International Journal of Advance Research in Science and Engineering
Volume No.06, Issue No. 11, November 2017
www.ijarse.com
IJARSE
ISSN: 2319-8354

Analysis of Community Dynamics in Open Source Software Projects

Rajdeep Kaur[1], Dr. Kuljit Kaur Chahal[2]

[1,2]Department of Computer Science, Guru Nanak Dev University, Amritsar (India)

ABSTRACT

Volunteer members play a significant role in survival of Open Source Software projects (OSS).An individual joinsopen source community through the socialization process known as "Onion Model". Every new member joins as a passive user, participates in mailing list discussions and reads messages.Later he playsthe role of bug reporter, bug fixer, or participates in more technical activities. After getting substantial knowledge about the functioning ofthe system and building a reputation among community members, he gets intothe core group. Hence, he obtains theright to modify source code. All newcomersface many difficulties when they place their first participation in an OSS project. Difficulties like social and technical problems, absence of community supportinterrupt the first contribution of a beginner, slow down the process of participation and may force him to leave the project. Moreover, there are many factors which motivate him to jointhe project and influence his retention behavior in the project.The interior motives like identity creation and self-enjoyment play a significant role in developer participation. However, exterior motives like career concerns, need for the software solution, improving programming skill highly influence the participation of developers in OSS projects. The objective of this study is to understand the, joining process, community organization, survival analysis, abandonment and obstacles, motivation, retentionand attractions of developers in OSS projects.

Keywords:abandonment, attractiveness, hindering factors, joiningprocess, motivation, retention.

I.INTRODUCTION

Open Source Software (OSS) is anexample of global software development.OSS work in a distributed environment. In OSS, source code is accessible to everyone and users of OSS can download, modify, distribute copies, and have right to make improvements in versions of the Software. Therefore, manycompanies' shows interest in OSS projects by allocating staff to contribute in OSS projects, and that is a part of the company'sstrategy [1]."GNOME(GNU Network Object Model Environment) desktop environment and Free BSD(Berkeley Software Distribution) operating systems are examples of open source software"[2].Now days OSS is gaining popularity in software engineering communities and public. Survival of OSS projects mostly relies on resources provided by communities for developers and mutual labors of users. During the development of an OSS project, responsibilities of members change dynamically according to the job performed by members in the project and the community. Jobs are not allocated to members in advanceand each member achievesspecific role when heinteract with other members, selects the job according to his interest and capability. Themost determined member becomes a member of developer group. A new member starts with

participation on mailing list, reads the messages on the mailing list, plays role of reader and does some documentation tasks in the initial stage to obtain information about the system functionality. After that he playsthe role of bug reporter, for which technical skillmaynot be required. After receiving complete understanding of the system, he performs some small bug fixing tasks. Through continuousparticipation in the project and building social status among community members, he become member of the core group andobtains the right to make changes in source code. The entire process is known as "Joining process"[3].JoiningProcess plays a substantial role in success of OSS project. This process starts with participation on a mailing list discussion where negotiations related to project design and development take place. After continuous participation in technical discussions, newcomers win faith of other members and selected for the job of developer and obtain the right to make changes in the source code repository[4].In this paper, we study joining process, community organization, survival analysisand abandonment of OSS projects and different factors which influence the motivations, retention and attraction of developers and the difficulties of newcomers that delay their first participation.

II. STATE OF THE ART

This section attempts to present the state of the art in community dynamics in OSS projects as discussed below:

2.1 Joining Process

The Joining process is a collection of activities and successive steps that are followed by beginner to become a member of the project. The joining process in open source projects starts with a contribution on a mailing list where public discussion related to engineering and design of the project takes place. During software development duties of members change dynamically. For example, every new member starts his ride by giving first contribution on mailing list discussions, then he play the role of bug reporter that does not require any technical knowledge. After obtaining information about the functioning of system and community he becomes bug fixer and access the right to modify source code. Continuous involvement in mailing list discussion and wining trust of other members of community through the presentation of their technical skill on mailing list help a newcomer to attain the status of developer and obtain rights to make changes in the source code repository. Patch submission plays an important role in selection of member for job of developer. If a member has submitted patch previously, then the likelihoods of members to become developer increase. This process is referred to as joining process and socialization [4],[5].Israel herraiz et al. [1] studied duration and basic characteristics of developer joining process in the GNOME project. The authors have identified two categories of developers with different joining patterns and relate these patterns to different behaviors of volunteer and hired developers. Volunteer developersfollow onion model to join the project and their joining process is slow and gradual. The duration of this process is two or three years. Onion model is not followed by hired developers. Hired developerscan participate in any activity simultaneously and joining process took less time as compared to volunteer developers. The joining procedure of hired a developer took less than one year to complete. AftabIqbal.[3]studied the contribution of members in term of bug reporting, bug fixing and social relationships

