Fostering Free/Libre Open Source Software community formation: guidelines for communities to support newcomers' onboarding

Igor Steinmacher[1] and Marco Aurélio Gerosa[2]

[1]Departamento de Computação – Universidade Tecnológica Federal do Paraná (UTFPR)
Campo Mourão – PR

[2]Instituto de Matemática e Estatística – Universidade de São Paulo (USP)
São Paulo – SP

Abstract. *Community-based Free/Libre Open Source Software (FLOSS) projects are usually self-organized and dynamic, receiving contributions from distributed volunteers. These communities' survival, long-term success, and continuity demand a constant influx of newcomers. However, newcomers face many barriers when making their first contribution to an OSS project, leading in many cases to dropouts. Therefore, a major challenge for OSS projects is to provide ways to support newcomers during their first contribution. In this paper, our goal was to provide a set of guidelines to communities that want to support and foster the participation of newcomers. The guidelines were proposed based on the analysis of data collected from: semi-structured interviews with 36 developers from 14 different projects; 24 answers to an open questionnaire conducted with OSS developers; feedback from 9 graduate and undergraduate students after they tried to join OSS projects; and 20 primary studies gathered via a systematic literature review.*

1. Introduction

Community-based Free/Libre Open Source Software (FLOSS) projects are generally self-organized and dynamic, receiving contributions from volunteers spread across the globe. These communities' survival, long-term success, and continuity demand a constant influx of newcomers [Park and Jensen 2009]. Thus, to build and keep a sustainable community it is essential to motivate, engage, and retain developers [Qureshi and Fang 2011].

Joining a FLOSS project is a complex process composed of different stages and a set of forces that push newcomers towards or away from the project [Steinmacher and Gerosa and et al. 2014]. The joining process can be summarized into two different stages: onboarding and contributing. While onboarding stage is highly impacted by a steep learning curve as well as reception and expectation breakdowns, longer-term forces influence the contributing stage. The process also shows the importance of what comes before onboarding. We characterize the forces that draw outsiders to a project, such as

motivation and project attractiveness. While motivation persists as an ongoing force, various hindering factors and retention forces influence onboarding, contribution, and members' permanence. A model that represents the stages and forces that influence the joining process of developers is presented in Figure 1. This model is composed of the stages that are common to and the forces that are influential to the joining process.

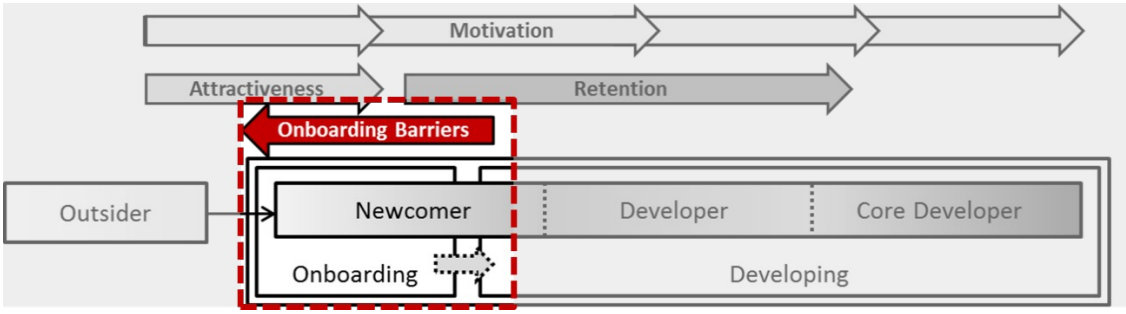

Figure 1. Developers joining model presenting the stages and forces that act during the joining process

The central elements of the model are the stages that developers go though and for which FLOSS communities should have different investments in terms of having more developers contributing to the project. An *outsider* represents a potential contributor to the project who is not involved with the development so far. A *newcomer* is a developer trying to place their first code contributions into the project. A *contributor* represents a developer that participates in the project, but who is not recognized as a member and does not have commit privileges. A *member* is someone recognized by the community as a developer or formal contributor.

