Universidade de São Paulo
Instituto de Matemática e Estatística
Departamento de Ciência da Computação

–

Proposta MAC0499
Trabalho de Formatura Supervisionado

–

O que observar enquanto um desenvolvedor contribui com o Kernel Linux?

Rubens Gomes Neto
Orientador: Prof. Dr. Paulo Meirelles[1]
Coorientador: M.Sc. Rodrigo Siqueira[2]
rubens.gomes.neto@usp.br, paulormm@ime.usp.br, rodrigosiqueiramelo@gmail.com

[1]Docente da UFABC e pesquisador-colaborador da USP.
[2]AMD (Advanced Micro Devices, Inc.)

1. Introdução

O processo de desenvolvimento de software vem evoluindo constantemente ao longo dos anos, passando por formas de trabalhar menos flexíveis como "modelo cascata" até a era dos métodos ágeis. Com essa evolução, também temos o desenvolvimento globalmente distribuído e a integração de desenvolvedores externos aos ecossistemas de software (Fagerholm and Münch, 2012) , em especial como ocorre nos projetos de software livre. Tais abordagens de colaboração em prol do desenvolvimento de software exigem um maior entendimento das percepções dos desenvolvedores em relação às suas atividades no projeto . Nesse contexto, na engenharia de software, estudamos a experiência do desenvolvedor que por sua vez pode ser entendida como um meio de capturar como os programadores pensam e interagem durante suas atividades em seus respectivos ambientes de trabalho (Fagerholm and Münch, 2012) . Em projetos como do kernel Linux, com mais de 30 anos de desenvolvimento, que cresce e torna-se cada vez mais complexo, o conhecimento precisa ser documentado e o processo precisa ser automatizado visando reduzir barreiras .

Software também diz respeito à aspectos que ultrapassam as questões técnicas, como (i) o processo de desenvolvimento do software; (ii) os mecanismos econômicos (gerenciais, competitivos, sociais, cognitivos etc.) que regem esse desenvolvimento e seu uso; (iii) o relacionamento entre desenvolvedores, fornecedores e usuários do software; (iv) os aspectos éticos e legais relacionados ao software (Kon et al., 2011). Assim, o que define e diferencia o software livre do que podemos denominar de software proprietário passa pelo entendimento desses quatro pontos dentro do conceito de *ecossistema do software livre* (Meirelles, 2013). O princípio básico desse ecossistema é promover a liberdade do usuário, sem discriminar quem tem permissão para usar um software e seus limites de uso, baseado na colaboração e num processo de desenvolvimento aberto (Kon et al., 2011).

Software livre é aquele que permite aos usuários usá-lo, estudá-lo, modificá-lo e redistribuí-lo, em geral, sem restrições para tal e prevenindo que não sejam impostas restrições aos futuros usuários. Normalmente, esse software existe por meio de projetos de desenvolvimento que estão centrados em torno de algum código-fonte acessível ao público, geralmente em um repositório na Internet, onde desenvolvedores e usuários podem interagir de maneira globalmente distribuída (Meirelles, 2013). O código é necessariamente licenciado sob termos legais formais que estão de acordo com as definições da *Free Software Foundation* [3] ou da *Open Source Initiative* [4].

[3] http://www.gnu.org/philosophy/free-sw.html
[4] http://www.opensource.org/docs/definition.html

O desenvolvimento de projetos de software livre e a dinâmica de suas comunidades é tema de diversas publicações, entre as mais famosas está a de Raymond (1999). Onde ele definiu dois modelos distintos de desenvolvimento com base em suas observações e experiência em projetos de software livre, ou seja, modelos empíricos de desenvolvimento de software: o modelo Catedral e o modelo Bazar (Raymond, 1999). O Catedral se aproxima da visão comercial, em que as partes críticas de um software devem ser construídas apenas por um grupo especial de pessoas. Por outro lado, o modelo Bazar, baseado, segundo Raymond, no mundo Linux, segue, entre outras premissas, a lei de Linus: "Havendo olhos suficientes, todos os erros são evidentes." Para isso, a medida que algo é desenvolvido, este deve ser liberado o quanto antes para apreciação de todos os interessados no projeto. A partir desse ensaio, o desenvolvimento de software livre passou a ser referenciado como um fenômeno homogêneo e idealizado (Wen, 2019).

