Challenges faced by students in an Open Source Software Undergraduate Course

Dias Issa[1][0000−0002−9114−4610]

Computer Science Department, Nazarbayev University, Nur-Sultan 010000, Republic of Kazakhstan
dias.issa@nu.edu.kz

Abstract. The Open Source Software (OSS) development is gaining popularity from year to year, however, entering the OSS community still remains a challenging task. In this work, we describe challenges faced by a beginner OSS code-developer during the first contribution. Additionally, we analyze our experience and offer hints for potential newcomers. Whole work was done as the project of the Open Source Software undergraduate course at the Computer Department of Nazarbayev University.

Keywords: Open Source Software · Code developer · OSS Challenges

1 Introduction

The paper reports the student's experience in contributing to the open-source software project from GitHub gained during studying at the undergraduate Open Source Software course. We included a number of failed attempts to contribute to different projects alongside the successful one. Each project's background information and each contribution attempt were analyzed, while the acquired experience was digested into some useful advises for a newcomer. Finally, we drew several conclusions and noted about our future plans.

Our background includes experience in developing software systems and applications, and Machine Learning models for different platforms including mobile devices, personal computers, and sensor devices. Therefore, we decided to take the Open Source Software (OSS) course in the final semester in order to have a broader view of the field of Software Engineering. In detail, our motivations in taking OSS class were both intrinsic and extrinsic [13] [1]. We love coding and solving problems, so working on OSS project was a new experience for us. Additionally, we wanted to enhance our coding skills and learn novel things. Last but not least was that contribution to the project could improve the "experience" section of our resume.

Furthermore, our expectations from the course were both in theoretical and practical areas. We wanted to identify the main features of open source software development together with the reasons for its emergence and successful existence. At the same time, we wanted to get a practical experience of contribution to one of OSS projects. Finally, the internal structure of the OSS community was an aspect which we were curious to know.

The Open Source Software course has perfectly met our expectations due to its twofold structure. The first part was comprised of regular lectures about the history of OSS development, its features and tools. This material was taught using *Open Source: A Multidisciplinary Approach* authored by M. Mufatto [13] and *Open Source Software: A Survey from 10,000 Feet* authored by Androutsellis-Theotokis et al. [1]. The material was checked during examinations.

The second part was about contributing to OSS project and it was totally independent work. A student was responsible for searching an open source software project and for becoming its active contributor. This part was examined utilizing the milestone reports and presentations. The next sections describe some of these milestones and outcomes of the work.

The paper is structured in the following order: the project selection section with specifications for each option and their comparison, the contribution section, the learned lessons, and conclusion.

2 Project Selection

Despite the enthusiasm at the beginning of the course, the process of finding an appropriate project for further contribution was not an easy task. During the class, we changed the chosen projects two times due to different reasons that are described below. Therefore, project selection state is one of the most important steps during the open source software contribution process. Right decision at the beginning would help young contributor a lot in decreasing the amount of resistance during the development. By "resistance" we mean the generalized term for various factors hindering the process of contribution. In the next subsections, we describe our project selections, motivations for such selections as well as expectations, and reasons for withdrawal.

2.1 First Project: AVA

The first project was chosen after a long time spent on project search in GitHub. The variety of interesting projects there combined with our knowledge of several programming languages made this decision last so long.

General Project Description AVA [21] is the testing library on Node.js [18], so the main language of the project is JavaScript. The library allows executing tests in parallel, therefore, much faster. The community was medium active. According to AVA's statistics by February 4, the project had over 1.4k commits, with the last commit done on 18th of January 2019. During the previous month, there were done 15 pull requests, 9 issues were closed and 5 issues were opened. AVA was released 50 times and had 200 contributors. The project has very decent documentation translated to 8 other languages including English. There were 141 open issues, with 73 issues good for the first contribution by February 2019. AVA is under MIT license and has support on Twitter, Stack Overflow and Spectrum [21].

Justification of the decision This project seemed for us as an appropriate option for OSS class due to its detailed documentation, friendly community, and a variety of issues for beginners. Additionally, parallel programming was a new and interesting field for involvement. Subsequently, we tended to consider that the code developing for this project could lead to learning a great variety of new concepts. Finally, we consider AVA as a good project to enhance our skills in JavaScript.

Detailed role description and provisional activity plan As a developer for this project, we planned to work on issues tagged as "enhancement" and "bug". It was also desirable that these issues were tagged as "good for beginner". As the first stage of contribution, we chose to get familiar with the code of the project and its documentation for contributors. Then, depending on the understanding of the code, we were going to solve bugs or perform enhancements.

