Skills needed in open source projects and how to learn them

Juho Leinonen, Tuomas Paulin & Marko Saari

Abstract

1. Introduction
2. Skills needed by contributors to succeed in a project
3. Acquiring skills
4. Discussion
5. Conclusions
References

Abstract

In this literature review we take a look on skills needed by developers to succeed in open source projects and how these skills are learned. Based on previous research we found out that technical skills provide the foundation of skills needed to work on open source software development, but social and people skills together with understanding different cultures forms an important part of the skillset needed to succeed in open source projects. Skills needed in open source projects are usually learned formally by younger participants and informally by older participants. Open source projects also provide an opportunity to learn new skills by working on real software projects. Some open source projects have mentoring systems in place to share knowledge to new participants. Also more experienced developers can help newcomers get familiar with the project by splitting bigger features to small tasks that are easier to understand.

1. Introduction

Acquiring programming skills is difficult and according to Winslow (1996) it takes 10 years of experience to turn from a novice into an expert programmer. In this paper we will take a look on skills of participants in regards of open source development and projects. The research
questions for our research is what kind of skills are needed to succeed in open source projects and how those skills are acquired.

Through a narrative literature review we will determine what kind of skills are needed by participants to in order to succeed in open source projects. We will also explain how these skills are learned. Another side of the skills and open source combination is what participating in open source projects can teach to the participant. By answering to the research questions we hope to give an overview that would help someone who is interested in open source participation, but still has not participated in any way because of doubting their skills.

The single most popular project utilizing open source model is Linux kernel development project. The product of that project is the operating system kernel used by all distributions of this Unix-like operating system. This project is the largest, or at least one of the largest, longest-lived and successful projects in Open Source Software. (Walton, 2002.) As such an important project, it has been studied in many ways over the years and at least Hertel, Niedner and Herrmann (2003) and Kimmelmann (2013) have studied aspects related to skills and learning among Linux kernel developers.

These big open source projects utilize a lot of different people from different countries and cultural backgrounds. Contributors in a project can be roughly divided into groups depending on their role in the project but also depending on the skills they have. (Ducheneaut, 2005.) In some cases, open source projects can utilize mentoring where newcomers are first trained in the technical and social aspects of the project via email or other communication method. After initial training is complete, newcomer can easily start working on the project on their own. (Canfora, Di Penta, Oliveto & Panichella, 2012.)

Learning by doing and listening to other people sharing their knowledge and skills are seemingly important for OSS community (Mehra & Mokerjee, 2012; Ghosh, Glott, Krieger & Robles, 2002). According to a study about open source developers, 78.9 % joining developers of open source software communities want to learn and develop new skills and furthermore 67.2 % stay to share learned knowledge and skills. This helps other members of the community to learn new skills and improve on their existing expertise. (Ghosh et al, 2002.) As so big population of open source contributors are interested in learning and sharing knowledge, skills in relation to open source projects is a topic worth researching.

Open source software has been around in different formats since 1970. In that time there has been already done research relating to different aspects of open source projects (Brethhauer, 2001.) Lenghthy time period and popular projects ensure that there is a lot of literature related to open source and as such, literature review is a fitting research method for the topic. We also found research material to support this research. Skills and roles in projects are sometimes deemed the same thing or researchers haven’t gone in deeper with analysis of contributors’ skills other than their role in projects so there is still a need for more research.
The structure of the paper is the following: First we take a look at previous research in the field of study to determine what kind of skills are needed by participants to succeed in open source projects. Second, we present how such skills are learned by open source participants and how participants learn skills through open source projects. Then, in discussion chapter we reflect on all this found data together. Finally, in conclusions we present our formed ideas on the topic based on the earlier chapters.