We represent in the model four different forces that influence the progress from one stage to the following. Motivation and project attractiveness are the triggers to push the outsider to contribute to the project. *Motivation* forces represent internal (e.g., learning, self-marketing, recognition) and external (e.g., scholarship, course assignment, feature need) motives that drive a developer to join (and keep contributing) to a project. Hence, motivation forces are presented in the whole joining model, since lack of motivation lead to drop offs. The motivation forces can change/evolve during the development process. For example, some developers onboard a project because of a short-term scholarship, such the one given by Google Summer of Code, or a grade in a university course, and after that, they remain contributing to learn and self-promote themselves.

Attractiveness forces represent the characteristics and actions that the project presents to bring new users and developers. These forces can include type of license, project visibility, project age, number of developers, etc.

Attractiveness and motivation work together to push outsiders toward the projects. In some cases, attractiveness forces play a special role, pulling motivated developers that did not decide which project to support.

Retention forces may help to push newcomers to stay willing to contribute. *Retention* forces represent the characteristics and actions that a project presents to bring/keep more

contributors to the project. Some of these forces are initiatives to support newcomers overcoming the barriers (such as providing tools to facilitate code understanding, or indicating good tasks or pieces of code to start with). Other forces represent mechanisms to support existent contributors to contribute more, triggering, in some cases, motivation change (e.g., granting commit rights to a developer, using *gamification* elements).

The period after newcomers decide to participate and before they make their first code contribution represents the transition from outsider to newcomer. This period is crucial to community formation, since it represents the first contact of the newcomers with the community. It occurs when a developer decides to contribute to a project. At this point, developers attempt to contribute to the project, and they start onboarding to the project. During this period, motivation keeps pushing the developer towards the project. However, some opposite forces, which we call onboarding barriers, hamper developers joining process. These forces comprise technical and non-technical barriers, including learning curve, lack of support from the community, difficulties finding how to start, etc.

Understanding how to deal with these barriers is critical to the community joining. These barriers can be powerful enough to lead developers to give up contributing to the project. An important thing to observe is that these barriers influence both developers willing to make a single contribution and those willing to climb higher and become a member of the project.

In this paper we present guidelines to communities that want to better receive newcomers, based on a set of barriers empirically evidenced on previous studies [Steinmacher and Chaves and et al. 2014; Steinmacher and Wiese and et al. 2014; Steinmacher et al. 2015].

2. Related Work

Newcomers' onboarding is not an issue exclusively faced by FLOSS. Many studies in the literature deal with newcomers joining process in collective production communities, including studies on Wikipedia [Halfaker et al. 2011; Vora et al. 2010] and on FLOSS projects [Canfora et al. 2012; Jensen et al. 2011; Malheiros et al. 2012; Von Krogh et al. 2003; Wang and Sarma 2011]. Dagenais et al. [Dagenais et al. 2010] and Begel and Simon [Begel and Simon 2008] present studies regarding newcomers joining process in software projects, but their focus is in industrial settings.

Von Krogh et al. [Von Krogh et al. 2003] analyzed interviews with developers, emails, source code repository, and documents of the FreeNet project. The authors proposed a joining script for developers who want to take part in the project. Nakakoji et al. [Nakakoji et al. 2002] studied four FLOSS projects to analyze the evolution of their communities. They presented eight possible roles for the community members and structured them into a model composed of concentric layers, like the layers of an onion. Although these papers deal with the evolution of members' participation in FLOSS communities, they focus on newcomers after the onboarding.

Some researchers tried to understand the barriers that influence the retention of newcomers. Zhou and Mockus [Zhou and Mockus 2012] worked on identifying the newcomers who are more likely to remain in the project in order to offer active support for them to become long-term contributors. Jensen et al. [Jensen et al. 2011] analyzed mailing lists of FLOSS projects to verify if the emails sent by newcomers are quickly answered, if gender and nationality influence the kind of answer received, and if the reception of newcomers is different in users and developers lists. Steinmacher et al. [Steinmacher et al. 2013] used data from mailing list and issue tracker to study how reception influences the retention of newcomers in an FLOSS project.