Activities and Reasons for withdrawal After a more detailed analysis of the project, it was found out that most of the "good for beginner" issues were outdated and were not solved for a long period of time. Moreover, most of them were in the field of documentation, while we wanted to solve developer issues [21]. Additionally, there almost did not appear new propositions with "good for beginner" tag, while other issues required skill and knowledge which, in our opinion, we could not provide. Figure 1 clearly demonstrates all these drawbacks. As a result, after several unsuccessful attempts to solve issues, we were compelled to search for a new project to contribute.

2.2 Second Project: Coala

The second project was chosen also after a long time spent on project search in GitHub. The first failure and aspiration for successful contribution forced us to treat this process carefully. Therefore, the project is analyzed and described in details in following subsections: the general description of the project, the reasons for selection, the role description and provisional activity plan, the governance structure of the project, its community structure, its architecture, and the justification for withdrawal of the project.

General Project Description The project is aimed to help programmers during revising their codes for bugs. Coala [5] [7] is a software designed for linting and fixing codes in a very broad range of languages. It is highly customizable, so the user could utilize Coala from his favorite coding environment (editor) and add extra Coala plugins whenever he needs. The main language of the source code of the project is Python.

Justification of the selection We had several reasons for choosing Coala for OSS class. Firstly, automatic code linting was an undiscovered and intriguing

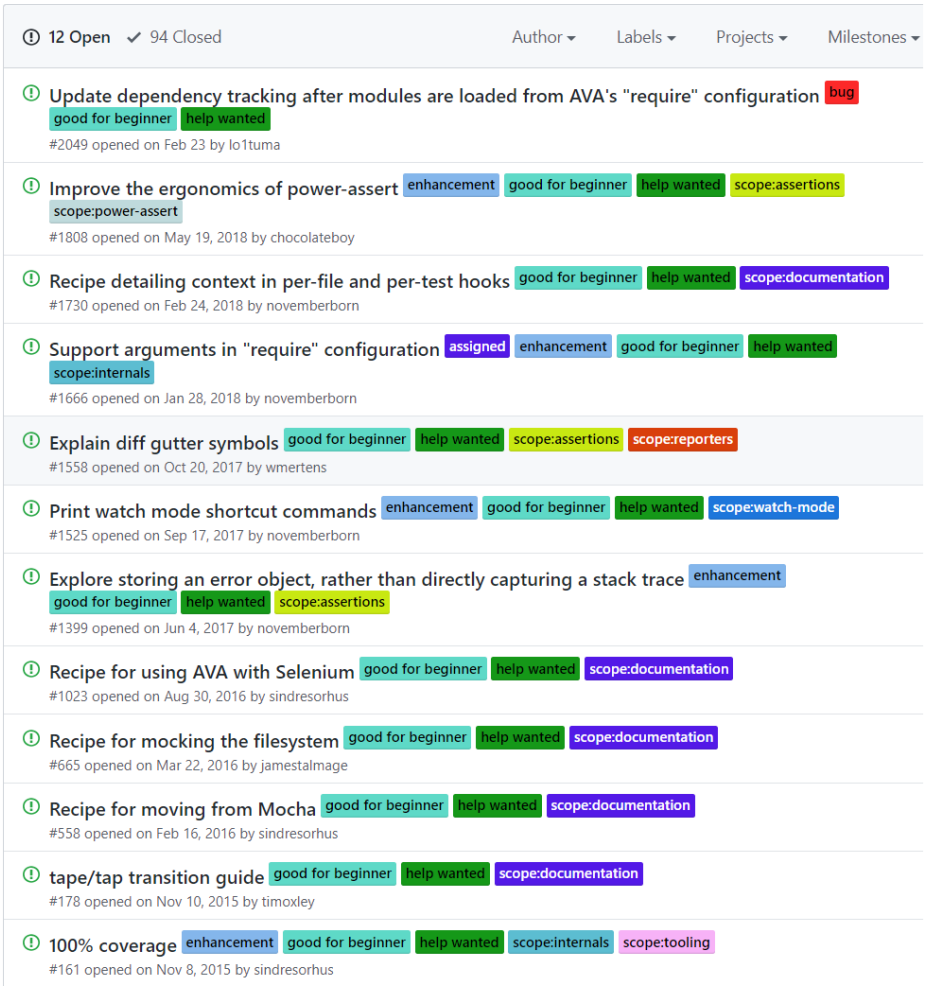

Fig. 1. Issues tagged as "good for beginner". AVA.

area for us. Secondly, the community is open for new members and helps them to make their first contributions. Thirdly, the project has good structured and well-written guide dedicated to newcomers. Fourthly, a wide range of issues with difficulty levels suitable for beginners. Finally, we liked their motto, the words of John F. Woods: "Always code as if the guy who ends up maintaining your code will be a violent psychopath who knows where you live." [5].

