From Periphery to Core: A Temporal Analysis of GitHub Contributors' Collaboration Network

Ikram El Asri, Noureddine Kerzazi, Lamia Benhiba, Mohammed Janati

From Periphery to Core: A Temporal Analysis of GitHub Contributors' Collaboration Network

Ikram El Asri, Noureddine Kerzazi, Lamia Benhiba and Mohammed Janati

National Higher School for Computer Science
and System Analysis (ENSIAS)
in Rabat, Morocco
[ikram.Asri,n.Kerzazi,lamia.Benhiba, a.Janati]@um5s.net.ma

Abstract. Open-source projects in GitHub exhibit rich temporal dynamics, and diverse contributors' social interactions further intensify this process. In this paper, we analyze temporal patterns associated with Open Source Software (OSS) projects and how the contributor's notoriety grows and fades over time in a core-periphery structure. In order to explore the temporal dynamics of GitHub communities we formulate a time series clustering model using both Social Network Analysis (SNA) and technical metrics. By applying an adaptive time frame incremental approach to clustering, we locate contributors in different temporal networks. We demonstrate our approach on five long-lived OSS projects involving more than 700 contributors and found that there are three main temporal shapes of attention when contributors shift from periphery to core. Our analyses provide insights into common temporal patterns of the growing OSS communities on GitHub and broaden the understanding of the dynamics and motivation of open source contributors.

Keywords: Collaboration, Core-Periphery, Socio-Technical Relationships, SNA.

1. Introduction

Open source communities grow and fade over time. Understanding how to maintain, sustain, and grow a community of contributors is crucial for the survival and success of any open source project [1]. That said, more than 12,000 individuals have contributed to Linux since 2005, from which more than 4,000 contributed in just the last 15 months (50% are first-time contributors); 3000 for Rails; and 1403 for AngularJs. Scaling from one to thousands highly distributed developers in an interesting challenge of collaboration [2]. However, there is very little evidence as to how those virtual communities grow? And more interestingly, how newcomers can navigate from the periphery of a given project (i.e., first-time contribution) to the core contributors of the project (i.e., constantly committing, commenting, and participating in important decision-making)?

Recent studies have shown that only small portion of contributors leads an OSS project making a large proportion of technical contributions [3-8]. For instance, Mockus et al.[8] studied two open source projects: Apache and Mozilla and revealed that only 10 to 15 developers collaborate to carried out 80% of the contributions. Similarly, Dinh-Trong and Bieman [9] stated that only 28 to 42 contributors performed 80% of the development activity. Koch and Shneider [3] showed that 17% (51 out of 301 developers) provides core functionalities to the GNOME project. However, again these previous works do not help to understand how these small portions of contributors shift from the periphery, or even stick, to the core.

The picture that emerged from this evidence- contribution of developers in OSS projects- has been taken to shape OSS organization structures. OSS communities can be seen such as a Core-Peripheral structure [5]. At the core, there are those contributors who have been involved with the project for a relatively long time, are leading the project, and making significant contributions (80%) to the evolution of OSS projects. In the other side, at the periphery there are newcomers or people interested in the project and making few contributions with much less notoriety.

The key idea in our work is to analyze temporal patterns by which newcomers to OSS project shift from the periphery to the core teams. This shift remains largely uncovered even in organizational theories. Understanding this phenomenon within open source project can help gain insights on how to maintain virtual communities and how to attract new world wild contributors in order to accelerate software development projects in both OSS and traditional commercial organizations [10].

In this paper, we have undertaken a socio-technical analysis of five OSS collaborative communities aiming at uncovering the dynamics of growing and fading of those communities over time. Particular attention has been paid to the migration of newcomers from the periphery to core team.

Paper organization. The remainder of the paper is organized as follows. Section 2 presents related work. Section 3 provides our reasoning about core-periphery structure for the open source context. Section 4 describes our methodology including description of studied systems and data collection, for which the results are presented in Section 5. Section 6 discusses our finding and highlights some limitations. Finally, section 7 draws conclusions and enlightens future work.