There are also some studies presenting tools to support newcomers' first steps. Čubranić et al. [Cubranic et al. 2005] presented Hipikat, a tool that supports newcomers by building a group memory and recommending source code, mails messages, and bug reports to support newcomers. Wang and Sarma [Wang and Sarma 2011] present a Tesseract extension to enable newcomers to identify bugs of interest, resources related to that bug, and visually explore the appropriate socio-technical dependencies for a bug in an interactive manner. Park and Jensen [Park and Jensen 2009] show that visualization tools support the first steps of newcomers in an FLOSS project, helping them to find information more quickly.

Mentoring is also explored as a way to support newcomers. Malheiros et al. [Malheiros et al. 2012] and Canfora et al. [Canfora et al. 2012] proposed different approaches to identify and recommend mentors to newcomers of FLOSS projects by mining data from mailing lists and source code versioning systems.

As listed, there are some efforts to study newcomers to FLOSS. However, we could not find any study focused on identifying and organizing the barriers faced by newcomers to FLOSS. In previous work, we report some preliminary results of this research. In [Steinmacher and Silva and et al. 2014] we report the results of the systematic literature review, which is part of the current study, and in [Steinmacher and Wiese and et al. 2014] we report the results of the analysis of the feedback from students and of the answers to an open-question sent to 9 FLOSS projects. In this paper, we present guidelines to communities that want to provide better support to newcomers, based on the results of the studies mentioned.

3. Barriers to newcomers

To better support newcomers' onboarding, the barriers they face must be identified and understood. Since FLOSS communities require developers with specific skills, and delivering a task to an FLOSS project is usually a long, multi-step process, some newcomers may lose motivation and even give up contributing if there are too many barriers to overcome during this process. As Karl Fogel states, *"if a project doesn't make a good first impression", newcomers may wait a long time before giving it a second chance."*

The first contribution period is particularly relevant to FLOSS projects, since many newcomers do not want to join or remain at the project, only to post a single contribution (e.g., a bug fix or a new feature). What happens in this period affects, for example, students in computer courses whose assignments include FLOSS project contribution, and professional developers who find a bug or wish to customize a particular software product. With a more in-depth understanding of the barriers, researchers and community can invest their efforts in building or improving tools and processes, ultimately gaining more contributions.

A study aiming to identify the barriers faced by newcomers was carried out by collecting data from interviews with newcomers and experienced members of FLOSS projects and feedback from students that attempted to contribute and from the literature [Steinmacher and Chaves and et al. 2014; Steinmacher and Wiese and et al. 2014; Steinmacher et al. 2015]. By analyzing the data, 58 barriers have been identified and organized in a model composed of seven different categories: Newcomers' orientation; newcomers' characteristics; communication; documentation; local workspace setup; code/architecture; change submission. The model with the categories is presented in Figure 2.

3. Research Method

We conducted a qualitative study relying on different data sources to identify and understand the barriers that hinder newcomers' onboarding to FLOSS projects. Our data sources consisted of data obtained from:

- **Source 1:** feedback from students that contributed to FLOSS projects;
- **Source 2:** answers to an open question sent to developers' mailing lists of FLOSS projects
- **Source 3:** semi-structured interviews conducted with newcomers and members of OSS projects.

The first source (Source 1) consisted of feedback received from four PhD candidates and five undergrad students after contributing to FLOSS projects as part of a course assignment. All the students were newcomers to the projects they were contributing. The students received the same assignment: contribute with code to an FLOSS project. The contribution should include bug fixes and/or new features implementation. The students contributed to the JabRef (2 graduate/2 under-graduate), LibreOffice (2 undergraduate), and Mozilla Firefox (3 graduate) projects. After the conclusion of the assignment, their feedback was collected by means of an open-ended questionnaire. The goal of the questions was to enable students to debrief, and provide the general problems they faced during their onboarding and pointing solutions and mechanisms they used to overcome such problems.