2. Skills needed by contributors to succeed in a project

Kimmelmann (2013) studied competencies that are needed in open source projects through a Grounded Theory content analysis. This analysis was based on the data collected from her interviews with open source software developers, project managers and human resource managers from companies that work on open source software. The participant companies worked on software based on Linux Kernel and the people interviewed each had themselves years worth of open source experience. Found competencies were broken into three distinct categories: technical, social, and personal competencies. Technical competencies include technical knowledge required to work on the project in question. Social competencies mean interpersonal skills to support development in the project. Personal competencies include attitude, values and motivation relating to the project. (Kimmelmann, 2013.)

In addition to dividing found skills to technical, social and personal skills, Kimmelmann also categorized the skills based on different kinds of characteristics of open source software development. For example technical skill “implementation of feedback” (Kimmelmann, 2013, p. 207) is related to the characteristic of “tradition of public discussion” (Kimmelmann, 2013, p. 207) as seen in the Table 1. The table displays two important characteristics of open source software development and related competencies (or skills) which were discovered in Kimmelmann’s research. Additionally social skills like language skills and intercultural competencies relate to the characteristic of international community. In short, this categorizing presents what kind of skills are needed to succeed in different parts of work in open source projects. In the following table from Kimmelmann’s research is summarized the categories of characteristics of relevant competencies of software developers in open source software development. (Kimmelmann, 2013.)
Table 1. Extract of two characteristics from table of relevant competencies of successful OS software developers (Kimmelmann, 2013, p. 207).

The results indicate that social skills are very important. Social and communication skills are seen as crucial characteristics for an open source contributor regardless of the level of said contribution. Technical competency is of course a stable foundation for working in open source software but social competency is becoming even more important as more people with from diverse backgrounds start working in open source projects. Ability to take feedback from others and complying to community’s social rules are personal competencies that are important in this respect. Overall, well-balanced competency profile was brought up often as an admirable skillset of a successful developer. (Kimmelmann, 2013.)

In his book Fogel (2005) considers the ability to write clearly as the most important social skill a contributor can have, even as more important as programming itself. When communication is asynchronous it may be hard for a contributor with lacking communication skills to get others to pay attention to him/her. Contributor with good communication skills can persuade and coordinate people to do different tasks, thus affecting the whole project in a positive way. A good programmer with lacking communication skills can usually only do one thing at a time. Describing technical issues is one type of communication that is usually needed but more
important is empathizing with one’s audience. With good coordination, people can divide big parts of the project into smaller tasks, and this can then make contribution easier for some people. Fogel presents clear instructions on what to consider when writing and how to read other people’s messages correctly. Bergquist and Ljungberg (2001) note that these messaging skills are especially important for newcomers to get into the community. When introducing oneself by writing some kind of problem on the forums, one can use his personal situation in his/her message to try to establish relationship with the listeners. Importance of communication skills presented by Fogel and Bergquist and Ljungberg is also backed by the findings Kimmelmann (2013) presents in her study.

Communication skills are also acknowledged by Ducheneaut (2005) from another angle: good (and open) communication skills accelerates personal growth. He considers the process of a developer getting into a open-source community, in this case Python, by first monitoring other developers’ activity, filing bugs and suggestions and later on getting access to develop fixes or components by themselves. In this case, one needs to be acknowledged by the community and to develop an identity to actually become a developer or contributor. This is considered within the community as evolving, which is also one of the requirements to become a good developer in an open source project.

While considering social skills in general in software development, Matturro, Raschetti & Fontán (2015) found little different skills to be most valuable skills which a software engineering professional must have. In this study they were researching valuable soft skills, which include non-technical skills, people skills, social skills, generic competencies and human factors. Matturro et al. divided their interviews for team leaders and team members. As a result they found that both team leaders and team members consider following skills to be five most valuable soft skills for a software engineering professional: Analytic problem-solving, teamwork, commitment, responsibility, eagerness to learn and motivation. Their research was done interviewing 35 software engineering practitioners from 11 software companies in Uruguay. Kimmelmann (2013) found similar results in her research as can be seen from table 1.