2. Related Work

Open source software is built by teams of volunteers. A series of efforts in recent years have focused on the OSS development organization [1]. Newcomers are explorers who must orient themselves within an unfamiliar landscape. As they gain experience, they eventually settle in and create their own places within the landscape [11].

Understanding Motivation- Members of OSS communities are volunteers whose motivation to participate and contribute is a necessary condition to the success of open source projects. Ye and Kishida [12] argued that learning is one of the major driving forces that motivate people to get involved in OSS communities. Hars and Ou [13] categorized open source participants' motivations into two broad categories: internal factors meaning that open source programmers are not motivated by monetary rewards but by their own hobbies and preferences. External rewards, when contributors are interested in receiving indirect rewards by increasing their marketability and skills or demonstrating their capabilities in programming. Whatever the motivation behind the contribution the most interesting is a contributor level of activity and engagement within a project. In this paper, we are interested in that engagement in the project and how contributors gain notoriety and fade over time from core-periphery structure throw investigating technical and social collaboration activities of developers.

Detecting the Core-Periphery Structure- A series of efforts in recent years have focused on detecting the core-periphery structure in OSS projects. For example, from Social Perspective- Dabbish et al. [14] performed a series of in-depth interviews with central as well as peripheral GitHub users. Authors found that people make a rich set of social inferences from the networked activity information within GitHub and then combine these inferences into effective strategies for coordinating their work, advancing technical skills and managing their reputation. More concretely, Bosu and Carver [15] proposed a K-means classifier based on SNA metrics. for detecting core-periphery structure. We build upon these previous works to develop a further social perspective of collaboration. Our approach is based on k-means classifier using three classes (core, gray area, and peripheral) instead of two. We use three classes to include transitional states where nodes are neither core nor peripheral, which gives us a higher accuracy (80%) in identifying and dealing with peripheral vs core contributors.

Amrit and van Hillegersberg [16] examined core-periphery movement in open source projects and concluded that a steady movement toward the core is beneficial to a project, while a shift away from the core is not. Toral and al. [17] found that a few core members post the majority of messages and act as middlemen or brokers among other peripheral members.

Our study, by contrast is a field study of the migration of contributors from the periphery to core team. We, therefore, aim to analyze and understand interactions and contributors' evolvement per month. Specifically, we would like to address a practical question: can the activities of newcomers reveal who would be part of the core team leading the project?

Predicting Who Will Stay- Zhou et al.[18] attempted to predict who will stay in OSS communities. The authors proposed nine measures of involvement and environment based on events recorded in the issue tracking system. One of their funding stipulates that newcomers who are able to get at least one issue reported in the first month to be fixed are doubling their odds of becoming a long-term contributor. Gamalielsson et al. [19] studied the sustainability of Open Source software communities beyond a fork. Forking an OSS means that a sub set of contributors take another direction of the project because they are not in line with decisions made by notorious contributors.

3. Core-Peripheral Contributors

The existing literature provides a number of theories and approaches that may help identifying core-peripheral structures in OSS projects [5]. We could classify the OSS structure, as most of these previous works, in two classes core/peripheral. However, found that classification into three groups provides a more accurate model as reported in Table 1.

More specifically, we start from discovering and retrieving SNA metrics for co-edition files networks. Then we run a k-means classifier to identify three classes (core, gray area, and peripheral). Finally, we evaluate our classification using O(m) Algorithm for Cores Decomposition of Networks [20].

Step 1: The first step in this analysis was to compute the social networks (SNA) metrics for each project. We use an SNA package implemented in python, named Networkx[1], to compute automatically SNA metrics for each graph. As described in section 4, historical collaboration data have been processed to create one Network per month. Table 1 shows, in column 1, the number of generated sub-graphs for each project.

Step 2: The second step was to identify core/periphery contributors. To identify the core/peripheral structure in our OSS projects, we use K_means classifiers (k=3). The underlying idea is to classify contributors in three groups according to their SNA metrics and see whether they belong to Core – Peripheral – Gray area groups. Table 1, presents the size of each cluster (k=2 and k=3).

