Relatório da Entrega 3 de Estudos Avançados em Sistemas de Software (24/04/2025)

Nome: David Tadokoro
NUSP: 10300507
Orientador: Paulo Meirelles

1) Contexto

No primeiro semestre de 2024, houve o oferecimento da disciplina de *Desenvolvimento em Software Livre* (MAC0470/5856). Grande parte da disciplina envolveu mentorar os alunos para contribuírem para algum subsistema do kernel Linux (o subsistema do *Industrial I/O*, no caso), partindo desde a construção de um ambiente de testes até o envio do patch e participação no processo de revisão. Neste primeiro semestre de 2025, a disciplina está sendo ofertada novamente nos mesmos moldes.

O projeto proposto para a disciplina envolve a escrita de um artigo científico sobre o fenômeno de entrada de novatos no kernel Linux, e como mentorá-los de uma forma eficiente e prática.

Nesta terceira entrega, o objetivo era fazer o abstract e o capítulo de introdução do paper (pelo menos uma versão original).

Pessoalmente, eu (David) optei por começar com estas duas partes do artigo (mesmo com a introdução sendo geralmente feita ao final), pois acredito que a síntese do contexto, problema, resultados e discussão estão bem madura no meu entendimento.

Tanto o arquivo fonte em Markdown (`entrega-3/relatorio.md`), quanto os arquivos em texto puro do abstract e da introdução (`entrega-3/{abstract,intro}.txt`) se encontram no **meu repositório de artefatos para a disciplina**.

Temos um **documento Latex no overleaf** dedicado para o artigo, mas, no momento, ele está como um "dump" dos textos, gráficos, etc.. Pedro, se quiser olhar o documento, me passe seu email do overleaf que nós te liberamos. De toda forma, o abstract e a introdução estão a seguir na íntegra para consulta fácil.

2) Abstract

Software development is a complex task that virtually always involves developers collaborating. In this sense, many Free Libre and Open Source Software (FLOSS) development models have successfully created large, scalable, and (more often than not) globally distributed communities that evolve and maintain high-quality software projects, with the Linux kernel ecosystem being a prime example. Beyond the technical challenges natural to the project, the entry barrier is

steep and discouraging for newcomers to the Linux ecosystem as there are many sub-projects (called subsystems), each with its specific contributing rules, processes, and practices that are usually undocumented and only learned by direct contact with the community; this risks the project long-term sustainability, as the renewal of the highly qualified workforce is a known problem. This work aims to validate an approach to mentor newcomers to the Linux ecosystem efficiently. The research questions that guided us are: (1) What teaching techniques are effective in introducing newcomers to the Linux ecosystem (in-loco workshops, tutorials, lectures from practitioners)? (2) What hard and soft skills are crucial to successfully entering the Linux ecosystem? (3) What processes and practices (workflows) should the mentors abstract for a smoother entry into the Linux ecosystem? In the first semesters of 2024 and 2025, the Free Software Development course was ministered at the University of São Paulo (USP) by the authors - one as the professor and the others as teaching assistants (TAs). During these offerings, students went from setting up a testing environment to learning the fundamental workflows involved, culminating in sending patches and interacting with Linux communities through the code review process. These students, who in the majority had no experience in Linux development, were closely mentored using a combination of teaching techniques, and their experience and feedback were collected through surveys and blog posts written by them. Among our findings, we can highlight: (1) Use of directed content (tutorials) produced by real practitioners in the Linux ecosystem, along with in-person mentoring during classes (workshops) and accessibility of the professor and TAs, produces a fertile environment for newcomers; (2) The experience in the course enhanced the qualification of students, from hard skills like git, and email and web-based models of code collaboration to communication skills; (3) The experience in the course demystified a lot of inaccurate concepts from the students about FLOSS development and made them more comfortable and ready to contribute to other FLOSS projects.