Contributor success in open source projects is also linked with the team success. People with advanced skills in problem solving can identify complex problems and divide them into simpler tasks for people whose skills are not at so advanced a level yet. When people with different backgrounds join a project, these different skills and people within project should be taken into account by offering tasks that comply with their current skill level and previous experience. Basic skills of the contributors should allow them to absorb new technologies and knowledge generated elsewhere. (Giuri, P., Ploner, M., Rullani, F., & Torrisi, S. 2010.)

Projects backed by a bigger community can use mentoring as a way of attracting newcomers with not so good skill levels in programming but willing to learn. Although mentoring can put larger pressure on skilled developers, these projects with mentoring mentioned usually gain bigger developer pools and stronger contacts in the long run. Newcomers integrate better into the projects and asking questions from the newcomers rather than waiting them to make the
question yielded newcomers that were happier with their commitment with the project. Newcomers also thought that it was easier to understand the structure when it was explained by the members of the team rather than trying to read or ask an array of questions about the structure of the project. (Dagenais, Ossher, Bellamy, Robillard & De Vries, 2010.)

3. Acquiring skills

Barcomb, Grottke, Stauffert, Riehle and Jahn (2015) have studied how developers acquire skills needed in open source projects. Through an online questionnaire they acquired a data set from over 5300 open source project contributors about their skills and how they have acquired them. The skills used in the survey were partly based on those found by Kimmelmann (2013). These skills were largely related to communication, including evaluating the work of others, communicating with different target groups, documenting skills, making a clear argument and showing respect to others for their work. Included were also technical skills like writing reusable code, basic programming skills and understanding code written by other people.

Results showed that people who had contributed code to open source projects were more likely to have used informal learning methods to learn skills than those who had not contributed to open source. Also, younger people were more likely to have utilized formal learning than the older respondents, which on the other hand means that older respondents had more likely learned those skills through informal means. By informal learning the authors mean that the learning is done by his or her own. The learning can be done individually or collectively but without actual presence of an authorized instructor. By formal learning the authors mean education received from for example university, or an educational center where education leads to a certification. It seems that open source projects’ contributors tend to have used informal learning methods to learn a number of coding skills, which matches the hypothesis presented by the authors. It is clear that learning method depends on person’s learning style. Usually younger people learn skills from formal learning method while older people tend to use informal learning method to learn new skills. (Barcomb et al., 2015) In a statistical study conducted by Gnosh et al. (2002) 70 % of participating open source developers possess some sort of university degree.

Participating in open source can in itself improve developer's skills. Mehra and Mookerjee (2012) studied developing skills by participating in open source projects. Employers can gain benefit by encouraging their employees to participate in open source projects, as their skills improve and this leads to improvement in the internal software projects of the company. Formal training in the form of classroom training is a good way to learn basic skills, but more advanced skills may be better learned by doing, which is possible by participating in open source projects. Even if working on open source during work time leads to some loss of time spent on internal projects, the employee becomes more productive in the long run by learning new things.

Within communities, there are certain ways that developers can learn skills from other developers. One tangible way of learning is inspecting the existing source code. Simply
reviewing code can serve as a way of getting information, but it doesn't explain the decisions developers make when creating code. Utilizing comments can give some insight in the code, but it is only a compression of the steps taken. Helpfulness of comments depends greatly on the writer. If deeper insight is needed, CVS (Concurrent Versions System) can be used to get all related changes to a certain code file in chronological order. This gives learners more concise view of what has happened and what choices have been made especially when the errors are also visible in the system. (Hemetsberger & Reinhardt, 2006.)

The same learning by participation can be seen in the study of open source community as almost 8 out of 10 developers original reasons to join the community was to learn and develop new skills. Almost the same number of developers expect other members of community to share their skills and knowledge with them. (Ghosh et al., 2002.)

Developing technical skills can also be a part of the motivation in participating in open source projects. Hertel et al. (2003) have studied the motivation of software developers working on Linux kernel. They found out in their study that part of the motivation for contributors can be improving their career options by getting expertise in Linux kernel development. Hars and Ou (2001) also found similar motivations. Majority of respondents in their questionnaire reported that they were motivated by improving their programming skills. Also learning other skills, like English or teamwork, motivated people to participate.