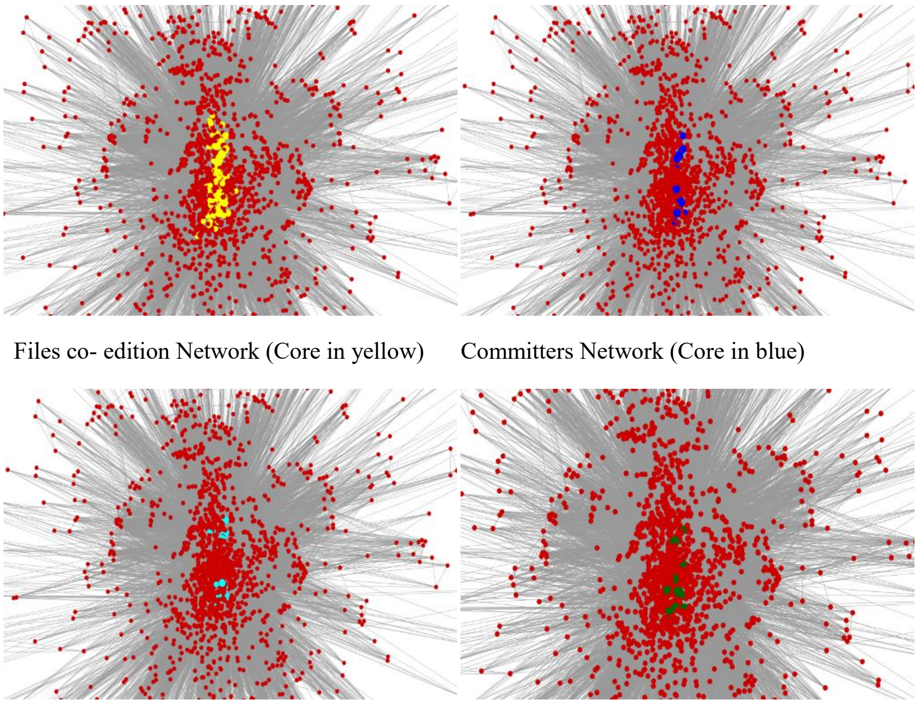
Table 1. k-means classifier precision

	Number of Sub-Graphs	Precision for 2 Classes (%)	Size (#Nodes by Cluster)	Precision for 3 Classes (%)	Size (#Nodes by Cluster)
AngularJs	79	67.0	(50, 1379)	80.1	(39, 169, 1221)
Docker	50	64.3	(92, 1540)	84.1	(24, 146, 1462)
Rails	62	67.1	(110, 3164)	85.0	(78, 519, 2677)
Symfony	86	68.7	(133, 1296)	81.7	(26, 172, 1231)
TensorFlow	16	74.2	(43 , 622)	87.6	(24, 56, 585)

Step 3: We cross validate the resulting k-means clusters for core-periphery using two methods. First, we compare k-means classification against the result of O(m) Algorithm for Cores Decomposition of Networks [20]. The algorithm takes as input a graph and provides partitions as an output. For randomly selected networks, we computed the k_core function provided by Networkx implementing the O(m) algorithm. The matching between the two algorithms was 100% agreement for small network. For large networks, we obtained a 68% agreement matching for core nodes(for example 24 node of 35 identified as core were in the partition with max k_core score).This can be due to the evolution of the collaboration network structure into an onion shape [21] with multiple layers of peripheral nodes and thus the increasing fuzziness of the gray-area-periphery border as the network grows in size.

[1] https://networkx.github.io/

We then manually inspect the visualization of a random sample of graphs using Cytoscape. We validated that identified core contributors effectively belong to a dense and cohesive bloc showing core members physically centered in the network, as depicted in Figure 1.



Files co- edition Network (Core in yellow) Committers Network (Core in blue)

Commenters Network (Core in blue) Reviews Network (Core in green)

Fig 1. Illustrative Examples of Networks Visualization.

