Engaging Students in Open Source: Establishing FOSS Development at a University

Matthias Müller
Graz University of Technology
mueller@ist.tugraz.at

Christian Schindler
Graz University of Technology
cschindler@ist.tugraz.at

Wolfgang Slany
Graz University of Technology
wslany@ist.tugraz.at

Abstract

Open source is widely used for educational purposes in higher education around the world. While many educators use open source resources for teaching, there seems to be few contributions to such projects of students as part of their university courses. In this work we present our experience on establishing open source development from student contributors as part of their university curriculum. Since 2010 more than 300 students from Graz University of Technology have been involved in the presented Catrobat project and have gained knowledge about agile software development as well as several related domains, e.g., project management, marketing, or graphical design. In this paper we provide detailed insights into the project's organization and evaluate in a study how students feel in this setting. As we conclude, bringing open source to university courses is an effective practical approach based on social learning and provides benefits for students and researchers.

1. Introduction

The digital transformation has changed education tremendously in the last years. E-learning and open education are nowadays commonly used at schools and universities, providing new possibilities and chances for educators and learners [1]. Open source software, such as Moodle or Wikimedia, are today accepted services and used in classrooms all over the world. Since the beginning of open source, universities, e.g., the MIT or Berkeley, have played an important role in shaping the idea of sharing code [1, 2]. Open source provides various opportunities for universities and especially its students [3]. Awareness of open source software, especially learning management systems, e.g., Moodle, and adapting it is on its rise in higher educational institutions [4]. Although open source software use and development seems to be mainstream at universities all over the world, there seems not to be much literature about how to foster its development in the curricula and research of universities. There is a small number of practical examples and considerations for doing so, pointing out potential gains but also challenges [5, 6, 7]. Participating in open source software development is beneficial for students since they gain access to expert code, development tools, and are connected with a community of skilled programmers [3, 6, 8, 9, 10]. Moreover, companies nowadays are widely using open source solutions and gain benefits in hiring students who already have experience using this software during their studies [2, 11]. Thus, fostering contributions of students to open source software at a university can have a positive impact on their learning success and later career [7]. Bringing open source to classes also has benefits for educators, since it is a cost-effective, scalable, and practical approach [10]. Nevertheless, it is also challenging in several aspects, such as community bonding or grading [6, 9, 11]. We present how open source development can be introduced at universities and how students can learn from it. We use the open source project Catrobat as a case study in our work. This project has its roots at Graz University of Technology (Austria), and the main proponents of the project are either employees or students of it. We introduce how this project is organized and how it is used for university courses. The results of a survey give insights into the background of the participating students.

For our work we consider the following questions:

- *RQ1:* How can open source projects be organized at universities?
- *RQ2:* How to respond to challenges in engaging students in open source during their studies?
- *RQ3:* How do students experience contributing to open source software projects during their studies?

The aim of our work is to provide a positive example how open source development can be brought to and driven forward by universities. We want to encourage

URI: https://hdl.handle.net/10125/60210
ISBN: 978-0-9981331-2-6

more educational institutions to motivate students to be engaged in open source software development and foster an open and innovative mindset. We also highlight challenges that need to be considered.

Several definitions of what is meant by open source can be found in literature, especially connected to the movements of Free (Libre) Software (FS), founded by Richard Stallman [12], and Open Source Software (OSS), described by Eric Raymond [13]. The studied Catrobat project is a classic free/libre software project that uses the GNU Affero General Public License version 3 and the Creative Commons Attribution-ShareAlike 4.0 International Public License. Since most of our findings apply to both FS and OSS projects, and since there is a lot of overlap, we use the term FOSS (Free and Open Source Software) in the following, with the understanding that the findings will also apply to some situations that do not strictly involve free/libre projects.

2. Used methodology and structure

Since our work represents a personal experience, we use qualitative methods to describe a case study, supported by the quantitative results of a survey. The focus of our work is "how" and "why" active involvement in FOSS projects can be beneficial for universities. Thus, the methodology of case studies is well suited to answer our research questions [14, 15]. We use a single case study for our work to analyze the benefits and challenges described in existing literature, and to present the unique project-setting [14]. To create a more holistic understanding of the presented case study and underpin the results, we also discuss quantitative data from two surveys [15]. This mixed method of surveys within a case study helps us to enrich the evidence and gives further insights into the proposed research questions [14].

First, we present the case study of Catrobat: how it was established and how it is organized as an international FOSS organization. To answer RQ1 in detail, we describe the project's setting that has been established in 2010 and been adapted since then to the needs of contributing students and educators. On the one hand, certain of these settings are provided from the university and/or externally from the organization, on the other hand there are several aspects that resulted from the nature of the community.

Second, we describe the insights from an educator's point of view about bringing open source development to universities. To answer RQ2, we point out in detail how students can participate, how certain challenges are handled, e.g., grading or community bonding, and how research projects in the field of open source are conducted.

