Let me in: Guidelines for the Successful Onboarding of Newcomers to Open Source Projects

Igor Steinmacher
Federal University of Technology, Paraná, Brazil / Northern Arizona University, USA

Christoph Treude
University of Adelaide, Australia

Marco Aurélio Gerosa
Northern Arizona University, USA / University of São Paulo, Brazil

ABSTRACT

Many community-based open source software (OSS) projects depend on a continuous influx of newcomers for their survival and continuity; yet, newcomers face many barriers to contributing to a project for the first time, leading in many cases to dropouts. In this paper, we provide guidelines for both OSS communities interested in receiving more external contributions, and newcomers who want to contribute to OSS projects. These guidelines are based on our previous work, which characterized barriers encountered by newcomers and proposed tools to support their onboarding. Moreover, these guidelines may help increase the number of contributors and contributions to OSS projects, as well as promote the growth and diversity of OSS communities.

1. Introduction

The Open Source Software (OSS) is an important driving force in today's software industry, resulting in many prominent projects that are extensively used throughout the development stack, from kernels to sophisticated end-user applications. It is, therefore, no surprise that the OSS movement attracts a large, globally distributed community of volunteers. Volunteers' diverse motivations for participation have been thoroughly studied [1], [2] and classified [3] into three categories: extrinsic (payment, career/portfolio building), externalized intrinsic (reputation, reciprocity, learning, code for own use), and intrinsic (OSS ideology, altruism, kinship, fun).

The survival, long-term success, and continuity of OSS projects requires a continuous influx of newcomers [4]. However, new developers face many barriers when attempting to contribute for the first time [5]; and, since delivering a patch to an OSS project is usually a long, multi-step process,

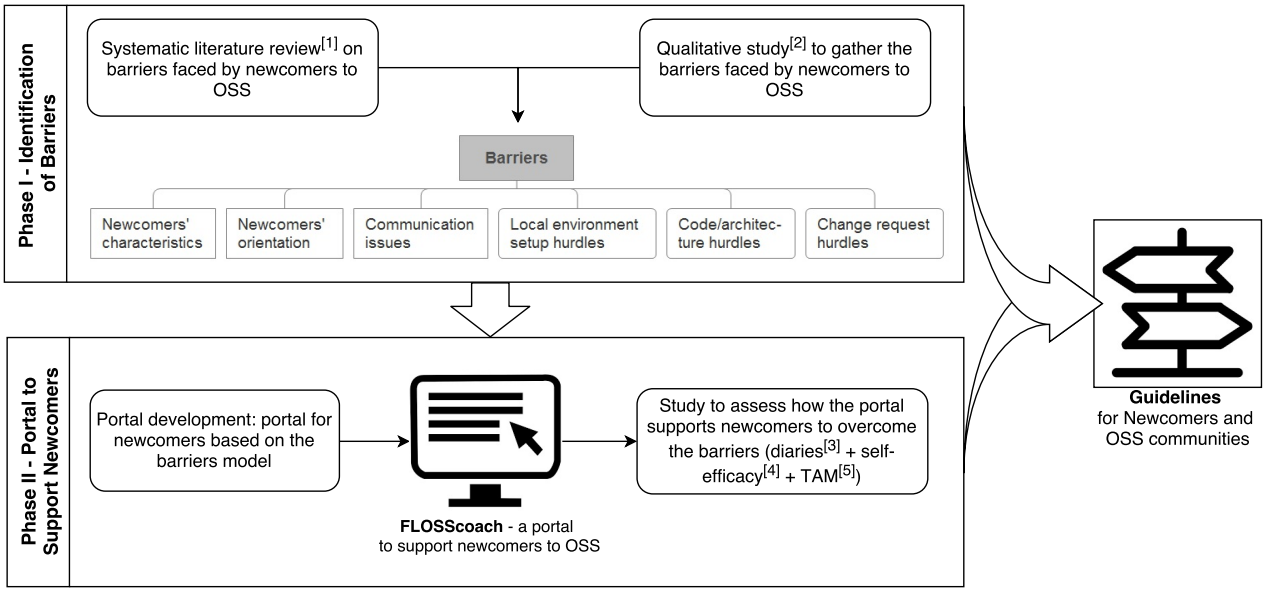
some newcomers lose motivation, or even give up on contributing. The first-contribution barriers affect not only those interested in becoming core project members, but also those who wish to submit a single contribution (e.g., a bug fix or a new feature). Such "casual contributors" are becoming increasingly common; we identified that around 50% of the contributors to top OSS projects make a single contribution, without any long-term commitment to the community [6]. Lowering first-contribution barriers may leverage newcomers' latent contribution potential, thereby benefiting communities, which ultimately receive more diverse contributions.