with other members of community on mailing lists in an OSS project of community. The authors have examined the contribution made by members who attained the position of developer and theproportion of contribution made by them before and after becoming a member of developer group.Christian Bird et al.[5] identified that how the probability of achievingthe role of developer differs according to tenure in a community and analyses the role of social status and technical skill in the attainment of developer status and focus on theroleof patch submission inthe attainment of developer position.

2.2 Community Structure

OSS communities don't have a strict layered organization. Influence of members on system and community vary according to the job performed by them. The tasks are not allocated to members in advance and they select tasks according to their skill and personal interest [7].Many researchers have examined the joining process of OSS projects. The most widely used model that represents the organizational structure of OSS is "onion model". The Onion model describes which kind of role is assigned to members of OSS communities. The core members which are closer to the center have high impact in the community. Only few a members belong to the core group. Nearby core member group is a huge number of contributors who play the role of bug reporter, bug fixer and take part in mailing list discussions [3]. *Project Leader*initiates the project and provides supervision in project[7].*Core Member*contributes in development work. They continue to work on the project for a long period and play a crucial role in growth and development of the project. Usually the size of the core member group is very small and surrounded by a large number of contributors who report bugs, fix bugs and participate actively in mailing list discussions. Case study of different projects shows that core group composed of 3 to 15 members[3], [8].*Active Developer*hasthe right to make changes inthe source code repository and they are technical skilled people[9].*Peripheral developer*Principal job of peripheral developer isto add advanced features tothe software in asymmetricalmode[10].*Bug Fixer*is responsible for fixing bugs that are discovered by him and reported by other members of the organization. He studies the source code to identify the part of the source code where bugs occurred [3].*Active users*identify and report bugs, but don't participate in bug fixing [10].*Passive users* use the system in a similar way as common users use the software. They don't interact with the development community and have a majority inthe community members. For example, 99% users of apache are passive users [7], [8].

1.Table.: Community Roles in Open Source Software Projects

Roles	Description
Project Leader	Start the project and provide guidance related to project
Core developer	Help in development activities and give a long term contribution in the project. Small in size and have high impact in the community
Active Developer	People with technical skill and authorized to make changes in source code repositories

Peripheral developer	Contribute new features to the software
Bug Fixer	Participate in bug fixing activities and discover defects in source code.
Active User	Report bugs, but don't fix them
Passive User	Uses the system similar way as ordinary user utilizes the software

2.3 Survival analysis

Survival analysis is a statistical method that is extensively used in medical science and biological applications. Previously it was used by insurance companies to calculate the insurance premium [11]. Survival analysis model the time it takes for the occurrence of a specific event. This method is suitable for right-censored data, where it is not sure if an event has occurred or not[12].Bin Lin et al. [13] applied survival analysis on five global open source projects to study the influence of time to join the project, the rate of maintaining files, activity type and job type on retention behavior of developers. The finding shows that developers who start contribution to project earlier their likelihoods of survival are more as compared to other developers. The developer whose main task is to modify file survive longer than developers who make files and developers whose chief job is coding have high probability to survive than developers who works on documentation.Goeminne et al. [12] studied five Java database frameworks of 3,707 GitHub Java projects and applied survival analysis to examine the co-occurrence of database framework,the survival of database framework usage in the project and impact of adoption of a database framework on the existing framework.Scanniello [14] used Kaplan Meier measure that is widely used survival function to study the effect of dead code on the development of software systems. Results indicate that two out of five project developers do not introduce unused code. Sentas et al. [15] used survival analysis to examine the distribution of project duration and factors that influence it.