Third, we answer RQ3. In spring 2018 we conducted two anonymous online surveys for active and former students that have contributed to Catrobat during their studies at Graz University of Technology. 58 of 103 current students we asked have answered the survey (56% response rate) and provided detailed insights into their background, their motivation, and their contribution to the project. We also asked 98 former students to take part in a smaller survey, giving insights into how they see their past contribution to the project and whether it helped their career. 31 of these alumni provided feedback, resulting in a response rate of 32%. Although we are aware that these response rates may lead to a non-response bias, we can identify certain arguments that can be used to answer the presented research questions. We provide a holistic perspective on this unique setting and how it has been driven forward in an innovative way, paying respect to the students, educators, stakeholders, and also the software's users. Last, we discuss the obtained results and their implications, leaving room for further research.

3. The Catrobat project

In 2010 project founder Wolfgang Slany, professor at the Institute of Software Technology at Graz University of Technology, came up with the idea of a mobile programming framework for smartphones similar to the well known Scratch framework developed at the MIT Media Lab. Since no mobile solution existed, he kicked off the Catrobat project, aiming to develop an easy-to-use mobile app allowing to create programs with simple visual bricks. The project's vision is to provide tools that foster computational thinking skills among teenagers, independently of traditional PCs and in an environment that they are nowadays used to: mobile devices. Mobile devices have become part of our everyday life, are widely used by teenagers all over the world, and provide a cheap alternative for computer science classes that often still rely on traditional PCs. Catrobat aims at enabling young smartphone users to express themselves creatively throughout their digital mobile life instead of remaining mere consumers of the underlying technology. Starting with a few interested students, first ideas were implemented, and over the years it gradually became an international project. Several hundred people from all over the world already contributed to the project and delivered code, translations, educational resources, or other support under FOSS licenses, helping to realize the project's vision. In 2014 the first version of Catrobat's free coding

app Pocket Code[1] was released on Google Play [16], attracting more than 500,000 users as of June 2018. An additional drawing app, Pocket Paint, got released at the same time, allowing users to create and design their own graphics for their games and apps. Several extensions for various hardware (e.g., Arduino boards or Lego Mindstorms robots) were added to Pocket Code over the time and further features implemented to provide more possibilities relevant for the young target group. Whereas this Android app is already available to the public, a version for iOS is currently in an alpha testing phase, and a beta version of an HTML5 player for desktop and mobile browsers is available on Catrobat's sharing website[2]. On this sharing site, users can publish their projects created with Pocket Code. Besides the development of the described services, further value has been added by contributors through creating different educational resources, maintaining the community of users, or translating the services in more than 50 languages. Although the project nowadays is based on an international community of contributors, the majority of its developers is connected to Graz University of Technology. The main reason for this is that students are actively recruited, motivated, and supported to contribute to the project during their studies.

4. Project structure

As mentioned, the project got kicked off by a professor and a couple of interested computer science students working on the main Android app (called Catroid in the project's beginning) and its sharing site. Over the years, a community of international contributors was established to bring the project forward. Although many contributors are connected to the university, "Catrobat" itself exists independently as a FOSS project. This ensures the independence of the project and also allows external contributors to easily contribute to the project. These external contributors led to a distributed network of project members who are used to work together from all over the world as well as with new developers, which supports teaching global software engineering in an educational context [10]. The closeness of the contributing community to the university is beneficial, since it makes it easier for students to get into it, something that has been identified as a potential challenge for bringing FOSS development to universities [9]. Open source projects and communities usually don't rely on a strict hierarchical structure, yet contributors are organized by their roles and influence within the community [3, 17]. This is also the case for Catrobat. Contributors are characterized by their role and team that focuses on a certain aspect of the project. Although there is no strict hierarchical structure, open source projects usually have a smaller, shifting leadership group that, while not being able to give strict instructions like in companies, instead can give recommendations and an overall direction to the volunteer contributors [2, 17, 18]. At Catrobat, the overall direction of the project is provided by a committee that is also motivating the community to contribute.

4.1. Project roles and teams

Open source communities are composed of a set of individuals, each having a unique background and personality. Not only developers are part of these communities, but also other types of contributors and users, who collaboratively drive such projects forward [17]. Since the source code is freely available anyone can contribute to the code base [13, 3, 18]. Each contributor can take over one or several roles according to personal interests [17]. These roles are not assigned and can change, depending on the contributor's commitment to the project [17, 19]. Thus, the structure of the organization of a FOSS project varies according to the project's nature and its members [3]. Also governing these communities of contributors usually depends on the project's needs and its structure. Although the structures in general tend to be flat, some leadership (providing a vision, giving recommendations, or keeping the community together) based on trust from the community is needed [2]. This trust is essential for contributors and the overall project, since mistrust may lead the entire project to collapse [20]. The general development and management approach of Catrobat is based on agile principles, e.g., eXtreme Programming [21]. Following these principles also fosters teamwork and guides the community towards a shared goal. It is important to mention that management as well as other responsibilities are usually shared among contributors, reducing complexity and dependencies on particular persons [18]. Although several common roles and structures of open source projects have been identified [3, 19], there are differences between the individual projects. In the Catrobat project, we differentiate between the following roles:

- Users: There are various types of users that are part of the community. However, they are mostly not contributing, or just providing value to other users. Thus, we will not discuss their role in this work.