Detailed role description and provisional activity plan For this project, we planned to work on coding issues rather than documentation issues due to the reasons listed in the introduction of this paper. In the beginning, besides studying contributors guide of the project and its code, we decided to solve bugs because

this procedure seemed to us less complex than developing new features. After that, when we would gain experience, we were going to perform enhancements of the code or to develop new functionality. According to the plan, the first issues with which we wanted to start would have had tags "difficulty/newcomer" or "difficulty/low", because it is the part of the requirements of Coala community for new developers [6].

Governance structure We could not properly identify the governance structure of the project, however, we tend to think that it is monarchical.

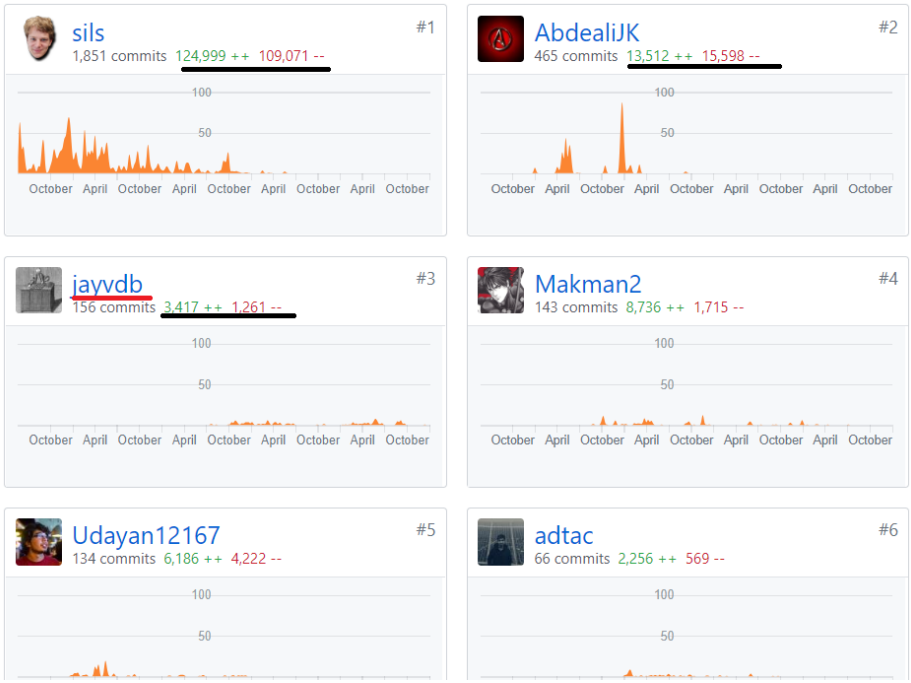

Fig. 2. Contributors. Coala.

The reasons for this claim are the following: from the Figure 2 one could clearly see that the owner of the project, sils [15], is the most valuable contributor with almost 7 times and 50 times larger contribution than the second and third valuable contributors. Also, according to the forum posts depicted on the Figure 3, he delegated some of his responsibilities to other players in his team, for example, he delegated to javdb [19] the work with the community. However, with the owner's much better knowledge of the project and his amount of contribution, we assert that the project owner's voice is the most valuable.