To better understand first-contribution barriers, in our previous work we empirically conceived of a model containing 7 categories of barriers faced by newcomers to OSS projects [5]. Based on this model, we designed and evaluated FLOSScoach (http://www.flosscoach.com), a portal to better organize information for newcomers and support their first steps [7]. Our results showed that organizing existing information and strategies according to the barriers model made newcomers feel more confident and oriented. Moreover, newcomers perceived the portal as useful and easy-to-use, and indicated they would use it in the future. We briefly summarize in Section 2 the method and results this research achieved.

Backed by the evidence gathered from these studies, the contribution of this paper is a set of guidelines for OSS communities that want to offer appropriate support for newcomers, and for newcomers who want to contribute to OSS projects (Section 3).

2. Identifying the barriers and building a portal to support newcomers

To identify barriers and to support newcomers to OSS projects, we conducted a set of studies, as depicted in Figure 1.



[1] Kitchenham, B. 2004. Procedures for Performing Systematic Reviews. Technical Report #TR/SE-0401. Department of Computer Science, Keele University.

[2] Strauss, A., Corbin, J.M. 2007. Basics of Qualitative Research: Techniques and Procedures for Developing Grounded Theory. SAGE Publications.

[3] Jepsen, L.O., Mathiassen, L., Nielsen, P.A. 1998. Back to Thinking Mode: Diaries for the Management of Information Systems Development Projects. Behaviour and Information Technology. 8 (3). 207-217

[4] Bandura, A. 1977. Self-efficacy: toward a unifying theory of behavioral change.Psychological review. 84 (2). 191-215

[5] Davis, F.D. 1989. Perceived Usefulness, Perceived Ease of Use, and User Acceptance of Information Technology. MIS Quarterly. 13 (3). 319-340.

Figure 1. Method followed to identify the barriers, build the portal, and create the guidelines

To identify and understand the barriers, we performed a Systematic Literature Review, followed by a qualitative analysis of three data sources: (i) feedback obtained from nine students of OSS courses who were newcomers to OSS projects; (ii) 24 answers to an open question sent to OSS communities; and (iii) 35 semi-structured interviews conducted with OSS project members, newcomers, and dropouts. Next, we merged the outcomes, resulting in a model of 57 barriers, which we further organized in 7 categories [4], as shown in Figure 2 (a). Each of the categories is briefly explained below:

- **Newcomers' characteristics (11 barriers)**. This category includes barriers related to the newcomers' experiences and behaviors, including how they demonstrate their knowledge and interact with the community when joining a project.

- **Newcomers' orientation (7 barriers)**. Newcomers often face rugged and unfamiliar landscapes when onboarding to OSS projects. Examples of barriers under this category include difficulty in finding a mentor and poor "how to contribute" documentation.
- **Communication issues (11 barriers).** This category comprises the barriers related to the interactions between newcomers and the community, including lack of or impolite response to a message.
- **Local environment setup hurdles (4 barriers).** This encompasses technical barriers related to setting up the local workspace, including problems with compilation, missing dependencies, and operating system issues.
- **Code/architectural hurdles (9 barriers).** These are barriers related to the existing architecture and source code, including problems related to code characteristics (e.g., code quality, codebase size) and cognitive issues.
- **Change request hurdles (4 barriers).** During the process of submitting the change request, barriers can include lack of information on how to submit the patch, and delay in the review of the request.
- **Documentation problems (11 barriers)**. To become familiar with a project's technical and social aspects, newcomers need to search and use artifacts; in doing so, they may face challenges such as outdated documentation, unclear code comments, information overload, and lack of documentation.