[1] https://catrob.at/pc
[2] https://share.catrob.at

- *Peripheral Contributor*: Contributors that do not interact with the community and contribute in a narrow or irregular way (e.g., one bug fix commit or a few translations).
- *Active Contributor*: Contributors that are regularly contributing (e.g., code or resources) to the project and are an active part of the community.
- *Senior Contributor*: Experienced contributors that advise newer contributors, take the responsibility to review pull-requests, and accept code.
- *Coordinator*: Coordinating a certain part of the project (team) and guiding the involved contributors. Communication and coordination are the main tasks.
- *Product Owners*: Committee of highly involved contributors (including the project's founder) providing the overall vision and direction of the project.

Students that contribute are treated independently of their status and are therefore not treated differently than other members of the project's community. This aligns with other open source communities, where the role is independent of any attributes (e.g., age) and is earned through contribution [17]. Contributors take one or more of these roles during their contribution, depending on their own preferences and situation within the project.

Besides roles, contributors can also be characterized by their team within the project. To keep such projects manageable and successful it is needed to divide the contributors in teams that can work on well defined tasks and almost independently from other teams [2]. As outlined, the Catrobat project with its many services, features, and other aspects such as education or design, provides many different possibilities for contributions. Contributors typically work in small teams that have a fixed scope and that can work more or less on their own. As illustrated in Figure 1, these teams have a special focus. This enables contributors, including students, to work in a domain they are personally interested in. As mentioned, these teams are guided by a "Coordinator", an experienced and highly engaged contributor. From an educational point of view, this structure gives students the choice to work on a field they like and to develop various skills depending on their interests.

4.2. Communication

A major aspect for open source communities is internal communication. Failing to interact with other members can upset contributors and slow down the whole project by hindering collaboration [20]. Providing communication, documentation, and guidelines for developers and also users is important for FOSS projects right from their start [18]. In the context of Catrobat, services and processes needed to be introduced to allow communication between distributed team-members and also ensure the flow of project relevant information [22]. Catrobat provides a variety of services to tackle these potential issues. As an example, the project itself offers public instances of Jira and Confluence to track the development process and document the overall project structure. While the project in early times used IRC (Internet Relay Chat) for communication [22], Slack channels are now the main place of communication between contributors. These tools are commonly used in industry, fostering the technical skills of the contributing students [9]. Local students are encouraged to use these tools but also meet with other contributing students in person. Although working remotely with external contributors is common, there are also regular meetings between students and the university staff. These meetings aim to enhance communication and also to identify potential problems early. Furthermore, students may use a dedicated room at the institute to meet and work on the project. This room, which has space for approximately 20 students, is the primary work space for more than a quarter (28%) of the currently participating students (over 100).

4.3. Infrastructure and setting

For a FOSS project, various infrastructure has to be maintained to keep the project running, e.g., hard- and software for development as well as internal and end user related services. Projects have to provide various tools, e.g., for communication, information management, or version control [18]. We give an overview over the infrastructure maintained for the Catrobat project and briefly describe it. Some of this infrastructure is provided by the university (especially testing infrastructure), others (e.g., general services and software) by the project itself. We distinguish between hardware, software, and provided services to keep the project running.

Hardware

Dedicated Servers: We are running 12 dedicated machines ranging from desktop-class (32GB Ram, quad-core) to mid-range server-class (256GB Ram, 14 cores/28 threads) used mainly to host virtual machines, except an Apple XCode server and Jenkins instances, to flexibly provide services.