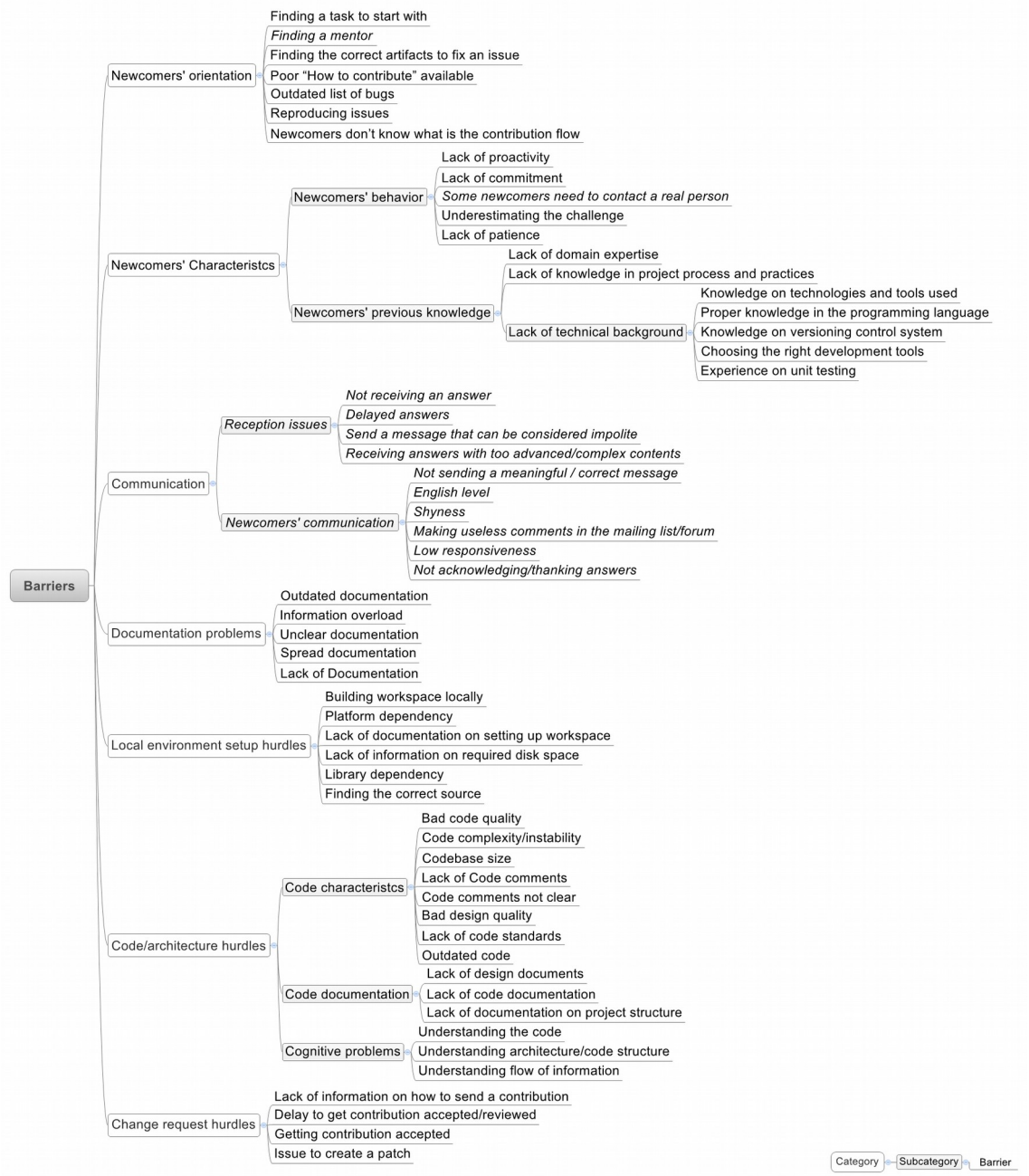

Figure 2. Model of barriers faced by newcomers to FLOSS

The second data source (Source 2) was composed of answers to a questionnaire sent to contributors of FLOSS projects. The data was obtained from 24 answers to an open question sent to developers mailing lists and forums of OSS projects. The messages were posted and the answers received during October 2013. We sent the message to 9 different projects: atunes, audacity, LibreOffice, Apache OpenOffice, Mozilla Firefox, jEdit, OpenVPN, FreePlane and emacs. We chose projects from different business domains. The questionnaire delivered to the community members comprised two

questions to profile the contributor (project and contribution time), and an open question: *"In your opinion, what are the main difficulties faced by newcomers when they want to start contributing to this project? (Consider technical and non-technical issues)."* We received 24 complete answers to the questionnaire, from contributors of nine different projects from people that contributed to the projects for different periods (ranging from newcomers to experienced members).