3) Capítulo de introdução do artigo

The development of modern software systems is a highly collaborative endeavor, often demanding the coordination of multiple individuals with diverse skill sets, experiences, backgrounds, and - more often than not - globally distributed. Free Libre and Open Source Software (FLOSS) development models employed by many successful software projects pervasive in our society have raised the attention of industry and academia to leverage its benefits and get insights into fields like Software Engineering (SE).

In these projects that employ FLOSS development models, contributions frequently originate from a distributed and (sometimes) volunteer-based workforce called contributors. At the same time, the approval, feedback, and decision-making of the changes are the responsibility of a smaller group with administrative privileges of the project. FLOSS development models have historically demonstrated a unique ability to sustain long-lasting, high-quality software, even

under decentralized organization and asynchronous collaboration constraints. The prime example of this success is the Linux kernel project, which has grown into one of the largest and most influential FLOSS initiatives in history. When we talk about the Linux project, we are talking about the umbrella project composed of many sub-projects (called subsystems) that form a complex and interconnected ecosystem.

Even though the Linux ecosystem is often celebrated for its technical excellence and community-driven development, some challenges risk the project's long-term sustainability. In particular, we want to focus on the problem of the steep entry barrier to the Linux ecosystem for newcomers, which can be intimidating, if not prohibitive, in some cases. Prospective contributors must navigate an enormous codebase fragmented in many development contexts and dedicated code repositories. These comprise the numerous subsystems governed by distinct and often undocumented conventions, processes and practices (workflows), and social norms. These characteristics complicate the onboarding process and pose a problem for renewing the highly specialized workforce. Without a steady influx of new and adequately trained contributors, the vitality and evolution of the ecosystem could be jeopardized.

Addressing this issue requires more than technical tutorials or improved documentation. It demands a pedagogical approach that recognizes the interplay of technical proficiency, social integration, and community practices. In this context, mentorship is necessary to keep newcomers motivated and smooth their entry into the ecosystem. Effective mentorship can help demystify the contribution process, transmit tacit knowledge, and build confidence in potential contributors.

This paper presents and evaluates a structured approach to mentoring newcomers to the Linux kernel ecosystem. Our research is grounded in practical experience teaching the Free Software Development course at the University of São Paulo (USP) during the first semesters of 2024 and 2025. The course, taught by the authors - one as the professor and the others as teaching assistants (TAs) - offered students a hands-on experience in FLOSS contribution focused on the Linux project. The students learned to set up a testing environment; configure, compile, and install custom-built kernels and modules; assess contribution opportunities; develop and send contributions to maintainers and mailing lists; and participate in the review process by interacting with the suggestions and requests of the community to refine initially proposed changes to the standard the project demands.

Through classroom workshops, curated tutorials authored by veteran Linux practitioners, and sustained direct mentoring (by the professor and TAs), students (in the majority with no experience with Linux or FLOSS development in general) went from learning the basics to (in most cases) having a merged contribution, within a couple of months. Qualitative data from students was collected via surveys and through analyzing blog posts to examine their learning trajectories and perceptions. Observations from the professor and TAs were also

systematically compiled to enrich the research.

The findings from this work indicate many interesting perspectives, which we highlight:

1. In-loco workshops, learning materials produced by veteran Linux practitioners, and accessible mentors can significantly lower the entry barrier to FLOSS projects like the Linux kernel;
2. The students, independent of their background, reported that they enhanced their technical skills related to software development (deeper proficiency in git, device drivers, and C programming language) while exercising their communication skills, which is paramount for successful collaborative development;
3. The students who had detached and imprecise perceptions of Linux and FLOSS development, which was the majority, reported that the mentoring experience helped them demystify those concepts and that they felt more empowered and comfortable to contribute to other FLOSS projects.

These contributions help to have a more comprehensive view of the movement of new contributors to the Linux ecosystem and provide an approach to successfully mentor them that can be applied to FLOSS projects in general.

4) Conclusão

Ambas as partes, principalmente a introdução, certamente devem sofrer alguns ajustes conforme o texto for sendo escrito e os dados do oferecimento de 2025 forem sendo gerados/analisados. No entanto, consideramos que a direção do artigo irá seguir nesta linha.