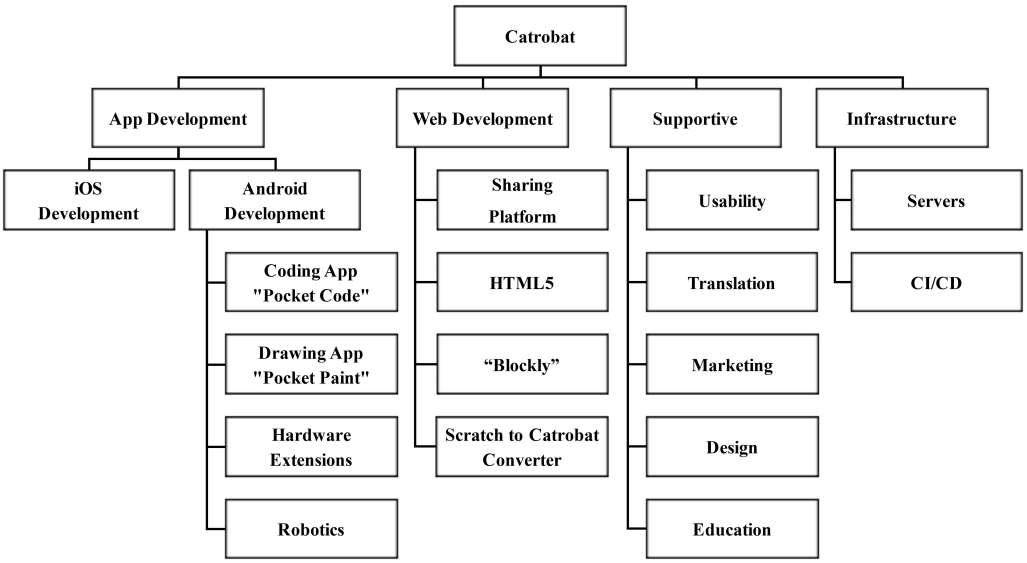

Figure 1. Team structure of Catrobat as of June 2018

Mobile Devices: We provide over 120 mobile devices (phones and tablets) for the testing and development of the project's iOS and Android applications.
Equipment: A dedicated project room at the Institute of Software Technology, open 24/7 for the students, is equipped with 20 LCD screens for the developers and testers, 4 white boards, a color laser printer, and a 50 inch flat screen TV for meetings, presentations, and Jenkins monitoring. Two mobile beamers for presentations and meetings are also available, as well as a coffee machine and air conditioning (both not standard for student projects).
Miscellaneous: This category consists of single-board computers, such as Raspberry Pi and Arduino, electronic and robotic assembly kits, tinkering material, micro/little bit kits, various robots, drones, as well as robot arms for hardware testing on the CI server.

Software
Jira: Atlassian Jira for issue tracking and management of the developers' backlog.
Confluence: Atlassian Confluence is used as a knowledge base for every sub team in the project.
Slack: The daily short term communication is facilitated by various slack channels for every sub team in the project. Most channels are open and can be subscribed if needed.
GitHub: The project's source code is hosted on GitHub.
Crowdin: All project's translations are handled with Crowdin, an online localization management platform.
Yourls: A short link service to resolve short URLs.

Services for internal use
Jenkins: Jenkins is used as a continuous integration platform. The complete tests are run on a regular basis and when GitHub pull requests are issued.
Backups: Additionally to the individual local backups of the different systems, a centralized backup solution with redundant storage is maintained.
LDAP: An LDAP instance is used to log-on to the different services (JIRA, Confluence, Share, Jenkins,...).
Workspace maintenance: The project's open workspace is equipped for convenience reasons with a coffee machine, a fridge, air condition, and office material. Coffee and snacks are refilled regularly.

Services for end users
Catrobat Share: The central point of service for the end user is the project's sharing website https://share.catrob.at/. On this site users can download projects and, if registered, they are able to upload projects.
APK generator: To foster sharing, projects can be downloaded as Android Application Packages (APKs). They are install- and executable on any compatible Android device without the need of the installed Pocket Code app.
Recommender System: The sharing site is backed by a recommender system to suggests similar projects.
Converter: A Scratch-to-Catrobat converter allows to run Scratch programs with Pocket Code.

5. Students' involvement

Students, as other contributors, participate at Catrobat on a volunteer basis only, and clearly understand the rights and limitations entailed by the project's FOSS nature. There is no compulsion for students to contribute, as there are ample alternatives for them to choose from among many other projects and it is not required for their studies. Nevertheless, we provide an attractive environment to get engaged in Catrobat as part of their curriculum if they want to. Various possibilities are available for students to participate in Catrobat, depending on their degree program, the field they are interested in, and current research projects connected to the project.

5.1. Organization at the university level

Besides the described general structure of Catrobat, there is also a university related framework behind the project. Wolfgang Slany and his team of currently three assistants manage the students and take care of them. This team has several research backgrounds, covering fields such as software development, economics, usability, or sociology. This also allows to supervise the students while they are working in different teams of the project, focusing on a variety of domains. Cooperations with other academic institutes and companies provide access to further expertise in specific fields. In the setting of open source projects, instructors act more as a guide, fostering the students' understanding built from experience [6]. Thus, the involved staff offers regular consulting hours, joins the local meetings of students, and actively participates on the project's communication channels. Furthermore, individual meetings with the currently over 100 actively involved students happen on a regular basis, also helping to ensure a positive learning outcome for them. This approach is time-consuming but apparently beneficial, since students get used to work individually in teams, learn from each other, but are still supervised in a way that ensures a relevant outcome.