Guided by this classification of barriers, we built FLOSScoach, a web portal for organizing information and strategies gathered from practitioners to support newcomers' first steps. To develop the FLOSScoach portal, each category of the barriers model (Figure 2a) was mapped onto a section that contains information and strategies aimed at supporting newcomers in overcoming those challenges (Figure 2b). Since documentation problems crosscut all other categories addressed by FLOSScoach, the portal does not include this category.

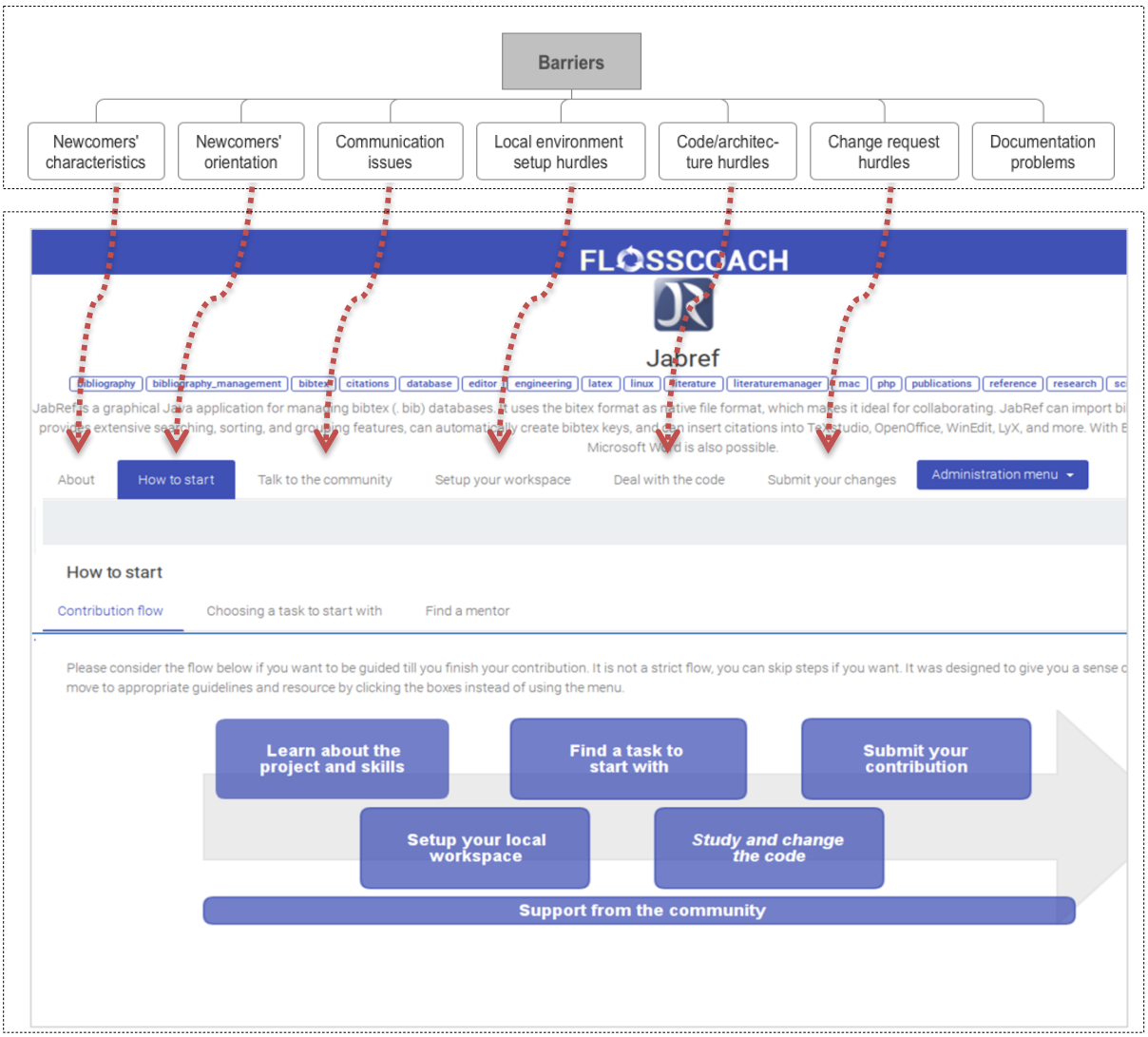


Figure 2. (a) Barriers aggregated by category and (b) how they map into FLOSScoach portal ("How to Start" section)

In the portal, newcomers find information on the skills needed to contribute to a project, a step-by-step contribution flow, the location of features (such as source code repository, issue tracker, and mailing list), a list of newcomer-friendly tasks (when provided by the project), and tips on how to interact with the community. To evaluate the portal, we pre-populated it with existing strategies and information gathered from interviews with experienced members, as well as from manual inspection

of the project web pages of 7 OSS projects: Amarok, Audacity, Banshee, Empathy, JabRef, LibreOffice, and VIM. We chose these projects since they are end-user applications from common domains, and because they align with the programming languages familiar to our study participants. To evaluate the portal, we conducted a controlled study with 65 newcomers (split between a group that made use of FLOSScoach and one that did not), relying on qualitative analysis of diaries, self-efficacy pre- and post-questionnaires, and the Technology Acceptance Model (TAM).

Our results showed that FLOSScoach helps newcomers, guiding them in their first steps and making them more confident in their ability to contribute to a project [7]. To evaluate how the use of FLOSScoach influenced participants' self-efficacy, we analyzed the variation in pre- and post-study answers. By applying a Wilcoxon signed-rank test, we found that the self-efficacy score