4. Methodology

4.1 Study Subjects

In order to understand how a newcomer makes a shift from the periphery to being part of the core team within OSS projects, we studied socio-technical interactions for five long-lived and highly stared projects from GitHub. Hence, we have carefully chosen projects with approximately a thousand of contributors, with different lifespan, programming languages, and different domain in order to have diverse histories. Table 2 shows general information about the chosen OSS projects. In total, we have analyzed 850 stories of contributors spanning five different projects.

Table 2. Overview of the Studied Systems

	Language	Contribu-tors	Com-mits	Commits comments	Reviews	Reviews Comments	Line of Code
AngularJs	JavaScript	1,430	8,403	1,292	497	3,013	543,246
Docker	Go	1,633	31,291	298	4,754	23,153	1,039,309
Rails	Ruby	3,273	61,782	9,986	302	5,028	413,393
Symfony	PHP	1,424	30,106	2,309	3,226	17,014	744,619
TensorFlow	C++	700	15,221	147	111	872	1,349,495

4.2 Data Collection

We used the REST (Representational state transfer) API[2] provided by Github in order to get access to all the available information about hosted projects. The API provides access to a lot of information in JSON (JavaScript Object Notation) format. For each of the five studied projects we retrieved data history including: (1) information on commits [author, date, code churn, count of comments on commits, reviews, and edited files]; (2) and then for each edited file we were interested to investigate the collaboration between contributors with respect to co-edition files (Two contributors collaborate if they modify the same file). It is worth noting that the collaboration in our context is asynchronous (timeless) because a contributor can edit files years after another contributor.

Collaboration Over Files Co-edition- OSS projects such as GitHub are collaborative repositories hosting services including social features brought a new transparency to the development project [22]. Collaboration among GitHub users can be seen in different ways and forms several kinds of social networks. The most intuitive one is the network of collaboration between developers over projects. On the other hand, collaboration within the repository is of great importance and gained interests in OSS collaboration analysis [14]. In our case, coediting the same file is the dependent variable indicating whether or not the collaboration between two developers happened. We leverage on information of co-edition files to construct our collaborative networks.

4.3 Data Processing: Building Networks

The data sets have been processed and sliced by month to provide time frames (TF) for dynamic data analysis. For each time frame (for instance, 86 TF for Angular), we constructed three undirected, weighted cumulative networks: The first network is re-

[2] https://developer.github.com/v3/

lated to **Files co-edition Network** (FCN) where nodes represent contributors and edge weights represent the quantity of interactions between those contributors based on the amount of files they both edited. The second is **Committed Comments Network** (CCN) where nodes are commenters and edge weights are the amount of commits they interacted together. Finally, the **Review's comments Network** (RCN) based on comments interactions on reviews. Figure 1 illustrates the three examples of generated graphs.

We performed a dynamic network analysis on monthly sequences of collaboration networks based on files co-edition. By progressively adding one month activity after another, we obtained a sequence of cumulative collaboration networks that allows us to study the evolvement of social structures of each community as well as its contributor's evolution according to the core-periphery perspective.

5. Results

RQ1. *Are some activities more prevalent than others for a newcomer to become a core contributor?*

Motivation: Few newcomers in OSS end up being into the core of the project suggesting that somehow, they are gaining notoriety due to their participation in some collaborative activities such as changing source code, reviewing contributions from others, and commenting on commits and reviews. We aim at discovering what kind of contribution or collaboration are more relevant. We are interested in detecting existing correlations between social position and technical participation. Our primary goal is to equip the community of OSS with a better understanding of collaborative activities and potential guidelines for newcomers to play an efficient role.

Approach. We first identify the ascension of the top 10 core contributors for each. Next, we trace back the history of contributions aiming at quantifying collaboration activities. Then we considered contributors' activities under three types of contributions (#commits, #comments on commits, #comments on reviews, #editedFiles, and #changed Line of Code. We use a k-mean clustering approach to identify core contributors (see section 3) in monthly networks. This task requires us to identify different stages through which sequences of activities progress and then segment individual sequences according to the discovered stages. Finally, we measured the most correlated collaborative activity with respect to the contributors' social network metrics.