The final data collection (Source 3) was done by means of semi-structured interviews with practitioners. The reason to conduct interviews was to complement the findings gathered from sources 1 and 2, deepening and broadening the understanding about the barriers faced by newcomers. We recruited subjects that belong to four different groups:

- **Experienced members:** project owners, managers, or developers allowed to commit code directly to the software repository for more than one year.
- **Newcomers that succeeded:** participants that started to contribute to the project less than one year before the interview.
- **Dropout Newcomers:** volunteers that tried to contribute to the project, but gave up;
- **Onboarding Newcomers:** volunteers that were attempting to make their first contributions.

We interviewed 36 participants from 14 different projects (Pardus, TextMate, zxing, Gephi, Hadoop, jEdit, Moodle, Integrade, Noosfero, OpenOffice, cogroo, etherpad, JabRef, and LibreOffice), including 11 experienced members, 16 newcom-ers that succeeded, 6 dropout newcomers, and 3 newcomers that were still trying to place their first contributions. The interviews were conducted from October 2013 to March 2014.

Our main goal was to find the barriers that potentially hinder newcomers from contributing to FLOSS projects. However, our interview guide included questions related to solutions and information that are in-place or that were used by the newcomers to overcome barriers.

We qualitatively analyzed the data aiming to identify of new concepts by using coding approach. Coding means attaching codes, or labels, to pieces of text which are relevant to a particular theme or idea, grouping and examining the ideas. We applied open coding, because our goal was to identify barriers. The coding was performed using the ATLAS.ti1 tool.

4. Guidelines for projects that want to facilitate the onboarding

Based on interviews, observation of students attempting to contribute to FLOSS projects and analysis of processes and practices of several free/libre open source software projects, a set of guidelines useful for projects that want to offer appropriate newcomer support have been proposed:

- **Answer quickly**. Remember that participants are people who chose to spend time trying to help. Do not let their motivation decrease. *Do not make the newcomers wait or leave them without answer*. *Automatic greetings* could help [Preece 2004], at least to send the message that someone will answer them quickly, or to guide them to another possible communication channel.
- **Be kind and make newcomers feel part of the team**. Make newcomers feel welcome; treat all of them as potential contributors and show them that the community cares about them. Experienced members should answer newcomers' messages and welcome them, even if they have already received an answer. *Designating a few experienced members* to deal with new members, or setting a code of conduct can potentially solve reception issues. Send thankful, welcoming messages to account for cultural differences and misunderstandings.
- **Local/regional communication channels**. Whenever possible, create regional mailing lists, IRC channels, and forums. This type of resource can encourage newcomers' first contact and reduce the barriers related to cultural difference and language.
- **Create a newcomer-specific page**. Give the newcomers every resource they need, and only the resources they need. Do not flood newcomers with every possible resource, since too much information can confuse them. Show only what is important for newcomers' first steps, like how the project is organized, and what/where are the important resources (code repository, mailing lists, issue tracker, IRC channel, code review tool). Keep the page clean, organized, up-to-date, and easy to follow. Make this space a kind of "new developers' guidelines" section.
- **Set expectations and needs early**. Show newcomers what is expected from them, where the difficulties lie, and what skills and level of expertise they need to have (what programming languages and technologies are used by the project, etc.). Place this information somewhere that newcomers access early in their journey.
- **Show the path, the easiest path.** Create an easy to access, easy to follow path on which newcomers can start their journey. Leave breadcrumbs, left intentionally for them, to guide the newcomers to easy tasks. The newcomers need to get rewards soon, so the less energy they dedicate to overcoming the initial set of barriers, the

better. This is what Fogel [2013] called '*hacktivation energy*'. Projects need to "*bring the hacktivation energy down to a level that encourages people to get involved.*"