significantly decreased for the participants who did not use FLOSScoach ($p=0.005$), while there was no significant difference in the group that used the portal. By taking a closer look, we found the decrease was mainly related to social interactions and code issues, while the self-efficacy of newcomers using the portal remained high. We attribute the decrease in self-efficacy to participants encountering unexpected barriers [7]. The results of the Technology Acceptance Model questionnaire indicated that participants perceived a high usefulness, ease of use, and self-predicted future use of the portal.

We qualitatively analyzed the diaries written during the contribution process following Grounded Theory procedures. We found evidence that FLOSScoach made newcomers feel oriented and more comfortable with the process, while those who did not have access to FLOSScoach repeatedly reported uncertainty and doubt on how to proceed. As reported by some participants:

> *"[FLOSScoach] offered the facility to understand how the contribution process works, the links to information about forks, pulls, git commands, tips to send a commit, etc."*

> *"That timeline is very good. I really liked it. I think that for those who are contributing for the first time it is very good, because the person thinks 'what should I do now?' and the answer is there."*

Other participants used the portal as a quick reference guide, to which they could return at any time to find the information needed to finish a task or overcome a problem.

> *"The flow [of FLOSScoach] was great. I always used it, and from here I accessed the other information. It is easy."*

3. Guidelines

Based on the evidence found during interviews, diary analysis, and observation of newcomers attempting to contribute to OSS projects, as well as our examination of several OSS projects, we propose guidelines both for communities that want to offer appropriate newcomer support as well as for newcomers who want to contribute to an OSS project for the first time. All guidelines presented here are backed by evidence previously collected and/or supported by the literature.