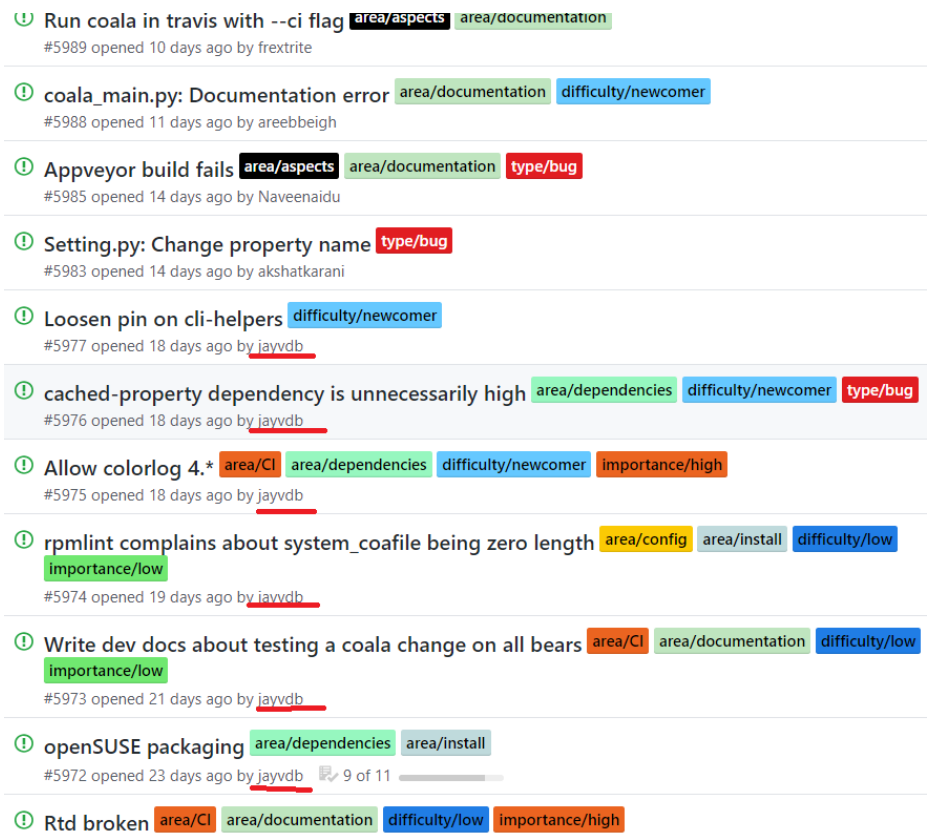

Fig. 3. Issue forum. Coala.

Community structure The community structure of Coala is hierarchical. This could be seen from their contribution guide [6], which states that in order to solve issues with a particular level of difficulty, you need to fix at least one issue and review at least one contribution that is one level of difficulty below. Additionally, in order to become a full developer of the project, you need to make a promotion request. Though Coala has such a strict structure that is typical for more proprietary software development, it resides under the strongest copyleft license: GNU Affero General Public License v3.0 [5]. This license obligates potential users to open the source code of possible derivatives of the project.

It could be said that such structure of Coala project is beneficial for the role of new developer of the project because a newbie has a clear understanding of which issues he or she should work and what to do next. At the same time, more experienced developers will not take out the potential issues that the new developer could solve due to the strict division of difficulty levels.

Architecture The general architecture of the project is modular. Basically, Coala consists of different modules each dedicated to a particular programming language. Most of the popular languages are supported right now. The project has working builds on Linux and MacOS, and it is planned to develop a working build for Windows because currently, it is failing [7]. Most of the issues of the project reside in the area of documentation and dependencies, some of the issues are bugs [5]. Also, due to such variety in supported programming languages, there are lots of issues connected with a particular language. Additionally, the project does not have a nice user interface and the work is performed using a command line [6]. This is acceptable for the current auditorium, however, better UI could attract more people who are not professionals (students, newbies in programming, etc.)

Activities and Interaction We started to search for an issue at the issue forum of Coala on GitHub [5]. There was a strong deficit of "newcomer" issues, so it was decided to monitor the forum for the appearance of new issues in this category. At the time when a new issue appeared, we were too hesitant in taking it. Therefore, because of this several minute long hesitancy, another contributor was assigned to the issue. After that, the sudden freeze of the project occurred, so we decided to communicate with the owner of the project in order to get any issues to work on. The owner of Coala, sils [15], stated that he is no longer engaged in the project and cannot help us much. Sils suggested us to check out "newcomer" issues on the GitHub forum. We also wrote an email to another active community member, jayvdb [19]. However, he did not answer the letter.

Reasons for withdrawal Coala was an ideal potential project for OSS class, the only drawback was that there were a large number of new contributors, so the number of issues was not enough for everyone. Nevertheless, the main reason for withdrawal was in project freeze, which occurred suddenly. Eventually, we were forced to search a new OSS project, because we needed to contribute as soon as possible due to deadlines of the OSS class schedule.

2.3 Third Project: Jarvis

The third and final project was chosen in a short amount of time, mostly spent on traversing GitHub. Jarvis [17] was listed as the third and the second project for potential contribution during two previous searches. Therefore, after two unsuccessful attempts, it was Jarvis' time to come on stage.