5.2. Participation model

The project is situated at Graz University of Technology, which currently offers four degree programs in the domain of computer science. Although the scope of these studies varies ("Computer Science", "Software Development and Business Management", "Information and Computer Engineering", and "Teaching Subject Computer Science"), the presented structure enables students of all these fields to contribute to the project within their curriculum. Professor Slany and his team of assistants are offering several practical courses (e.g., "Mobile Applications" and "Software Technology") that are directly part of these curricula. Students have the choice whether they want to work on a traditional task offered in these courses, or whether they want to work on tasks that are directly related to the "Catrobat" project, but still following the scope of the course. The majority of students still take the "classical" courses, with no relation to Catrobat. However, several dozen students every year (e.g., over 40 in the first five months of 2018) choose this participation model. Although the majority of students just stays with the project for a short period of time (usually less than a year), there are students that get committed to the project for a longer period. As shown in Figure 2, by spring 2018 there are students in the project who started 6 years ago with their first contribution and are still an active contributor to it. The introduced setting allows students to stay longer than just a single course, getting committed to the project and gain deep knowledge in different fields related to their studies. But there are also students and even university alumni who voluntarily stay longer with the project for various reasons and do not earn any further credits.

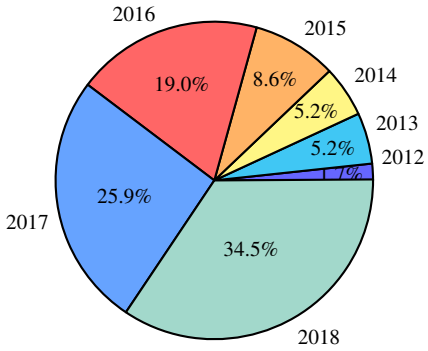

Figure 2. Year active students started contributing to Catrobat as of June 2018

Besides the course-related contribution, students also have the opportunity to write their Bachelor's, Master's, or PhD thesis on a topic related to their contribution to Catrobat. This enables students to stay with the project for a longer time than just a certain course, as described above. The scope of the project provides many domains, e.g., software quality, usability, computer education, or project management, from which students can choose more or less freely for their thesis, according to their interests, the relevance to their studies, the level required by the thesis, and the requirements of the project. This allows them to write about a topic they are personally interested in. Furthermore, their practical work gets directly applied to a worthwhile, relevant, and publicly available

project, impacting a real user-base that provides fast feedback and data for their work. Till today, over 190 bachelor's projects, more than 30 master's theses, and two PhD theses are related to Catrobat. In addition to that, university staff uses the project for their research and in courses, e.g., to teach coding concepts to beginners. Several grants and fundings also allowed to employ student contributors as university staff either part time during their studies or full time after they have graduated. As an example, these employees developed specific project relevant features, e.g. the Right-To-Left language version of Pocket Code, but also support students in their work. Employing students for FOSS development also worked for other universities, such as at the Oregon State University Open Source Lab [7].

This approach allows students and researchers to focus on open source by making it their primary work. By doing so, opportunity costs are minimized, meaning that there is no disadvantage for the participating contributors (e.g., students could get slowed down in their studies if they work on open source unrelated to their curriculum, or an academic's output could be affected negatively if he works on open source besides his or her main research topic) [2]. This approach also ensures the sustainable development of the project, since students' contribution is not limited to a certain course. Time constraints can be challenging since they delay the contribution process and make the engagement of students in FOSS projects difficult [9]. Especially the writing of a thesis or doing project work is not necessarily restricted to the official semesters and can also happen during summer or winter breaks. This gives the students further freedoms and independence, as long as they are discussed with the university's assistants in advance.

As outlined by previous research [6, 11], grading might become a problem in such settings. In general, theses and project works are graded individually, also independently of this approach. Thus, there is no additional effort needed compared to the traditional setting. Students taking the described courses are graded in the same individual way. All students have an initial meeting with the professor and one of the assistants, setting the conditions for their work, also outlining the expected outcome in regard of the learning success. This expected outcome and work is defined with respect to the course they take, the credits they will earn for it, their field of study, as well as their personal interest in a topic. The expected contribution for the defined outcome is hard to predict since the therefore needed work is a group effort and varies by a multitude of factors, e.g. previous experience of the student. Thus, all contributions rewarded by university credits are time boxed by the amount of time expected to be spent depending on the course credits for the student. This is similar to the way most regular employees are rewarded for their work, and allows students to plan their contribution in a tractable way, as well as it ensures that they do not get overloaded with work, e.g., by a too large defined expected outcome. It also forces us to split up the work in small parts, which works well with agile software development methods, and fosters cooperation between contributors. Still, although the contribution is time boxed, the outcome itself, in terms of quality and scope, must relate to the requirements of the taken course. After having their work completed, the students' work gets evaluated by the staff. Therefore, also the feedback of the community is used, since all pull requests to Catrobat get reviewed by experienced contributors before they get merged. Pure peer-grading would not be sufficient, since contributors might not be critical enough [6] or too critical. All progress of the student is tracked through the project's coding tools (i.e., GitHub, Jira, and Confluence), helping the staff to get an overview of the completed work. Although the effort of grading is still higher compared to traditional grading, a good project's infrastructure setting and communication eases the process for educators. Also students benefit from this individual feedback since their strengths and weaknesses can be discussed in a final meeting, showing up potential room for improvement.