Um dos aspectos do desenvolvimento do software livre citado e muito elogiado por Raymond é o caráter voluntário de seus contribuidores. Para ele, os contribuidores eram nada mais que usuários especialistas do sistema que, ao encontrar bugs, realizavam esforços para reportá-los e até consertá-los por conta própria. Embora que durante muitos anos a tese de Raymond tenha sido muito referenciada pela comunidade de engenharia de software, alguns estudos começaram a questionar alguns pontos da dicotomia "Catedral e Bazar" (Wen, 2019). Uma das discordâncias é com relação ao caráter homogêneo que o desenvolvimento de software livre tem sido retratado (Osterlie and Jaccheri, 2007). Assim, estudar e compreender a dinâmica atual das comunidades de software livre, em especial do kernel Linux, mostra-se fundamental para o avanço das técnicas de desenvolvimento de sistemas. Nesse sentido, estamos propondo investigar a experiência do desenvolvedor ao contribuir com kernel Linux.

2. Objetivos

Ecossistemas de software livre desenvolvem maneiras de equilibrar a carga de trabalho dos contribuidores, desde os desenvolvedores periféricos aos mantenedores (Zhou et al., 2017). À medida que os ecossistemas crescem, novos mecanismos de colaboração precisarão evoluir. Por isso, o objetivo deste projeto é investigar a experiência do desenvolvedor ao contribuir com o kernel Linux. Iremos propor um arcabouço de métricas (para coletar dados estatísticos) para capturar como os desenvolvedores interagem durante suas atividades em seus ambientes de desenvolvimento.

Essas métricas serão implementadas, criando um módulo de estatísticas, em uma ferra-

menta que ajuda a automatizar o fluxo de trabalho dos desenvolvedores do kernel Linux, chamada KWorkflow (kw[5]). Ela visa reduzir a sobrecarga para configurar o ambiente de trabalho para o desenvolvimento do kernel Linux e fornece ferramentas de apoio para tarefas diárias. Nesse sentido, o kw proporciona um ambiente que automatiza a execução de diversas tarefas repetitivas e enfadonhas para a maioria dos desenvolvedores, além de reduzir a chance de erros ocasionados por etapas manuais.

Queremos evoluir o kw para ter um módulo de estatísticas para fornecer aos usuários dados relacionados as tarefas diárias de desenvolvimento. Por exemplo, um usuário é capaz de ver quantas vezes ele usou o kw para compilar e instalar um kernel, além de informações de quanto tempo demorou e quantas vezes esses comandos falharam. Todas essas informações serão armazenadas localmente, separadas em uma estrutura que facilite o processamento dos dados. Também é interessante que entre as opções de estatísticas do kw, os usuários possam ver quais opções do kw eles mais usam, ou mesmo quais são os seus dias e horas mais ativos usando a ferramenta.

Do ponto de vista da pesquisa e coleta de dados, os usuários poderão opcionalmente enviar suas estatísticas para o servidor do kw (desabilitando esta opção sempre que quisessem) gerando um painel colaborativo para a comunidade observar tais dados. Para oferecer suporte à proteção da privacidade, todos os dados armazenados serão anonimizados. Uma maneira de fazer isso é como o pacote Debian Popularity-contest[6]: os dados são enviados para um e-mail, tornados anônimos assim que chegam ao servidor, analisados e então publicados em seu website. O kw poderá funcionar de maneira semelhante, tendo um site simples para exibir essas informações relacionadas ao fluxo de trabalho de desenvolvimento do kernel.

3. Metodologia

A literatura de engenharia de software discute pouco sobre o que é realizado pelos profissionais da área (Osterlie and Jaccheri, 2007), ou seja, quais atividades eles realizam e com que frequência acontecem. Para preencher a lacuna entre prática e teoria, devemos adotar a abordagem da indústria como laboratório (Wen, 2019), no nosso caso, uma comunidade de software livre como laboratório. Dessa forma, conduziremos um estudo de caso para observar a experiência dos desenvolvedores durante a execução das diferentes tarefas para contribuir com um dos subsistemas do kernel Linux.

[5] https://github.com/kworkflow/kworkflow
[6] https://popcon.debian.org

A primeira etapa consiste no estudo exploratório sobre a experiência do desenvolvedor e o processo de desenvolvimento do kernel Linux. Numa segunda etapa, depois de definir e implementar o conjunto de métricas do módulo de estatísticas da ferramenta kw, usaremos métodos de pesquisa qualitativa para validarmos o nosso arcabouço de coleta de dados. De acordo com Merriam (2009), "pesquisa qualitativa tenta entender e dar sentido aos fenômenos a partir da perspectiva do participante. O pesquisador pode abordar o fenômeno a partir de uma postura interpretativa, crítica ou pós-moderna". Para isso, vamos usar o estudo de caso, que é uma investigação empírica para analisar um fenômeno em seu contexto real (Yin, 2009).