According to Fogel (2005) social skills can be trained by writing a lot and communicating with people. If a person tends to ramble and bring up many different questions in his/her e-mails people could miss the crucial message or question that is being conveyed in a side sentence. This should be avoided in official interactions via email, while it is ok to exercise in unofficial communications in e.g. Internet Relay Chat (IRC) or other instant messaging formats. There are certain rules, which are used to maximise reading efficiency, on formatting and sending emails that are settled on by experienced open source developers. Composition, tone and structure are important points to focus on when creating messages for people you may never contact again, because your messages act as your business cards for those people. Inherently rude messages should not be transferred between people, but on the other hand, messages should not be interpreted as rude without a good reason. These rules also apply to other kinds of projects and communication between people.

Fogel (2005) also states that communicating properly with people in projects and generally gives a good impression of you and your skills. Practicing to write and writing good code doesn't correlate with each other so practicing coding doesn't make you a good communicator and vice versa. The exception is technical issues, where good coding skills will give your message more meaningful content. This type of technical communication is only a small part of project communication, but as Kimmelmann (2013) found in her study, it is an important skill in the crossroads of technical and communication skills.
Later on these skills gained from open source experience can be used to gain monetary or non-monetary rewards or even a job in the field. This is seen clearly in a statistical study conducted by Ghosh et al. (2002) where more than a half of developers interviewed had gained some sort of reward from open source experience. When asked about money only 4.3 % of people joined in the community to make money but then reasons for staying in the community making money was seen as a factor by 12.3 % of developers.

4. Discussion

The types of skills needed for successful participation in open source projects consist both of technical skills and other skills that could be summed up as people skills. The skills Mehra and Mookerjee discuss in their article consist of mainly technical skills, whereas Kimmelmann (2013) highlighted personal and social skills as the ones that can help to succeed in open source projects. Overall, as Kimmelmann also noted, the skillset of a successful open source participant is well rounded and has room for social and people skills, which is of the contrary to the myth of a socially incompetent nerd living in the basement.

It is interesting to note how Barcomb et al. (2015) and Mehra and Mookerjee (2012) come to the topic of skills and open source software development from different sides. Barcomb et al. studied how developers acquire skills needed in open source projects and Mehra and Mookerjee studied how developers learn skills by participating in open source projects. It’s like it’s two sides of the same coin, where some skills are necessary to be successful in open source participation, but the participation in itself also teaches the participant new things in different areas of expertise. Both are important as there is need to learn some skills, which can be for example certain programming language, before joining open source project. And after joining open source project it is important to keep yourself up to date in the field of the project so that the result wouldn’t be outdated.

On a high level, skills can be acquired by formal or informal methods. Certain skills related to the project should be acquired before participating, even though some skills can be learned or fortified from participation in open source development. Like Kimmelmann (2013) stated, technical competency is needed as a foundation for participation, but after that more advanced skills can be learned while working on an open source project. As Mehra and Mookerjee (2012) stated, classroom training is a good way to learn basic skills, but more advanced skills are learned better by doing, e.g. working on an actual project. This is an interesting thing to consider together with Barcomb and associates’ (2015) finding that younger people were more likely to have used formal learning methods and older people informal methods. Could it be, that younger people have the needed basic skills obtained through formal learning methods, like formal education from schools and universities, and older people have had the time to get more expertise through work or open source experience, both of which could be categorized as informal learning? In Ghosh et al. (2002) study containing a large online survey, 70 % of developers have some kind of university degree. While this study does not tell in which time
period the education was gained (was it before or after joining open source), it does tell that people with some level of university education are predominant in open source projects.