6. Motivation for students to contribute

Although this open source project was initiated at a university, and students may earn credits for their contribution, choosing to participate happens on a voluntary basis. This voluntary basis is not only legally required because of the special relationship between students and their university, as they are basically a kind of "customer" of their university, but also psychologically essential for the success of the project, since it is directly connected to the motivation of the contributors [3]. The students can take this either as elective courses, project work, or as part of their thesis. Thus, the role of motivation also needs to be considered to attract motivated students and run such a project in a sustainable and successful way. Not just for contributors in general there is a diverse number of motivations to contribute, but also for students in particular [11]. Research by Ellis, Morelli and Hislop [6] already pointed out several motivational aspects for students (e.g., working on real world projects) that we want to analyze with our survey.

6.1. Motivation to contribute to open source

Previous research pointed out different reasons why people contribute to open source projects (e.g., Hars & Ou [23], Ye & Kishida [3], or van Krogh et al. [24]). Contributing to open source comes along with a variety of benefits and costs for the contributors [2]. A main aspect is that contributors aim to receive a net-benefit, meaning that the benefits of contribution, or innovation, exceed their costs [2, 25]. In this respect, the net benefit is defined individually for each contributor. Failing to recognize the individual goals of community members is a potential source of failure for such projects and can hinder their success [20]. A variety of motivators, such as reputation, one's own use or career, got identified in the literature and got pointed out as overall drivers that get people into open source [24]. Motivation of contributors is a core research topic in open source and has attracted a large number of researchers [26]. Nevertheless, many aspects and questions related to the motivation of volunteer contributors in open source are still not answered properly [24].

It is important to note that there are also paid contributors to open source projects, whose motivation might differ. Having long-term contributors, e.g., through hiring them, allows sustainability, since swapping programmers or introducing new ones slows down the development [18]. Whole businesses emerged around open source and enable contributors to profit from their contributions [27]. Furthermore, corporations started actively to involve themselves in the development of open source software, since it provides manifold benefits [18, 27]. Also Catrobat benefits from university research projects in that developers can get employed for a certain timespan. Also grants for contributing to open source are available for students. As an example, Catrobat has been selected as a mentoring organization for Google Summer of Code for several years now, allowing the project to fund students from all over the world to work on our project during their summer break.

6.2. Surveying active and former project's students on motivational aspects

Following previous literature, we examined the motivation of currently active students of the Catroabt project. As described, we asked 103 active students and received feedback from 58 of them (56% response rate) in an anonymous online survey. The results align with previous studies that outline the importance of future rewards for the motivation of contributors [23]. As illustrated in Figure 3, the majority of students see the credits they earn for participating (72%), the learning of new skills (69%), and the experience they gain (64%) as motivators for their participation. At the same time, the project's vision and idea also got identified as an important motivating aspect for almost all of the students (78%). This matches research by Hars & Ou [23], who found that students are strongly motivated by intrinsic factors (self-determination and altruism), but also strengthening their human capital. One factor that surprisingly has not been rated as high as we expected by the surveyed students is the potential impact on their future career. Only 36% of the participating students claimed that they see it as a reference for their career after graduation. This also aligns with the results that came up for the questions how relevant they would rate their contribution to the project for their career. Students showed very mixed feelings from "not at all" to "very relevant" about this question, without allowing us to get a more in depth answer to this question. This is in contrast to the conducted survey of the project's alumni students who already graduated from university and stopped contributing to the project. We received answers from 31 out of 98 former students asked (response rate of 32%). 71% of these alumni, all of which are working in a related field (e.g., software development or project management), claimed that contributing to this project helped their career, compared to just 36% of currently involved students who subjectively see a benefit for their future career related to their contribution. One possible reason for this could be that students still at the university may not be able to clearly foresee the benefits for their later career, since they do not yet have had the experience of working in industry. Alumni students further had the opportunity to anonymously provide an optional comment within the survey. 13 of the alumni left a comment, of which 8 positively highlighted the impact of working on this project to their career. Especially the usage of professional tools such as Jira or Confluence, the application of agile methods, and working in interdisciplinary teams got pointed out in these comments. Although there is a strong probability of a non-response bias, since there is the chance that only students who have positive feelings about the project participated in the survey, we nevertheless can infer that participating in open source projects during university studies can have a positive impact on the students' later careers. This is also underpinned by the personal informal feedback we received from local ICT companies that employ former students. They state that the onboarding of these employees is sped up due to their previous knowledge in working in development teams, applying common software development methods, and using professional software tools (e.g., git or Atlassian).