3.1. Guidelines for OSS communities

We present guidelines for OSS communities organized in three categories: contribution process; social behavior; and technical. Following these guidelines is especially important for those newcomers who are intrinsically motivated, and who have no further commitment to contributing.

3.1.1 Contribution Process Guidelines

Create a newcomer-specific page or portal. The study conducted with FLOSScoach showed that presenting a structured page, with clean and organized information, orients newcomers and increases their self-efficacy [7]. Therefore, we suggest providing newcomers with all the resources they need (and only those) in a well-organized and easy-to-follow way. This was highlighted by one of our interviewees: *"searching and filtering information would require too much effort."* It is important to show what is essential for their first steps, how the project is organized, and what/where the important resources are (e.g., code repository, mailing lists, issue tracker, IRC channel, and code review tools). Some projects (e.g., Gnome projects and Open Office) offer a "how to contribute" or "introduction to development" page. Complementarily, GitHub encourages maintainers to have this kind of information in a CONTRIBUTING.md file, including general information about ways to contribute, paths to obtaining the current codebase, mailing list addresses and etiquette, introductions to issue tracker systems, build guidelines, etc.

It is also important to **identify and dismiss outdated information.** If it is hard to maintain up-to-date documentation, community members should remove outdated information, or at least clearly identify it as such. Outdated documentation demotivates newcomers, as described by this newcomer we interviewed: *"the information was outdated on the wiki. So, at least as a newcomer, it was quite challenging to get past those errors… you do tend to get bored of that after awhile."* By recognizing

the absence or obsolescence of documents, communities can request help from newcomers to update or create such documentation. This is also an opportunity for (semi-)automated documentation-generation, involving gathering documentation from different sources inside and outside the project and filtering it down to up-to-date information relevant to newcomers.

One way to make the path easier is to **point newcomers to easy tasks.** "Finding a task to start with" was the second most recurring barrier in our model, which greatly impacted newcomers' motivation. Some projects tag issues to help newcomers find tasks that are easy and suitable for them (tagging them as "easy-enough," "good-for-newcomers," etc.). Some additional information that guides newcomers includes task difficulty level, modules affected, language/technology skills needed, and project members who can help. Some projects, such as LibreOffice, include a section called "Easy Hacks" on their wiki page. On this page, newcomers find tasks filtered by difficulty, skills needed, and topics. Apache Open Office, Mozilla, Gnome, Media Wiki, and Ubuntu apply similar strategies. Another recommendation for projects that are part of an ecosystem is to create a common pool of tasks, which can increase the diversity of tasks and promote knowledge transfer among projects [8]. For example, the project Up-for-grabs (http://up-for-grabs.net/#/) provides "*a list of projects which have curated tasks specifically for new contributors*." By using this service, a newcomer can search for easy/starter tasks that need attention, filtering the tasks by, for example, language and platform.

Keep the issue list up-to-date: Providing a tagged list of issues supports newcomers. However, keeping the issue status updated and cleaning outdated tasks requires frequent upkeep. Outdated issues scare and demotivate newcomers, as reported by two newcomers: *"[the project page] guided me through this particular task. But when I tried the next task, turns out it was outdated too. So I kind of gave up;"* and *"Choosing a task was hard […] I started many different tasks and then, in the middle of the job, I discovered that someone else was also working on the same issue."* This last newcomer posted a comment on the issue mentioning he was starting to work on the task; yet, after a few days, another person committed the solution.