- **Point newcomers to easy tasks**. If there is no way to map a special path for newcomers, at least tag the issues to help newcomers find suitable tasks for new contributors. Some informative tags that can guide newcomers include: difficulty level, module affected, language/technology skills needed, and members who can help.
- **Create different kinds of tutorials and documents.** Provide videos, demos, screenshots, and step-by-step tutorials for the different contribution phases. Make these tutorials simple and clear, especially for more complicated activities.
- **Make it easy for newcomers build the system locally.** Setting up the local workspace was the most reported barrier in our study that demotivated and frustrated many newcomers. One option for helping newcomers overcome setup barriers is to create a step-by-step detailed tutorial, which is linked to information about usual problems and possible solutions (an excerpt of FAQ section).
- **Keep the issue list clean and triaged**: Read the issue list frequently in order to clean outdated tasks and triage/tag issues. Outdated, comment-less issues can be scary to newcomers.
- **Document the processes and practices**: Document all design decisions, processes and practices defined and make them easily accessible. Information like code/naming standards, patch submission process, task assignment process, communication practices, design decisions etc. need to be properly documented, helping newcomers in their first contributions. Remember to present the technical and social processes and practices.
- **Dismiss or identify outdated information**: if it is hard to keep documentation up-to-date, community members should remove outdated information or, at least, clearly identify it as outdated. Making newcomers aware of the absence or the status of a document can save their time and set their expectations. By recognizing the absence or obsolescence of some documents, communities can request help from the newcomers to update or create such documentation.
- **Create a live FAQ section**. Create FAQ sections to help developers finding answer to recurrent questions. The FAQ must not be a static section; it must be live and grow according to questions and issues recurrently asked or reported. The community can build the FAQ cooperatively, in a wiki-like page, enabling anyone to contribute with entries (question + answer).
- **Keep the code as simple as possible**. Code is supposed to be read by humans. Refactor the code constantly to make it more readable. The source code is the set

of artifacts that need to be understood and changed by the members to contribute. Sometimes it is not easy to make it simple because the core of a large application is inherently complex, and the code reflects this complexity. However, directing newcomers to peripheral modules (at least warning them of the complexity) would benefit newcomers during their first steps.

- **Use and update code comments**. It makes it easier for newcomers to understand the code.
- **Document the code structure.** Clearly document the way that the code is organized, and how the components, modules, classes, packages relate to each other. A thorough documentation about the structure and relationship among its modules can make it easier for the new developers to understand the code and find the artifacts they need.

5. Conclusion

A key aspect in FLOSS community formation is the influx of newcomers. We brought an explanation and presented the categories of barriers that can influence newcomers' onboarding to FLOSS projects. Communities that want to receive more newcomers and build a stronger community should start by addressing the barriers presented. To help communities, in this paper we brought a set of guidelines that can be followed to offer a better support to newcomers. We expect that, by applying the guidelines, the projects receive more contributions and eventually support community formation.

References

Begel, A. and Simon, B. (2008). Novice Software Developers, All over Again. In *4th Intl. Workshop on Computing Education Research.* , ICER '08. ACM.

Canfora, G., Di Penta, M., Oliveto, R. and Panichella, S. (2012). Who is Going to Mentor Newcomers in Open Source Projects? In *Proceedings of the ACM SIGSOFT 20th International Symposium on the Foundations of Software Engineering*. , FSE '12. ACM.

Cubranic, D., Murphy, G. C., Singer, J. and Booth, K. S. (2005). Hipikat: a project memory for software development. *IEEE Transactions on Software Engineering*, v. 31, n. 6, p. 446–465.

Dagenais, B., Ossher, H., Bellamy, R. K. E., Robillard, M. P. and Vries, J. P. De (2010). Moving into a new software project landscape. In *32nd International Conference on Software Engineering*.

Fogel, K. (2013). *Producing Open Source Software: How to Run a Successful Free Software Project*. First ed. O'Reilly Media.