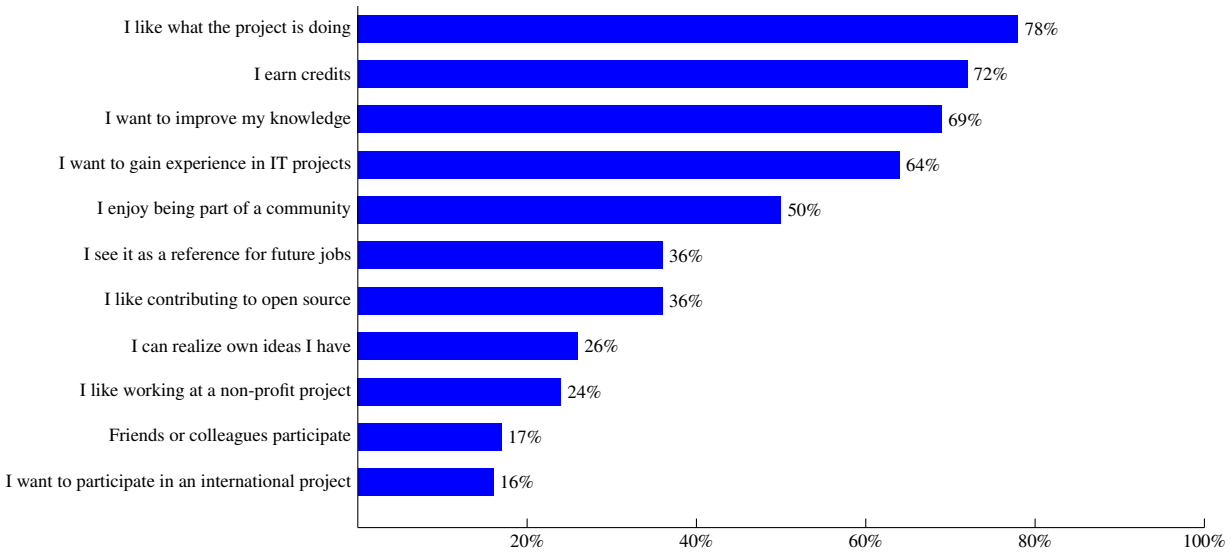

Figure 3. Catrobat students' motivators to join the project

Another factor we surveyed has been the contribution of students to other open source projects. 36% of the surveyed students of Catrobat have already been contributing to other open source projects before their involvement in Catrobat. This aligns with the results of an additional survey done with 104 students of a coding course at the university. A similar number of these students (35%) has already been contributing to open source projects. Also 42% of the surveyed alumni stated that they did so. We can see that students, but also alumni, have an active interest in open source. By establishing the possibility of working on open source projects during their studies, students are supported in their interest in such organizations and furthermore gain important experience in practical software engineering.

7. Discussion

This paper describes a single case that has already been running for several years and was developed from the needs of the involved students and educators. We are aware that the results on the motivation of students and long-term effects on alumni can only be snapshots based on the data over a short period of time. Further insights into the long-term effects are expected from a continuous evaluation of new and leaving students of the project. The described setting shall help other institutions to establish similar projects. More published cases can help to further analyze and evaluate the development of open source software as part of students' university work. Our results suggest that this approach can be repeated at other universities and can help to prepare computer science students for their later career. Nevertheless, we want to encourage more researchers at universities to report their personal experiences of developing open source software in classes in order to create a larger basis for research in this field.

8. Conclusion

The presented approach of bringing open source development by students to universities comes with many benefits but also challenges, especially for the involved educators. The students' personal perception of this approach is very positive and considered as beneficial by alumni. There are strong indicators that this practical setting has a positive effect on the participating students' later career, since they get exposed to real-world problems, have to work in teams, and get to know common professional tools in the field. Furthermore, students in general show a strong interest in open source software. By enabling them to bring this interest to courses, they can gain knowledge and advance their studies at the same time. Also researchers can benefit from the described setting, as it fosters the general research in this domain and gives direct access to real-world problems for potential research projects. Manifold possibilities for research are created, as the case of Catrobat and Graz University of Technology shows. Especially from an organizational point of view (e.g., providing the infrastructure, guiding students, and keeping track of their involvement) additional work and resources have to be invested compared to traditional course settings. But the personal experience of all involved entities shows that the gained benefits, at least in the presented case, outweigh the effort.

9. References

[1] S. E. Lakhan and K. Jhunjhunwala, "Open source software in education," *Educause Quarterly*, vol. 31, no. 2, p. 32, 2008.

[2] J. Lerner and J. Tirole, "Some simple economics of open source," *The Journal of Industrial Economics*, vol. 50, no. 2, pp. 197–234, 2002.