3.1.2 Social Behavior Guidelines

Answer quickly. Some of our interviewees reported that they never received a reply: "*They never answered our forum post. We spent a lot of effort on something that was already being done…*" An interview with a core member also revealed this challenge: "*In my opinion, the first [barrier] is not*

getting any reply." Newcomers are mostly volunteers who dedicate their time trying to help. The community should not let their motivation decline by making them wait or leaving them without an answer. Automatic greetings could be used to help [9], at least to say that someone will answer quickly or to guide newcomers to the appropriate communication channel. Leaving a good first impression is very important, as stated by Karl Fogel [10]: *"if a project doesn't make a good first impression, newcomers may wait a long time before giving it a second chance."*

In addition to providing an answer, it is important to **be kind and make newcomers feel part of the team**. A community can make newcomers feel welcome and keep them motivated by treating them as potential. Sending thankful, welcoming messages helps deal with cultural differences and misunderstandings. It is known from the literature [11] that receiving impolite answers demotivates newcomers. In our study, we found an experienced member reporting that "*...some developers may not be suitable for receiving newcomers, they may get angry pretty quickly and kill the interest of the newcomers. Very few of the newcomers know how to behave against this kind of tough developer.*" Designating a few experienced members to communicate with new members or setting a code of conduct are possible solutions to such reception issues.

Identify mentors or experts. As noted by a newcomer, "*[for someone] who wants to do some stuff with an open source project, probably some basic handholding would help."* Lack of mentorship was identified by both newcomers and core members as a barrier, as mentorship can play a role in keeping newcomers motivated and helping them overcome potential barriers. Large OSS projects (e.g., Apache, LibreOffice, and Mozilla) already provide mentoring programs. In Mozilla projects, for example, some bugs are mentored. On Mozilla's page, they are presented as bugs that *"have a mentor who commits to helping you every step of the way. Generally, there should be enough information in the bug to get started…".* Apache also offers a mentoring program that focuses on providing mentors for anyone interested in contributing. The LibreOffice community provides a wiki page called "Find the Expert," where they list a set of developers who are experts in specific knowledge areas within the project. In addition, Google's Summer of Code program, which provides scholarships for students interested in writing code for OSS projects, assigns mentors to support the students during their scholarship period [12].

3.1.3 Technical Guidelines

Make it easy for newcomers to build the system locally. Setting up the local workspace was the most-reported barrier in our first study. This barrier also demotivated and frustrated many newcomers during the FLOSScoach study: *"I am still trying to build, because many errors occurred […] I was expecting to move forward, because so far I did not have time to look at the source code […] It is frustrating."* One option is to create a detailed, step-by-step tutorial that links to information about common problems and possible solutions (an FAQ section). Another potential solution would be a virtual machine with preconfigured build environments [13], web-based IDEs, or a container management tool such as Docker (www.docker.com).

To help newcomers understand the code, projects should **document the code structure**. It is important to clearly document the code's organizational structure, and how the components, modules, classes, and packages relate to each other. Some newcomers mentioned in their feedback that thorough documentation of the structure and relationship among modules makes it easier for them to understand the code and find the artifacts they need. A newcomer suggested the use of diagrams to help understand the project organization: "*We need documentation, preferably diagrams that show how the project is organized…".* Another one suggested "*visual maps to support comprehending the overall system,"* which has already been positively evaluated [14]. Our study with FLOSScoach showed that simply organizing the existing information and documentation was not enough to help newcomers overcome technical barriers, however, and we believe that this is a gap that demands additional research and tools.

3.2. Guidelines for newcomers to OSS projects

In addition to the guidelines for OSS communities, in the following we present advice for newcomers to OSS projects.

3.2.1 Contribution Process Guidelines

Find an easy task to start with. Newcomers should try to identify whether the community tags its issues with a specific keyword to identify suitable tasks. A core member reported that "*The task makes some sense, but it is huge, and the newbie thinks she'll be able to implement it in a few days*." According to this member, attempting to tackle a huge task can be demotivating. Choosing small

tasks with fast rewards can be the first step to both becoming known in the project community and staying motivated.

Keep the community informed about decisions. When a newcomer chooses a task to work on, they should add a comment to the issue stating that they are trying to address it. In the same way, they should inform the community if they give up or find any problem related to the task keeping the task updated. This way, they avoid reproducing the work of other developers, which was found as a barrier [5].