2.4 Abandonment

Ewusi Mensah et al. [16]investigated the behavior of IS (Information System) project abandonment in diverse organizations. The authors observe that all IS project's abandonment become an issue. Cost overrun, schedule slippage, technical inadequacies and behavior, political or organizational issues contribute in project abandonment. The findings show that organizational behavior and political issues have a high influence on abandonment of the project and financial and technical factors have less influence on project abandonment.Bao et al.[17]examined the activity data of developers and applied data mining approach to predict the turnover of software developers in close source projects. The authors observed after joining company for one year, whether developers shall abandon the company. The results indicate that there is significant correlation between several factors and developer turnover and the key factors which cause developer turnover.Sharma et al. [18] used hierarchical linear modeling technique to empirically analyze the factors affecting developer turnover and how these factors are different in developer and project. They studied the factors from individual and project perspective to understand the developer turnover in OSS projects. The analysis illustrates that earlier activities,

developer role, project age and project size is predictors of turnover. The findings show that turnover rates are different in source forge projects and such variance is due to project age and size.Bin Lin et al. [2] analyzed five open source projects to examine the effect of time of joining, rate of maintaining files, action type and job type on developers who abandon the project. The authors identify subgroups of developers who have a high probability to leave the project. They find that the developers who join earlier have high likelihoods to stay in the project. The developer who performs modification in files stays longer than developers who make files and the developer whose main task is coding stay longer as compared to developers who perform documentation work.Steinmacher et al. [19]hasStudied the effectof the lack of answers, answer courtesy and effectiveness and author of the response on abandonment of newcomers. The authorsdiscoveredthat a developer decision to abandon the project influenced by the writer of the answer and kind of response received. The lack of answer has no effect on newcomer decision to leave the project.

2.5 Developer Motivations, Retention, Attractivenessand Hindering factors

2.5.1 Motivation

Motivation indicates the powers that motivate the developers to give a contribution in the project. Every developer has different motives to start contribution that evolve with time and constantparticipation in the project[20].Shah. [21]studied the motivations of developer by collecting data through interviews, reading mail archives and documentation of two open source projects. The authors divided contributors in two categories: *need-driven* and *hobbyists*. Need-driven participants have different purposes to contribute in theproject like Reciprocity, need of particular features, desire to add own source code, career concerns, receiving feedback, future enhancements. Hobbyist's participants contribute in coding for entertainment and enjoyment. Lakhani et al. [22]haveanalyzed efforts and motivations of individuals that encourage them to participate in OSS projects. The authors studied684 software developers of 287 open source projects to understand the motivation behind participation. They discovered that external motivations like better job and career development are the key derivers behind the effort. The results show that enjoyment, difficulties in writing code, and improving programming abilityare topmost motivations for participation.Har et al. [23]havesurveyed the data of 389 developers involved in Free OSS projects to understand the internal and external motives of developers which inspire them for participation. The findings reveal that internal motives like identity creation and enjoyment contribute in developer participation, but external motives like building human capital and personal needs for software solution have great impact on developer participation.

2.5.2 Retention

Retention is capability of project that support newcomer onboarding and encourage developers to continue participation in the project.Steinmacher et al. [19]haveexamined the first interaction of beginners on a project to verify whether the lack of answers, courteous and supportive answers and author who respond newcomer have any effect on retention of developers. Theycollected five year data ofthe developer mailing list and issue manager (jira) conversation of the Hadoop common project and identified the causes of dropout of newcomers based on their first interaction in the project. Throughout the study period less than 20% newcomers continue

participation for a long time. The findings show that authors who send reply and type of answer received influencethe decision of beginner to stay or leave the project, but absence of response does not have much influence on choice to leave or stay in the project. Through observing mailing list discussions the authors find that the newbies who are willing to participate in the project,but getreplyfrom another newcomer have high probability to abandon the project. Results show that the newcomer retention rate is small which is 18% in mailing list and 13% in issue manager. The authors conducted survey of dropout to understand the causes of developer dropout and find that adversemessages, unsatisfactory responses and experience of respondents influence the decision of newcomers.Khan et al. [24]havestudied the influence of adoption of new development practices and tools like TRAVIS CL (Continuous Integration) - a popular service provider on developer attraction and retention. The authors investigated the association among adoption of TRAVIS CL and developer attraction and retention. They studied 217 projects of the MSR challenge dataset. The Results indicate that attraction and retention of developer are higher before adoption of TRAVIS CL and decline after the introduction of TRAVIS CL. They applied Wilcoxon signed rank method to check varianceamong pre and post TRAVIS CL values.Vishal et al.[25]haveexamined the influence of software complexity and modularity on softwarequalityand retention of developers. The authors identify that high modularity of software increase the retention of developers and reduce the time to fix bugs and software complexity result into a drop in developer retention and require more time to fix bugs. The association between modularity and complexity has investigated by extracting data of 70 different projects of SourceForge repository. The results demonstrate that the complexity and modularity of software have a high effect on developer retention and the amount of bugs reportedin a software project.Fang et al. [26] qualitatively investigated the relationship between continuous participation and situated learning and identity creation. They find that situated learning and identity construction associated with sustained participation and play significant role in retention of developers.Sen et al. [27]conducted exploratory study to examine the factors which contribute in survival of open source software projects that estimated by number of subscribers and developers working on the project. The authors identify that OSS projects which develop software for Window/Unix/Linux operating system and use C language for software development influence more developers as compared to projects without this feature. The OSS projects where license rules are not hard having more developers as compared to projects where license have more restrictions.Lee et al. [28]studied impressions, motivations and difficulties of OTCs (One Time Code Contributors). Analyzed the OTCs of 23 well-known open source projects .The authors discovered that OTCs has a positive opinion about their FLOSS projects and they have several motives for participating in the project. The Most OTC's motive behind participation is to fix bugs that interrupt their work. They are not willing to become a long term contributor of the project. Moreover, based on findings, suggestions are provided to Open Source projects to increase the retention of OTCs.Baldwin et al. [29]haveanalyzed the correlation among the architecture of codebase and Free Open source Software development process. The authors identify that modularity of code and option values are attributes of the project, which strongly influence the retention behavior of the developer.