General Project Description Jarvis is an open source personal assistant for Linux and MacOS platforms, which works using command line interface. Additionally, it supports voice response. The assistant has such features as telling the weather, finding nearby places for having meal, etc. [17] The community is medium active. Jarvis has the following statistics statistics by March 26: the

project has over 800 commits, with the last commit done on the 26th of March. During the last month, there were done 11 pull requests, 7 issues were closed and 5 issues were opened. Jarvis has 74 contributors. The project is young and ambitious. Jarvis is under MIT license and has support on Gitter [17].

Justification of the selection We state that this project was a good option for the OSS course due to its young age, which gives an opportunity to find bugs and design new functionality. Additionally, personal assistance was a very interesting field to study and develop, especially for us, because we could apply our knowledge in Machine Learning in order to design new features for Jarvis. Also, we assert that we learned Python better during solving the issues because all code of the project is written in this language.

Detailed role description and provisional activity plan As a developer for this project, we planned to work on issues tagged as "bug". We also wanted to design new features for Jarvis, several of them we published at the forum [9] [8].

Governance structure We again could not properly identify the governance structure of the project due to the absence of a document with the rules of Jarvis' community [17]. Nevertheless, we tend to think that the governance structure is federal. The reason for this claim is the following: the owner of the project is only the 6th most valuable contributor of the project (Figure 4).

Moreover, according to Figure 4, appi147 [4], the core member of Jarvis, has the largest amount of code contributed. Additionally, the owner allowed to the other major contributors to work with the master branch [10]. Also, according to the forum posts, the new functionality could be confirmed by the core team of the project, not only by the owner [8]. Therefore, we state that the major contributors have the most valuable vote, which corresponds to the meritocracy that is the base of the federal model.

Community structure The community structure of Jarvis is not clear, because of its small size and anarchistic way of contributions. Subsequently, we assume that the community structure of this project is based on a fluid community organization model. The reasons for such assertion are the following: every member of the community, even newbies with no contribution, could offer new features and implement them, which was proved by our contribution [8]. This, in turn, leads to the second point, that membership in the community and its roles are fluid, so a developer could be an idea creator or a tester. Finally, according to the issue forum, the project evolution depends on the innovations offered by the members of the community [17].

Architecture The architecture of Jarvis is highly modular. Basically, Jarvis operates using a variety of plugins, which are independent of each other and

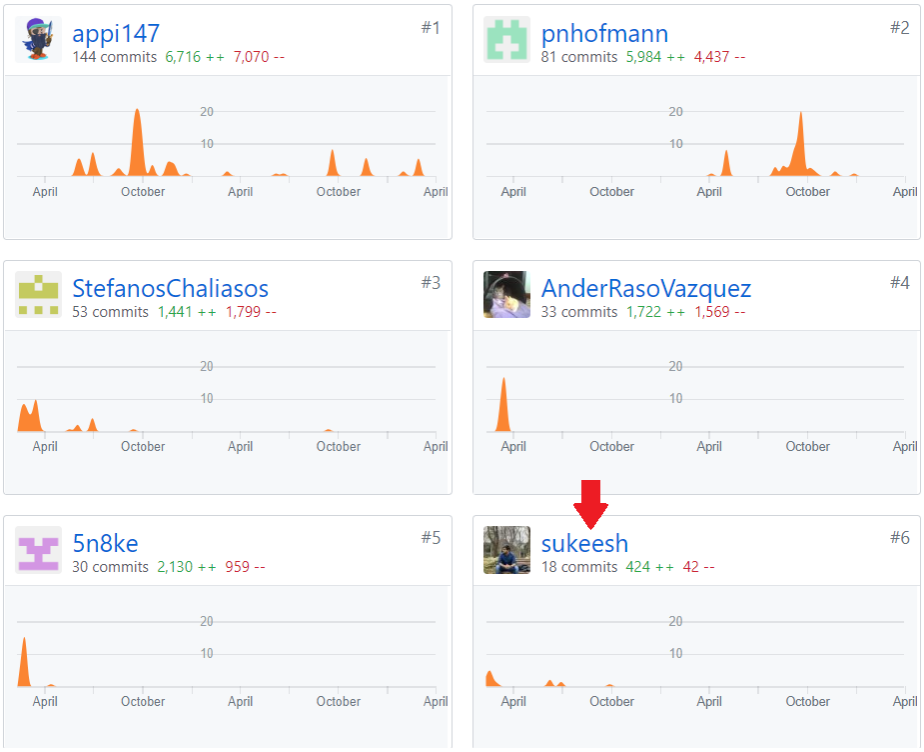

Fig. 4. Contributors and the highlighted owner of the project. Jarvis.