3.2.2 Social Behavior Guidelines

Be proactive. Newcomers ought to try to overcome the barriers they face by searching for solutions themselves. This is indeed expected by the community, as stated by one core member: *"You cannot wait for other people. You have to be willing to study new stuff by yourself."* A member of another project mentioned, *"I think the only requirement is that when a newcomer asks, we want to see that he or she did some research before asking."* Therefore, before contacting the community, we suggest searching the mailing list archives, other resources made available by the community, as well as specialized forums to solve problems. Newcomers may find that their question has been answered previously.

Do not be afraid of the community. If newcomers cannot solve problems by themselves or by using the available resources, it is important they reach out to the community through the appropriate communication means. We found that shyness is a barrier; however, communicating is necessary in collaborative environments such as OSS projects. Newcomers should thus take caution when interpreting the answers they receive. Cultural differences may impact the way people communicate, as one experienced member highlighted: *"a guy sent me a rude message, but, you know, we Brazilians are not used to the 'German way of talking directly'."*

Send a kind and meaningful message. Tips on how to send a message to the community include: be kind, mention your skills and goals, ask your question clearly and objectively, and explain the steps you took to solve your problem before referring to the community. While evaluating FLOSScoach, we found that using a template was helpful ("*I liked the message template, showing how to introduce myself and to present the problems I am facing."*). The mentioned template is presented below:

Hello,
My name is [your name] and I am a newcomer trying to place my first code contribution to Amarok. I am facing problems [during my first steps/finding a task/setting up my workspace]. Can someone help me [clarifying some questions / mentoring me]?
I have already [mention the things you have done already to try to solve your problem]
[If you are getting an error, include it in the message]
[Mention the OS you are working on and the tools you are using]
Thanks in advance
[YOUR SIGNATURE]

3.2.3 Technical Guideline

We recommend that newcomers **use a virtual machine to set up their local workspace**. We observed many problems related to previously-installed packages, dependencies from other software, and unsupported versions of operating systems. Some newcomers reported that they had to reinstall their operating systems after installing and uninstalling dependencies. Installing a new operating system in a virtual machine prevents newcomers from encountering some of these problems and from crashing already-installed applications. In addition, by using a virtual machine, newcomers can use the operating system and development tools recommended by the project.

4. Conclusion

The FLOSScoach portal and the proposed guidelines can serve as a starting point for open source communities aiming to support newcomers and alert them to barriers they might face. We expect that the positive results we obtained when using a portal that organizes information to support newcomers encourages communities to invest in this direction.

A smooth first contribution may increase the number of contributions an OSS project receives from new developers. Our results are in line with a recent study by Carillo et al. [15], which demonstrated the importance of key socialization factors (e.g., task segregation, task purposefulness, interaction intensity, and supportiveness) in engaging and retaining newcomers. We believe that the guidelines presented in this paper have implications for how communities receive newcomers and for guiding

newcomers' behavior while attempting to contribute. Ultimately, newcomers can help to expand the OSS movement, including its diversity and range.

References

[1] K. R. Lakhani and R. G. Wolf, "Perspectives on Free and Open Source Software," in *Perspectives on Free and Open Source Software*, J. Feller, Ed. Cambridge, Mass.: The MIT Press, 2005, pp. 1–22.

[2] C. Hannebauer and V. Gruhn, "Motivation of Newcomers to FLOSS Projects," in *12th International Symposium on Open Collaboration*, 2016, p. 1.

[3] G. von Krogh, S. Haefliger, S. Spaeth, and M. W. Wallin, "Carrots and Rainbows: Motivation and Social Practice in Open Source Software Development," *MIS Q.*, vol. 36, no. 2, pp. 649–676, Jun. 2012.

[4] I. Qureshi and Y. Fang, "Socialization in Open Source Software Projects: A Growth Mixture Modeling Approach," *Organ. Res. Methods*, vol. 14, no. 1, pp. 208–238, Jan. 2011.

