24, 2015, Volume 1**. [S.l.: s.n.], 2015. p. 358–368.

HATTORI, L. P.; LANZA, M. On the nature of commits. In: IEEE PRESS. **23rd IEEE/ACM International Conference on Automated Software Engineering**. [S.l.], 2008. p. III–63.

HISLOP, G. W. et al. Using open source software to engage students in computer science education. In: **40th ACM Technical Symposium on Computer Science Education**. [S.l.: s.n.], 2009. p. 134–135.

HOLMES, R.; ALLEN, M.; CRAIG, M. Dimensions of experientialism for software engineering education. In: **Proceedings of the 40th International Conference on Software Engineering: Software Engineering Education and Training**. New York, NY, USA: ACM, 2018. (ICSE-SEET '18), p. 31–39. ISBN 978-1-4503-5660-2.

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

MORGAN, B.; JENSEN, C. Lessons learned from teaching open source software development. In: **10th International Conference on Open Source Systems, OSS 2014, San José, Costa Rica, May 6-9, 2014.** Berlin, Heidelberg: [s.n.], 2014. p. 133–142.

NASCIMENTO, D. M. et al. Using open source projects in software engineering education: A systematic mapping study. In: **2013 IEEE Frontiers in Education Conference (FIE)**. [S.l.: s.n.], 2013. p. 1837–1843.

PETRENKO, M. et al. Teaching software evolution in open source. **Computer**, v. 40, n. 11, p. 25–31, nov. 2007.

PINTO, G.; DIAS, L. F.; STEINMACHER., I. Who gets a patch accepted first? comparing the contributions of employees and volunteers. In: **11th IEEE/ACM International Workshop on Cooperative and Human Aspects of Software Engineering, CHASE@ICSE 2018, Gothenburg, Sweden, May, 2018**. [S.l.: s.n.], 2018.

PINTO, G. et al. Training software engineers using open-source software: The professors' perspective. In: **30th Conference on Software Engineering Education and Training (CSEET)**. [S.l.: s.n.], 2017. p. 117–121.

SARMA, A. et al. Training the Future Workforce Through Task Curation in an OSS Ecosystem. In: **2016 24th ACM FSE**. [S.l.]: ACM, 2016. p. 932–935.

SEDELMAIER, Y.; LANDES, D. A research agenda for identifying and developing required competencies in software engineering. **International Journal of Engineering Pedagogy**, v. 3, n. 2, 2013.

SMITH, T. M. et al. Selecting open source software projects to teach software engineering. In: **45th ACM Technical Symposium on Computer Science Education**. [S.l.: s.n.], 2014. p. 397–402.

APÊNDICE A – ROTEIRO DA ENTREVISTA

- **Perfil do Entrevistado**

  1. Qual é o seu nome?
  2. Qual é a sua idade?
  3. Qual é o seu gênero?
  4. Qual é a sua escolaridade?
  5. Qual a sua ocupação atual?
  6. Como você avalia o seu desempenho nas disciplinas relacionadas a desenvolvimento de software?
  7. Você possui quanto tempo de experiência com desenvolvimento de software?
  8. Você possui algum conhecimento adicional em desenvolvimento de software ? Por exemplo: Certificações, treinamentos, etc.

- **Experiência anterior com software livre**

  9. Você já havia feito algum tipo de contribuição em algum projeto de software livre antes desta disciplina?
     a) Caso sim, quantos e quais foram? Quais foram as dificuldades que você enfrentou para contribuir?
     b) Caso não, porque não contribuiu?

- **Uso de Software livre na disciplina**

  10. Qual a instituição, e o curso em que foi ministrada a disciplina?
  11. Qual o nome da disciplina? Em qual semestre ou período foi ministrada?
  12. Qual era o objetivo com o uso de SL?
  13. Como foi feita a escolha do projeto de SL para o qual contribuíram?
  14. Qual era o nome do projeto de SL?
      a) Qual a linguagem de programação utilizada? Você conhecia a linguagem? Teve dificuldades, quais?
  15. Qual foi a tarefa (ou as tarefas) passadas para vocês?
      a) Como foi definida a tarefa que vocês deveriam cumprir? Quem definiu?
  16. Você detinha um conhecimento prévio necessário para a execução das tarefas?
      a) Quais são esses conhecimentos?
      b) Caso não, Como você poderia adquiri-los?

17. Como era sua interação com a comunidade do projeto de SL? Encontrou alguma dificuldade, quais foram?
18. Durante a disciplina, quais foram as dificuldades que você enfrentou para contribuir em projetos de software livre?
    a) Caso tenha tido dificuldades, como você conseguiu saná-las ao longo da disciplina?
    b) Teve alguma dificuldade que considera mais importante ou que tenha influenciado mais negativamente?
19. Você conseguiu fazer alguma contribuição para o projeto de SL ao longo da disciplina?
    a) Caso sim, conte-me quantas e quais foram?
        i. Ter uma contribuição aceita o motivou a querer contribuir mais? Se sim, porquê ?
    b) Caso não, Você poderia descrever quais foram os motivos?
20. Na disciplina teve algum integrante do projeto de SL que ficou responsável em auxiliar vocês na execução das tarefas?
    a) Como essa pessoa foi escolhida/indicada?
    b) Ocorreu alguma situação, em que o apoio foi necessário? Você poderia relatar?
21. Além do professor da disciplina, outra pessoa da instituição auxiliou vocês nas contribuições?
    a) Caso sim, em que situações ocorriam esse auxílio? Você poderia relatar?
22. Que tipo de conhecimento que você tinha obtido na teoria foi possível pôr em prática contribuindo para OSS?
23. Quais benefícios reais você sentiu contribuindo para OSS?
24. Você continua participando em projetos de software livre?
    a) Caso sim, a disciplina motivou você a continuar contribuindo?
    b) Caso não, o que fez você não contribuir mais?

- **Específicas sobre a disciplina**

25. Como eram as aulas relacionadas à contribuição? Você poderia me contar em detalhes como era um dia de aula?
26. Como eram feitas as avaliações da disciplina?
27. Como você avalia o seu desempenho na disciplina?
28. Como você avalia a participação do professor da disciplina?

29. Como você avalia a duração da disciplina? Foi suficiente para cumprir as tarefas e realizar as contribuições?
30. Como você compara uma disciplina como essa, como outras disciplinas tradicionais?
    a) O que é melhor?
    b) O que é pior?
31. Se você pudesse sugerir algo relacionado ao uso de SL em disciplinas de graduação/pós-graduação, o que seria?

- **Encerramento**

32. Há alguma informação que eu não perguntei, mas você acha interessante comentar sobre a sua experiência nesta disciplina?
33. Você poderia compartilhar o link do repositório com suas contribuições no projeto de software livre?