are responsible for different features of the personal assistant. The project has working builds only on MacOS and Linux platforms, while Windows is not supported at all and is not planned to be supported in the near future. Most of the issues of the project are located in the area of enhancements of the current functionality and in the area of implementation of new features. Jarvis has no GUI and operates through the command line and voice commands, which are not supported well [17]. Therefore, we claim that in order to increase the popularity of the project, Jarvis needs GUI.

2.4 Comparison of the projects

Table 1 shows the comparison of the three projects described in this section. The "starting complexity" entry of the table was assessed by the following criteria: the effort spent for understanding the code of the project, the amount of knowledge of the project needed to perform a contribution, time spent on learning the documentation to become familiar with the project and its order of contribution, and the effort spent for getting assigned for an issue. From the table, we can see that both Coala [7] and AVA [21] were founded in 2014, while Jarvis [17] is relatively young. Also, according to the number of contributors, AVA and Coala

Table 1. General comparison

	AVA [21]	Coala [5]	Jarvis [17]
Commits number	>1.4k	>4.4k	>800
Contributors	200	439	74
Releases	50	17	0
License	MIT	AGPL-3.0	MIT
Documentation	very decent	decent	poor
Time of 1st commit	Nov. 2014	Jul. 2014	Mar. 2017
Starting complexity	Hard	Medium Hard	Easy

projects are significantly more popular than Jarvis. At the same time, starting complexity at Jarvis is much lower than in the two older projects. Therefore, using the given data, it could be said that as a project gets older and more popular, the "resistance" for making the first contribution increases, such that even well-written documentation does not facilitate this procedure to the right degree. By resistance here we mean such hindrances for contribution as individual skills requirement, the complexity of issues, the requirements on the knowledge of the project to make the contribution, etc. This issue could be addressed using the following method utilized by some OSS projects [12]: new contributors start their development via pair coding sessions organized by other experienced members of the project, which in turn helps newbies to adapt, popularizes the project and leaves a good impression that motivates them to join its community.

To conclude, entering the OSS community for a new member is a complex procedure that is mostly dependent on the user's individual skills, project's age and its attitude towards new contributors.

3 Contribution

As said before, Jarvis was the most appropriate project for us to contribute. We contributed to the project in four different ways: fixed the bug, offered the new features, implemented the new features, and found the bug in the code of the project [8] [10] [9]. The subsections below describe each of the contributions.

3.1 Bug fix

We decided to start our first contribution to Jarvis with the simple task of fixing a bug. Fortunately, the project owner had found an "easy to fix" bug and offered it to community members [10]. We took into the account the last experience with the issue assignment in Coala project, and answered immediately, without any idea of solving the bug. Fortunately, we were assigned this bug and started working on it. Though the bug was easy and eventually was solved by us, we spent a very large amount of time fixing it. This happened mostly because of the absence of experience in contribution to OSS projects using GIT framework. Additionally, the time was also spent on multiple corrections of the submitted

code due to our limited knowledge of such OSS code writing rules as no empty lines with spaces, or space after comment sign, etc. Finally, we had done our pull request and it was merged with the master branch of the project [10].

3.2 New feature offers

Furthermore, we offered two new features for Jarvis. First of them was about adding the functionality of searching images using their description [9]. We even implemented the basic functionality using deep neural networks, however, the core team did not give an answer for this request. We assume that this happened due to the large complexity of implementation and usage of the offered feature.

Additionally, we offered a console game "Bulls and Cows", which was immediately accepted with strong enthusiasm [8]. We suppose that this occurred due to the simplicity of the idea and implementation process. Also, there could be other reasons for this. The nature of OSS encourages community members for voluntary contributions. The silence of the core members of Jarvis to our previous feature offer could made them feel uncomfortable due to possible "misconduct" on the subconscious level. Thus, unconscious desire to improve could affected the level of their enthusiasm.

Finally, we have a lot of ideas of new features that could be added to Jarvis, and which we could offer to the community.