[5] I. Steinmacher, T. Conte, M. A. Gerosa, and D. F. Redmiles, "Social Barriers Faced by Newcomers Placing Their First Contribution in Open Source Software Projects," in *18th ACM Conference on Computer Supported Cooperative Work & Social Computing*, 2015, pp. 1–13.

[6] G. Pinto, I. Steinmacher, and M. A. Gerosa, "More Common Than You Think: An In-depth Study of Casual Contributors," in *23rd International Conference on Software Analysis, Evolution, and Reengineering, (SANER 2016)*, 2016, pp. 112–123.

[7] I. Steinmacher, T. U. Conte, C. Treude, and M. A. Gerosa, "Overcoming Open Source Project Entry Barriers with a Portal for Newcomers," in *38th International Conference on Software Engineering*, 2016, pp. 273–284.

[8] A. Sarma, M. A. Gerosa, I. Steinmacher, and R. Leano, "Training the Future Workforce Through Task Curation in an OSS Ecosystem," in *2016 24th ACM SIGSOFT International Symposium on Foundations of Software Engineering*, 2016, pp. 932–935.

[9] J. Preece, "Etiquette Online: From Nice to Necessary," *Commun. ACM*, vol. 47, no. 4, pp. 56–61, Apr. 2004.

[10] K. Fogel, *Producing Open Source Software: How to Run a Successful Free Software Project*, First. O'Reilly Media, 2013.

[11] V. Singh, "Newcomer integration and learning in technical support communities for open source software," in *17th ACM International Conference on Supporting Group Work*, 2012, pp. 65–74.

[12] J. O. Silva, I. S. Wiese, D. M. German, I. Steinmacher, and M. A. Gerosa, "How Long and How Much: What to Expect from Summer of Code Participants?," in *33rd IEEE International Conference on Software Maintenance and Evolution (ICSME 2017)*, 2017.

[13] V. Wolff-Marting, C. Hannebauer, and V. Gruhn, "Patterns for tearing down contribution barriers to FLOSS projects," in *12th International Conference on Intelligent Software Methodologies, Tools and Techniques*, 2013, pp. 9–14.

[14] Y. Park and C. Jensen, "Beyond pretty pictures: Examining the benefits of code visualization for open source newcomers," in *5th IEEE International Workshop on Visualizing Software for Understanding and Analysis*, 2009, pp. 3–10.

[15] K. Carillo, S. Huff, and B. Chawner, "What makes a good contributor? Understanding contributor behavior within large Free/Open Source Software projects--A socialization perspective," *J. Strateg. Inf. Syst.*, 2017.

Igor Steinmacher *is an Associate Professor in the Department of Computing at the Federal University of Technology—Paraná and a postdoctoral scholar in the School of Informatics, Computing, and Cyber Systems at the Northern Arizona University, where he researches human aspects of software engineering and related topics. Steinmacher received a PhD in computer science from the University of São Paulo. Contact him at igorfs@utfpr.edu.br.*

Christoph Treude *is a faculty member in the School of Computer Science at the University of Adelaide, Australia. Before joining the University of Adelaide, he worked as a postdoctoral researcher at McGill University in Montréal, Canada, and he conducted research in Brazil at DIMAp/UFRN in Natal as well as at IME/USP in São Paulo. He received his Diplom degree in Computer Science/Management Information Systems from the University of Siegen, Germany, and his PhD degree in Computer Science from the University of Victoria, Canada. Contact him at christoph.treude@adelaide.edu.au.*

Marco Aurélio Gerosa *is an Associate Professor at the Northern Arizona University. His research lies in the intersection between Software Engineering and CSCW/Social Computing. He is or was program committee chair of events such as IEEE ICGSE and CRIWG, member of the program committee of events such as FSE, ACM CSCW, MSR, and SANER, and guest editor of 4 journal special issues. Gerosa received a PhD in computer science from the Pontifical Catholic University of Rio de Janeiro (PUC-Rio). Contact him at marco.gerosa@nau.edu.*