Halfaker, A., Kittur, A. and Riedl, J. (2011). Don't Bite the Newbies: How Reverts Affect the Quantity and Quality of Wikipedia Work. In *7th Intl. Symposium on Wikis and Open Collaboration*. , WikiSym '11. ACM.

Jensen, C., King, S. and Kuechler, V. (2011). Joining Free/Open Source Software Communities: An Analysis of Newbies' First Interactions on Project Mailing Lists. In *System Sciences (HICSS), 2011 44th Hawaii International Conference on*.

Malheiros, Y., Moraes, A., Trindade, C. and Meira, S. (2012). A Source Code Recommender System to Support Newcomers. In *36th Annual Computer Software and Applications Conf. (COMPSAC)*.

Nakakoji, K., Yamamoto, Y., Nishinaka, Y., Kishida, K. and Ye, Y. (2002). Evolution Patterns of Open-source Software Systems and Communities. In *Proceedings of the International Workshop on Principles of Software Evolution*. , IWPSE '02. ACM. http://doi.acm.org/10.1145/512035.512055.

Park, Y. and Jensen, C. (2009). Beyond pretty pictures: Examining the benefits of code visualization for open source newcomers. In *Proceedings of VISSOFT 2009 - 5th IEEE International Workshop on Visualizing Software for Understanding and Analysis*.

Preece, J. (apr 2004). Etiquette Online: From Nice to Necessary. *Communications of the ACM*, v. 47, n. 4, p. 56–61.

Qureshi, I. and Fang, Y. (jan 2011). Socialization in Open Source Software Projects: A Growth Mixture Modeling Approach. *Organizational Research Methods*, v. 14, n. 1, p. 208–238.

Steinmacher, I., Chaves, A. P., Conte, T. and Gerosa, M. A. (2014). Preliminary empirical identification of barriers faced by newcomers to Open Source Software projects. In *Proceedings of the 28th Brazilian Symposium on Software Engineering*. , SBES '14. IEEE Computer Society.

Steinmacher, I., Conte, T., Gerosa, M. A. and Redmiles, D. F. (feb 2015). Social Barriers Faced by Newcomers Placing Their First Contribution in Open Source Software Projects. In *Proceedings of the 18th ACM Conference on Computer Supported Cooperative Work & Social Computing*. , CSCW '15. ACM.

Steinmacher, I., Gerosa, M. A. and Redmiles, D. (2014). Attracting, Onboarding, and Retaining Newcomer Developers in Open Source Software Projects. In *Proceedings of the Workshop on Global Software Development in a CSCW Perspective*. , CSCW '14 Workshops.

Steinmacher, I., Silva, M. A. G. and Gerosa, M. A. (2014). Systematic review on problems faced by newcomers to open source projects. In *10th International Conference on Open Source Software*.

Steinmacher, I., Wiese, I., Chaves, A. P. and Gerosa, M. A. (2013). Why do newcomers abandon open source software projects? In *International Workshop on Cooperative and Human Aspects of Software Engineering (CHASE)*.

Steinmacher, I., Wiese, I. S., Conte, T., Gerosa, M. A. and Redmiles, D. (2014). The Hard Life of Open Source Software Project Newcomers. In *International Workshop on Cooperative and Human Aspects of Software Engineering (CHASE 2014)*.

Von Krogh, G., Spaeth, S. and Lakhani, K. R. (2003). Community, joining, and specialization in open source software innovation: A case study. *Research Policy*, v. 32, n. 7, p. 1217–1241.

Vora, P., Komura, N. and Team, S. U. (2010). The n00b Wikipedia Editing Experience. In *6th Intl. Symposium on Wikis and Open Collaboration*. , WikiSym '10. ACM.

Wang, J. and Sarma, A. (2011). Which bug should I fix: helping new developers onboard a new project. In *Proceedings of the 4th International Workshop on Cooperative and Human Aspects of Software Engineering*. , CHASE '11. ACM.

Zhou, M. and Mockus, A. (jun 2012). What make long term contributors: Willingness and opportunity in OSS community. In *Software Engineering (ICSE), 2012 34th International Conference on*.