O kw está sendo usado, especialmente, pela equipe do engenheiro de software da empresa AMD, Rodrigo Siqueira, criador da ferramenta kw e ex-aluno do professor Paulo Meirelles, orientadores deste trabalho. Um grupo de alunos e ex-alunos da USP, que fazem parte do grupo FLUSP (FLOSS at USP)[7], também são usuários ativos do kw. Nosso plano é conduzir um estudo de caso observando, em momentos diferentes, pelo menos 10 desenvolvedores do kernel Linux. De acordo com as atividades de desenvolvimento dos contribuidores que participarão do estudo de caso, o principal subsistema a ser estudado é o *Direct Rendering Manager* (DRM).

Esta pesquisa, para a condução do estudo de caso planejado, será submetida ao comitê de ética em pesquisa do IME-USP (ou ao comitê de outro instituto da USP), via a Plataforma Brasil[8].

4. Resultados esperados

Este projeto está associado às pesquisas que realizamos no Centro de Competência em Software Livre (CCSL) do IME-USP. Atualmente, esta frente de pesquisa que estuda ecossistemas de software livre conta com três mestrandos: Melissa Wen, Eduardo Pinheiro e Marcelo Schmitt. Minhas atividades neste projeto irão auxiliar na condução de oportunidades de pesquisa já identificadas nessas pesquisas de mestrado, além de colaborar mais diretamente com os avanços dos estudos do mestrando Eduardo Pinheiro (sobre os modelos de manutenção do kernel Linux).

Do ponto de vista tecnológico, este projeto irá gerar algumas contribuições importantes para a evolução da ferramenta kw, impactando diretamente nas atividades de dezenas de contribuidores do kernel Linux no curto prazo. Em outras palavras, como resultados espe-

[7] https://flusp.ime.usp.br
[8] https://plataformabrasil.saude.gov.br

rados, vamos contribuir com a automação do fluxo de trabalho, consolidar o conhecimento da gestão de trabalho do kernel Linux e melhorar a questão da carga de trabalho. Dentre as consequências diretas dessa automação, esperamos observar melhorias diretas para os desenvolvedores experientes, assim como facilitar o processo de inserção de novos programadores. Por fim, essa ferramenta possibilitará a coleta de dados de maneira a permitir uma análise próxima dos desenvolvedores.

Posteriormente, a disseminação dos resultados obtidos na condução deste projeto poderão ocorrer via veículos científicos (anais de eventos científicos e periódicos) de seletiva política editorial. Especificamente, para as publicações científicas, planejo participar da escrita de um artigo para o *Journal of Open Source Software* (JOSS) e para o *International Symposium on Open Collaboration* (OpenSym).

5. Cronograma

Na Tabela 1, detalhamos, por trimestre, as atividades que serão realizadas durante os 9 meses deste projeto de pesquisa, de acordo com a metodologia descrita e objetivos supracitados.

Tabela 1: Cronograma proposto para o TCC por 9 meses

Atividade	T1	T2	T3
Introdução ao desenvolvimento do kw	•		
Estudos exploratórios	•	•	
Proposta de arbouço de métricas		•	
Implementação do módulo de estatística do kw		•	•
Execução e análise do estudo de caso			•

Referências

Fagerholm, F. and Münch, J. (2012). Developer experience: Concept and definition. In *2012 International Conference on Software and System Process (ICSSP)*, pages 73–77.

Kon, F., Lago, N., Meirelles, P., and Sabino, V. (2011). *Atualizações em Informática (SBC)*, chapter Software Livre e Propriedade Intelectual: Aspectos Jurídicos, Licenças e Modelos de Negócio, pages 59–107. PUC-Rio, Rio de Janeiro.

Meirelles, P. R. M. (2013). *Monitoramento de métricas de código-fonte em projetos de software livre*. PhD thesis, Instituto de Matemática e Estatística, Universidade de São Paulo.

Merriam, S. (2009). *Qualitative Research: A Guide to Design and Implementation*. Jossey-Bass higher and adult education serie. John Wiley & Sons.

Osterlie, T. and Jaccheri, M. L. (2007). A critical review of software engineering research on open source software development.

Raymond, E. S. (1999). *The Cathedral & the Bazaar*. O'Reilly & Associates, Inc., Sebastopol, CA, USA.

Wen, M. (2019). What happens when the bazaar grows: A comprehensive study on the transformations of the floss development model.

Yin, R. K. (2009). *Case Study Research: Design and Methods*. SAGE Publications.

Zhou, M., Chen, Q., Mockus, A., and Wu, F. (2017). On the scalability of linux kernel maintainers' work. In *Proceedings of the 2017 11th Joint Meeting on Foundations of Software Engineering*, ESEC/FSE 2017, page 27–37, New York, NY, USA. Association for Computing Machinery.