[3] Y. Ye and K. Kishida, "Toward an understanding of the motivation open source software developers," in *Proceedings of the 25th International Conference on Software Engineering*, ICSE '03, pp. 419–429, 2003.

[4] S. W. van Rooij, "Higher education sub-cultures and open source adoption," *Computers & Education*, vol. 57, no. 1, pp. 1171–1183, 2011.

[5] J. D. N. Dionisio, C. L. Dickson, S. E. August, P. M. Dorin, and R. Toal, "An open source software culture in the undergraduate computer science curriculum," *ACM SIGCSE Bulletin*, vol. 39, no. 2, pp. 70–74, 2007.

[6] H. J. Ellis, R. A. Morelli, T. R. De Lanerolle, and G. W. Hislop, "Holistic software engineering education based on a humanitarian open source project," in *Software Engineering Education & Training, 2007. CSEET'07. 20th Conference on*, pp. 327–335, IEEE, 2007.

[7] A. Casson and L. Hawthorn, "Introducing the oregon state university open source lab," *Open Source Business Resource*, 08/2011 2011.

[8] J. Seely Brown and R. Adler, "Open education, the long tail, and learning 2.0," *Educause review*, vol. 43, no. 1, pp. 16–20, 2008.

[9] G. Pinto, F. Figueira Filho, I. Steinmacher, and M. A. Gerosa, "Training software engineers using open-source software: the professors' perspective," in *The 30th IEEE Conference on Software Engineering Education and Training*, pp. 1–5, 2017.

[10] S. Beecham, T. Clear, D. Damian, J. Barr, J. Noll, and W. Scacchi, "How best to teach global software engineering? educators are divided," *IEEE Software*, vol. 34, no. 1, pp. 16–19, 2017.

[11] G. DeKoenigsberg, "How successful open source projects work, and how and why to introduce students to the open source world," in *Software Engineering Education and Training, 2008. CSEET'08. IEEE 21st Conference on*, pp. 274–276, IEEE, 2008.

[12] R. M. Stallman, *Free Software, Free Society: Selected Essays of Richard M. Stallman*. Gnu Press, 2002.

[13] E. Raymond, "The cathedral and the bazaar," *Knowledge, Technology & Policy*, vol. 12, no. 3, pp. 23–49, 1999.

[14] R. Yin, *Case Study Research: Design and Methods*. Applied Social Research Methods, SAGE Publications, 2009.

[15] P. Baxter and S. Jack, "Qualitative case study methodology: Study design and implementation for novice researchers," *The qualitative report*, vol. 13, no. 4, pp. 544–559, 2008.

[16] W. Slany, "Pocket code: a scratch-like integrated development environment for your phone," in *Proceedings of the companion publication of the 2014 ACM SIGPLAN conference on Systems, Programming, and Applications: Software for Humanity*, pp. 35–36, ACM, 2014.

[17] K. Nakakoji, Y. Yamamoto, Y. Nishinaka, K. Kishida, and Y. Ye, "Evolution patterns of open-source software systems and communities," in *Proceedings of the international workshop on Principles of software evolution*, pp. 76–85, ACM, 2002.

[18] K. Fogel, *Producing open source software: How to run a successful free software project*. " O'Reilly Media, Inc.", 2005.

[19] M. M. Torres, S. Toral, M. Perales, and F. Barrero, "Analysis of the core team role in open source communities," in *Complex, Intelligent and Software Intensive Systems (CISIS), 2011 International Conference on*, pp. 109–114, IEEE, 2011.

[20] D. Ehls, "Open source project collapse–sources and patterns of failure," in *Proceedings of the 50th Hawaii International Conference on System Sciences*, 2017.

[21] A. Harzl, "Can foss projects benefit from integrating kanban: a case study," *Journal of Internet Services and Applications*, vol. 8, p. 7, Jun 2017.

[22] S. Fellhofer, A. Harzl, and W. Slany, "Scaling and internationalizing an agile foss project: Lessons learned," in *IFIP International Conference on Open Source Systems*, pp. 13–22, Springer, 2015.

[23] A. Hars and S. Ou, "Working for free? - motivations of participating in open source projects," in *Proceedings of the 34th Annual Hawaii International Conference on System Sciences (HICSS-34)-Volume 7 - Volume 7*, HICSS '01, 2001.

[24] G. Von Krogh, S. Haefliger, S. Spaeth, and M. W. Wallin, "Carrots and rainbows: Motivation and social practice in open source software development," *MIS quarterly*, pp. 649–676, 2012.

[25] E. Von Hippel, "Innovation by user communities: Learning from open-source software," *MIT Sloan management review*, vol. 42, no. 4, pp. 82–86, 2001.

[26] G. Von Krogh and E. Von Hippel, "The promise of research on open source software," *Management science*, vol. 52, no. 7, pp. 975–983, 2006.

[27] H. Chesbrough, *Open business models: How to thrive in the new innovation landscape*. Harvard Business Press, 2006.