2.5.3 Attractiveness

Attractiveness is the ability of the project to attract new developers. Researchers studied the interaction of developer from three different points of view. They studied developer attractions from individual, group and project perspective.Khan et al. [24]haveinvestigated the effect of adoption of new development practices and tools like TRAVIS CL (Continuous Integration) – a well-knownservice provider on developer attraction and retention. The authorsstudied 217 projects of MSR challenge dataset to analyze the correlation among adoption of TRAVIS CL and developer attraction and retention.The Results indicate that attraction and retention of developer are higher before adoption of TRAVIS CL and decline after the introduction of TRAVIS CL.Yamashita et al.[30] used two metrics magnet and sticky to estimate the attraction and retention of developers in Open Source Software Projects. Magnet measure the amount of new developers attracted to project and sticky estimate the retention of old developers. Findings reveal that 23% of developers stay in the same project. Large project attracts more developers and developers stay longer in large projects.Meirelles et al. [31] have examined 6773 Open Source Software Projects of SourceForgenet to study the correlation among source code metrics and attractiveness and analyzed the effect of code size and structural complexity on the attractiveness of the project. The authors discover that source code measure LOC (line of code) has a great impact on size of users in the project and structural complexity negatively influences the project attractiveness.Andreas Schilling et al. [32] investigate the influence of reputation of developers among user and developers of the project community and outside community on project capability to attract new developers and how relationship with other members supports project to attract new developers. The authors have built Social Resource Theory to identify that how association among high reputation developers is necessary to attain financial outcomes.Stewart et al. [33] have studied the impact of ideology on team efficiency of Open Source Software Development.The authors examined two types of team effectiveness: the attraction and retention of developer input and creation of project output. The authors developed the framework of the principles of OSS ideology and identify that some mutual ideological principles positively influence the team effectiveness of OSS projects and help in enhancement of trust and quality. Apart from this some ideological ideas negatively influence the effectiveness.

2.5.4 Hindering Factors

Hinderingfactors are difficulties encountered by novices that hinder their first contribution when they eager to contribute in the project. These factors interrupt their contribution and force them to discontinue participation in the project.Steinmacher et al.[34]studied the social barriers of newcomers that become an issue in their first contribution. The authors proposed theoretical model comprise 58 hurdles in which 13 related to social barriers and discovered these factors through qualitative analysis of data that is obtained from responses of OSS contributors, students who involved in OSS projects and interviewing 36 developers of dissimilar projects . They identify that many hindering factors affect the contribution of beginners such as technical and social issues, lack of knowledge, absence of community support and problem faced during starting period. These issues slow down the contribution of newcomer. Study Identify that 28.3% newcomer halt participation due to hesitation. Mentoring alsoinfluence the participation of new member when they don’t receive correct guidance

from community, they decide to leave the project. Moreover, cultural differences also effect the participation of new members.Carlos Jensen et al. [35] analyzed eight mailing lists of four free open source projects to study the impact of gender, nationality, civility, helpfulness and timeliness of reply on future participation of newcomers. They identify that 80% newcomer get response and timely response,within 48 hours, positively influence the future participation of a beginner.