3.3 New feature implementation

We implemented our first feature before it was offered, however, the feature was not accepted by the core team of Jarvis [9]. Therefore, we moved to create a plugin for our next offered feature - the game [8]. The programming part of the game was easy, so we wrote the code considerably faster than in bug fix case. However, the way of plugin creation and its insertion to Jarvis was not so clear, so most of the time was spent on these two tasks. Finally, after some attempts, we were able to submit our pull request, which passed all tests. The plugin was tested by the core members of Jarvis, and the pull request was merged with the master branch [8].

3.4 Bug investigation

Finally, we detected bugs occasionally while submitting our pull request. It was found out that some of the tests on Python 2.7 were not passing on the remote code checking server. However, according to the error report, the problem was not in the game plugin but in another plugin, which was part of the project, and, subsequently, was downloaded during cloning Jarvis. After the post on the forum, one member of the core team answered that he had merged his contribution with the master branch without checking it with lower Python version. Finally, he fixed the bug himself, which allowed us to finish our pull request [8].

4 Lessons learned

The project contribution procedure is not an easy process. Additionally, the right decision at the beginning could substantially affect the future performance of a contributor. This could be clearly seen from the evidence given in section 2.

From sections 2.1 and 2.2 we learned several lessons. Firstly, the contribution to OSS project requires some amount of courage. One should not stand in awe of potential failures because failure always could happen even with the most experienced programmers as clearly illustrates "bug investigation" section [8]. A newbie should be worried more about not contributing. It is better to try and fail rather than stay in silence. In the end, the only important thing in OSS contribution is an experience both technical and behavioral.

Secondly, it is better to search for projects, which have a considerable amount of issues dedicated to beginners. Also, these issues should be fresh enough, not problems that are not solved for months. The issue of finding a task to start was emphasized in works of Von Krogh et al. [20], Ben et al. [2] and Capiluppi and Michlmayr [3].

Finally, communicate with core community members in advance, do not wait for an appropriate moment. Core members could be busy and answer after a significant amount of time. In cases when no answer is received, it could be a sign for a newbie to immediately withdraw from a potential project. This situation was described in the work of Jensen et al. [11], where they note that the posts of newcomers which were replied, especially within 48 hours, had a positive correlation with their future project activity.

These lessons were learned by us at the time when we approached our third project selection - Jarvis [17]. Another lesson was learned from Jarvis's section: do not hesitate to offer something new. Even if one's offer is rejected, it gives him an experience, which could be used in the next attempt. Additionally, people do not like to reject, especially in the OSS community, where volunteering is encouraged. Therefore, one should propose his offer because even rejections eventually will lead to acceptance.

Furthermore, governance and community structures substantially affected the challenges we had to face. According to our own experience, the strict hierarchical structure could be beneficial for newbies due to its clear guidance offered by a project. This assertion is indirectly supported by the works of Park and Jensen [14], and Von Krogh et al. [20], where they claim that the community delegates the process of picking up the task for contribution to a newcomer [20], while the newcomer is not aware how to perform it [14]. However, on the other hand, the severe hierarchy limits the potential of a new contributor, it confines him in particular boundaries restricting from different possible ways of contribution. We encountered this phenomenon during interaction with Coala project's community [6], eventually ending with the contribution to Jarvis [17], which has fluid community structure.

Additionally, younger projects could be a better source of contribution than mature ones. The reasons are that younger projects are less complex and have more space for new ideas and creativity. At the same time, they offer a decent

opportunity of finding new bugs and designing new features. Studies of Capiluppi and Michlmayr [3] support this point by stating that new members of a project "tend to work more easily on new modules than on older ones". Moreover, they claim that new developers should be encouraged to create new ideas for a project. At the same time, mature projects are more stable, therefore, have fewer bugs. They tend to enhance existing features rather than creating new ones. Mature projects could be a good option for experienced developers, while young projects are better suited for beginners.

To sum up, we faced a number of challenges connected with the lack of knowledge, hesitancy to contribute, difficulty in getting feedback from the community, convincing its members, issues connected with the code design and its readability, etc. We discussed these issues with other members of the OSS course, a large amount of them faced similar problems. From one of them, we have discovered the excellent OSS project Gatsby [12], which has a very active and friendly community. This motivated us to not give up after failures and keep trying. Therefore, the last lesson learned and which could be suggested to a new OSS contributor: share the unsuccessful experience with the community, as well as, successful. In the first case, someone could help you to overcome challenges. At the same time, your failure would prevent others to make the same mistake. While in the second case, your success could encourage other community members and it could be a source for important lessons.