It would seem that learning the technical skills nowadays through internet is easier than ever and probably increasingly “informal”. When looking at the rapid movement speed of for example the web development scene, it would be very hard to keep up on the newest technologies if one would only study formally. Comments and logs of other developer activity give a way of seeing the decisions and ways of a developer when he/she tackles a problem (Hemetsberger & Reinhardt, 2006). Ghosh et al. note that exactly one third of open source developers in their study are software engineers in their professional life and 15.8 % were studying IT in some form followed by smaller groups of programmers and IT consultants. Other professionals were groups of 5 % or less. Professional work may or may not be related to open source software but dealing with software in professional work life, even though it’s proprietary, probably helps to keep skills honed for the tasks at hand in open source community.

If open source projects are used to learn advanced programming skills and new technologies, it would be interesting to know how the main developers and maintainers react to contributions that may not be of the best quality. As social and people skills were found to be very important for success in open source projects, one could assume that these situations would be handled well in successful projects through discussion and seen as an opportunity to teach and learn.

Unfortunately there are cases that suggest the contrary. Even if Linux is one of the most successful open source projects in history, Linus Torvalds has been continuously in headlines for his language and verbal abuse to contributors whose patches he has deemed subpar, last time as recently as October 2015\(^1\). This type of behavior does not demonstrate those skills that were deemed most important by Kimmelmann (2013) and Fogel (2005). As Linux is so vastly popular, Torvalds can probably get away with this behaviour, even if it causes discussion and offense. Newer and smaller projects would probably be hurt bad for this kind of communication and negative publicity. Even if that was not the case, on a personal level such rude feedback could be very demoralizing to the contributor. For a newcomer even small criticism can be difficult to handle, if the person is not accustomed to the way a certain group of people work, as Fogel (2005) and Bergquist and Ljungberg (2001) noted. Such claim is not hard to believe.

5. Conclusions

In this paper we reviewed findings from previous research related to needed skills in open source projects. We found out that most of the skills that are related to success in open source project participation are not technical, but rather social and communication skills are the most important ones. Technical skills are a foundation, but being able to work and co-operate with other project members are an important aspect in open source project participation. Writing is

\(^1\) http://lkml.iu.edu/hypermail/linux/kernel/1510.3/02866.html
an important skill, when most, if not all, communication happens asynchronously using text based messaging as a communication channel.

Skills needed in open source projects are acquired in many ways. Open source contributors are more likely to have used informal learning methods than those developers, who do not contribute to open source. Older people tend to have used more informal methods than younger people. This could be interpreted that younger people learn first from formal education and later keep on learning new things through other ways. Inside open source projects people have a lot of ways to learn from others, such as reading others’ code comments and/or inspecting their work in the project CVS. Also as the community has people with different experiences, more experienced members can help newcomers by dividing bigger parts of the project to small tasks which are easier to do or even make tutorials and guides on how to accomplish tasks in correct manner.

Participation to open source projects can also be motivated by the will to learn new things. Many people take part in projects to learn new things. The motivation can come from the will to improve their career options by learning some technology, willingness to improve programming skills (learning by doing) and to improve teamwork skills or even improve language skills. Working on open source projects is learning by doing and can lead to gains in productivity at work, even if employees use part of their work time on working on open source projects that are not directly related to their work at hand.

All in all, the findings of this study imply that there are possibilities for success in open source projects for people that have good people skills. We believe that these findings can help more people to participate in open source projects, as perhaps many have felt that their technical expertise is not strong enough. As our findings suggest that there is also a need for other kinds of skills in open source projects, perhaps some people will be encouraged to look for roles that fit the skills they have on the spectrum of people and communication.

As this study was done as a literature review, the biggest limitation of this study is that there was no empirical research done on this topic. As such, there is no empirical evidence on how the reported skills are demonstrated in the open source projects. This aspect was not really taken into account in the studies we found either.

Future research on this topic in could focus more on what are the benefits of different ways to learn different skills needed when working on open source software project. Another interesting research aspect would be how the reported importance of people skills is visible within open source project communities.

References