III. CONCLUSIONS

In this paper,we studied community structure, the sequential processthrough which newcomer joins Open Source Software Projects,Structure of the community, survival analysis, project abandonment, developers key motives behind participation and retentions and hindering factors that interrupt their first participation, influence their retention in project and force them to leave the project. Community members should provide guidance to newcomer to eliminate contribution barriers. Therefore more research is required in this direction. The study shows that many new members are not interested in joining OSS projects.Their main motive for joining is to satisfy their own needs, clear doubt, receive a correct answer to their questions and they don't stay for long time in the project.The developers have many internal and external motives for participation. Self-enjoyment and identity creation are an interior motivation of the developer that plays key role in developer participation. However, exterior motivations like career concerns, improving programming skill and future advancement have high impact on developer participation.There are numerous reasons behind the dropout of developerssuch as when they are not satisfied by response of community members, inadequate responses and lack of community support. Moreover,the complexity and modularity of source code, situated learning and identity creation, code modularity and option values, license constrains influence the retention behaviors of developers.

REFRENCES

[1] Herraiz, I., Robles, G., Amor, J. J., Romera, T., & González Barahona, J. M. (2006, May). The processes of joining in global distributed software projects. In Proceedings of the 2006 international workshop on Global software development for the practitioner(pp. 27-33). ACM.

[2] Lin, B., Robles, G., & Serebrenik, A. (2017, May). Developer turnover in global, industrial open source projects: insights from applying survival analysis. In Proceedings of the 12th International Conference on Global Software Engineering (pp. 66-75). IEEE Press.

[3] Iqbal, A. (2014). Understanding Contributor to Developer Turnover Patterns in OSS Projects: A Case Study of Apache Projects. ISRN Software Engineering, 2014.

[4] Bird, C., Gourley, A., Devanbu, P., Swaminathan, A., & Hsu, G. Quantitative Study of Open Source Immigration.

[5] Bird, C., Gourley, A., Devanbu, P., Swaminathan, A., & Hsu, G. (2007, May). Open borders? Immigration in open source projects. In Proceedings of the Fourth International Workshop on Mining Software Repositories (p. 6). IEEE Computer Society.

[6]Steinmacher, I., Conte, T., Gerosa, M. A., & Redmiles, D. (2015, February). Social barriers faced by newcomers placing their first contribution in open source software projects. In Proceedings of the 18th ACM conference on Computer supported cooperative work & social computing (pp. 1379-1392). ACM.

[7] Nakakoji, K., Yamamoto, Y., Nishinaka, Y., Kishida, K., & Ye, Y. (2002, May). Evolution patterns of open-source software systems and communities. In Proceedings of the international workshop on Principles of software evolution(pp. 76-85). ACM.

[8] Bosu, A. (2012, March). Mining repositories to reveal the community structures of Open Source Software projects. In Proceedings of the 50th Annual Southeast Regional Conference (pp. 397-398). ACM.

[9] Jergensen, C., Sarma, A., & Wagstrom, P. (2011, September). The onion patch: migration in open source ecosystems. In Proceedings of the 19th ACM SIGSOFT symposium and the 13th European conference on Foundations of software engineering (pp. 70-80). ACM.

[10] Jensen, C., King, S., & Kuechler, V. (2011, January). Joining free/open source software communities: An analysis of newbies' first interactions on project mailing lists. In System Sciences (HICSS), 2011 44th Hawaii International Conference on (pp. 1-10). IEEE.

[11] Sentas, P., & Angelis, L. (2005, September). Survival analysis for the duration of software projects. In Software Metrics, 2005. 11th IEEE International Symposium (pp. 10-pp). IEEE.

[12] Goeminne, M., & Mens, T. (2015, September). Towards a survival analysis of database framework usage in Java projects. In Software Maintenance and Evolution (ICSME), 2015 IEEE International Conference on (pp. 551-555). IEEE.

[13] Lin, B., Robles, G., & Serebrenik, A. (2017, May). Developer turnover in global, industrial open source projects: insights from applying survival analysis. In Proceedings of the 12th International Conference on Global Software Engineering (pp. 66-75). IEEE Press

[14] Scanniello, G. (2011, September). Source code survival with the Kaplan Meier. In Software Maintenance (ICSM), 2011 27th IEEE International Conference on (pp. 524-527). IEEE

[15] Sentas, P., & Angelis, L. (2005, September). Survival analysis for the duration of software projects. In Software Metrics, 2005. 11th IEEE International Symposium (pp. 10-pp). IEEE.