According to the categorization offered by Steinmacher et al. [16] the challenges described above cover 4 out of 5 barrier classes:

- Social Interaction
- Technical Hurdles
- Finding a Way to Start
- Newcomers' Previous Knowledge

This clearly indicates that the challenges described by Steinmacher et al. [16] still remain prevalent in the field of OSS. Despite the growth of open source movement, the quality of its organization stays the same, so that contributors face the same issues again and again. This leads to the idea that the organization of OSS should be enhanced in order to overcome the barriers. For example, by creating some common organizational criteria which are mandatory in order to a project be part of the OSS community, for instance, pair programming sessions as in case of Gatsby [12].

5 Conclusion

To conclude, in the paper we described our experience, as a new open source software programmer, about the entrance to the world of OSS development. According to the evidence given above, it could be said that contributing to OSS, especially for the first time, is a tricky procedure. However, it could be clearly seen, that a successful contribution motivates the contributor to contribute more. Therefore, the first experience is very significant during the development of open source software. The experience with Jarvis motivated us to

continue contributing to OSS projects in the future. We plan to enter Gatsby's community suggested by one of the members of the OSS course.

Acknowledgement

Thanks to Professor Antonio Cerone from the Department of Computer Science in Nazarbayev University for valuable discussions.

References

1. Androutsellis-Theotokis, S., Spinellis, D., Kechagia, M., Gousios, G., et al.: Open source software: A survey from 10,000 feet. Foundations and Trends® in Technology, Information and Operations Management **4**(3–4), 187–347 (2011)
2. Ben, X., Beijun, S., Weicheng, Y.: Mining developer contribution in open source software using visualization techniques. In: 2013 Third International Conference on Intelligent System Design and Engineering Applications. pp. 934–937. IEEE (2013)
3. Capiluppi, A., Michlmayr, M.: From the cathedral to the bazaar: An empirical study of the lifecycle of volunteer community projects. In: IFIP International Conference on Open Source Systems. pp. 31–44. Springer (2007)
4. Choudhary, A.: appi147 - overview (Apr 2019), https://github.com/appi147
5. Developers, T.C.: Coala github (Apr 2019), https://github.com/coala/coala
6. Developers, T.C.: Coala newcomers' guide (Apr 2019), https://api.coala.io/en/latest/Developers/Newcomers_Guide.html
7. Developers, T.C.: Coala website (Apr 2019), https://coala.io/
8. Issa, D.: New feature - game, https://github.com/sukeesh/Jarvis/issues/448
9. Issa, D.: New feature - image search using captions, https://github.com/sukeesh/Jarvis/issues/438
10. Issa, D.: Solution for two broken methods in movie.py, https://github.com/sukeesh/Jarvis/pull/447
11. Jensen, C., King, S., Kuechler, V.: Joining free/open source software communities: An analysis of newbies' first interactions on project mailing lists. In: 2011 44th Hawaii international conference on system sciences. pp. 1–10. IEEE (2011)
12. Mathews, K., Mathews, S.: gatsbyjs/gatsby (Apr 2019), https://github.com/gatsbyjs/gatsby
13. Moreno, M.: Open source: A multidisciplinary approach, vol. 10. World Scientific (2006)
14. Park, Y., Jensen, C.: Beyond pretty pictures: Examining the benefits of code visualization for open source newcomers. In: 2009 5th IEEE International Workshop on Visualizing Software for Understanding and Analysis. pp. 3–10. IEEE (2009)
15. Schuirmann, L.: sils - overview (Apr 2019), https://github.com/sils
16. Steinmacher, I., Silva, M.A.G., Gerosa, M.A., Redmiles, D.F.: A systematic literature review on the barriers faced by newcomers to open source software projects. Information and Software Technology **59**, 67–85 (2015)
17. Sukeesh: sukeesh/jarvis (Apr 2019), https://github.com/sukeesh/Jarvis
18. Tilkov, S., Vinoski, S.: Node. js: Using javascript to build high-performance network programs. IEEE Internet Computing **14**(6), 80–83 (2010)
19. Vandenberg, J.: jayvdb - overview (Apr 2019), https://github.com/jayvdb

20. Von Krogh, G., Spaeth, S., Lakhani, K.R.: Community, joining, and specialization in open source software innovation: a case study. Research policy **32**(7), 1217–1241 (2003)
21. Wubben, M., Sorhus, S., Demedes, V.: Avajs/ava (Apr 2019), https://github.com/avajs/ava