[16]Ewusi-Mensah, K., & Przasnyski, Z. H. (1991). On information systems project abandonment: an exploratory study of organizational practices. MIS quarterly, 67-86.

[17] Bao, L., Xing, Z., Xia, X., Lo, D., & Li, S. (2017, May). Who will leave the company?: a large-scale industry study of developer turnover by mining monthly work report. In Proceedings of the 14th International Conference on Mining Software Repositories (pp. 170-181). IEEE Press.

[18] Sharma, P., Hulland, J., & Daniel, S. (2012). Examining turnover in open source software projects using logistic hierarchical linear modeling approach. Open Source Systems: Long-Term Sustainability, 331-337.

[19] Steinmacher, I., Wiese, I., Chaves, A. P., & Gerosa, M. A. (2013, May). Why do newcomers abandon open source software projects? In Cooperative and Human Aspects of Software Engineering (CHASE), 2013 6th International Workshop on (pp. 25-32). IEEE.

[20] Steinmacher, I., Gerosa, M. A., & Redmiles, D. (2014). Attracting, onboarding, and retaining newcomer developers in open source software projects. In Workshop on Global Software Development in a CSCW Perspective.

[21] Shah, S. K. (2006). Motivation, governance, and the viability of hybrid forms in open source software development. Management science, 52(7), 1000-1014.

[22] Lakhani, K. R., & Wolf, R. G. (2005). Why hackers do what they do: Understanding motivation and effort in free/open source software projects. Perspectives on free and open source software, 1, 3-22.

[23] Hars, A., & Ou, S. (2001, January). Working for free? Motivations of participating in open source projects. In System Sciences, 2001. Proceedings of the 34th Annual Hawaii International Conference on (pp. 9-pp). IEEE.

[24] Gupta, Y., Khan, Y., Gallaba, K., & McIntosh, S. (2017, May). The impact of the adoption of continuous integration on developer attraction and retention. In Proceedings of the 14th International Conference on Mining Software Repositories (pp. 491-494). IEEE Press.

[25] Midha, V., & Palvia, P. (2007). Retention and quality in open source software projects. AMCIS 2007 Proceedings, 25.

[26] Fang, Y., & Neufeld, D. (2009). Understanding sustained participation in open source software projects. Journal of Management Information Systems, 25(4), 9-50.

[27] Sen, R., Singh, S. S., & Borle, S. (2012). Open source software success: Measures and analysis. Decision Support Systems, 52(2), 364-372.

[28] Lee, A., Carver, J. C., & Bosu, A. (2017, May). Understanding the impressions, motivations, and barriers of one time code contributors to FLOSS projects: a survey. In Proceedings of the 39th International Conference on Software Engineering (pp. 187-197). IEEE Press.

[29] Baldwin, C. Y., & Clark, K. B. (2006). The architecture of participation: Does code architecture mitigate free riding in the open source development model?Management Science, 52(7), 1116-1127.

[30] Yamashita, K., Kamei, Y., McIntosh, S., Hassan, A. E., & Ubayashi, N. (2016). Magnet or sticky? Measuring project characteristics from the perspective of developer attraction and retention. Journal of Information Processing, 24(2), 339-348.

[31] Meirelles, P., Santos Jr, C., Miranda, J., Kon, F., Terceiro, A., & Chavez, C. (2010, September). A study of the relationships between source code metrics and attractiveness in free software projects. In Software Engineering (SBES), 2010 Brazilian Symposium on (pp. 11-20). IEEE.

[32] Schilling, A., Laumer, S., & Weitzel, T. (2014, May). Stars matter: how FLOSS developers' reputation affects the attraction of new developers. In Proceedings of the 52nd ACM conference on Computers and people research (pp. 5-10). ACM.

[33] Stewart, K. J., & Gosain, S. (2006). The impact of ideology on effectiveness in open source software development teams. Mis Quarterly, 291-314.

[34] Steinmacher, I., Conte, T., Gerosa, M. A., & Redmiles, D. (2015, February). Social barriers faced by newcomers placing their first contribution in open source software projects. In Proceedings of the 18th ACM conference on Computer supported cooperative work & social computing (pp. 1379-1392). ACM.

[35] Jensen, C., King, S., & Kuechler, V. (2011, January). Joining free/open source software communities: An analysis of newbies' first interactions on project mailing lists. In System Sciences (HICSS), 2011 44th Hawaii International Conference on (pp. 1-10). IEEE.