Results. Figure 2 shows the monthly evolution of core contributors for each studied project. It also illustrates how attractive is the project in terms of its capacity to build a large community of contributors. We were also interested to quantify the amount of contributors' transition from a status to another. Figure 3 illustrates a dynamic movement from periphery to the core and vice versa. For instance, we can track only the evolvement of core developer over time (CC) as well as the contributors that quit the core for a gray area (CG).

Our monthly analysis of OSS structure networks reveals a relatively stable evolution of the core teams. An average of 19.1 contributors per month for Docker, 11.6 for Angular, 16.46 for Symfony, and 16.2 for Tenserflow. We observed, for all projects, a monthly evolution of core contributors ranging between [.4 and 10.4].

In terms of featured activities, we found that newcomers spend significant portions of their time committing, editing source code files obviously adding and deleting lines of code more than commenting on commits and reviews.

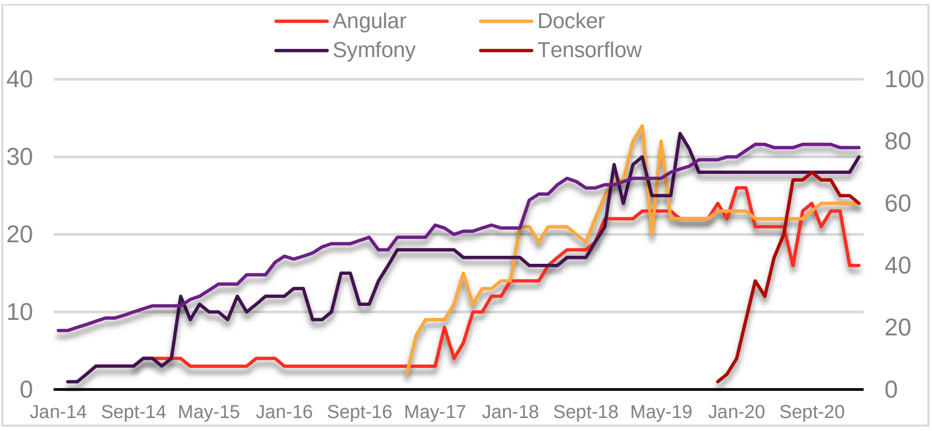


Fig 2. Monthly Evolution of Core Contributors.

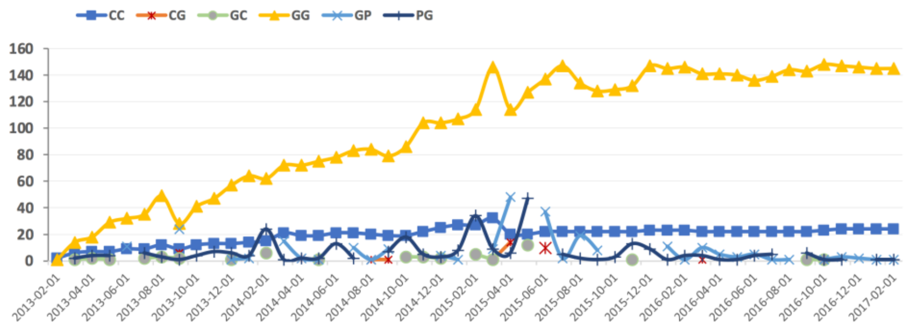


Fig 3. Monthly Evolution of contributors' status shifting for Docker

Table 3 shows the correlation between SNA centrality metric and the measure of each activity feature. One can notice that activities related to source code changes are more correlated to the positon of contributors within the structure core/periphery. For instance, we found for AngularJs project a correlation factor of .76 between staying in the core team and the number of contributor's commits.

Table 3. Average correlation between centrality metric and activity features

	Source Code Changes				Commenting	
	Commits	**Edited Files**	**Lines Additions**	**Lines Deletions**	**Commits comments**	**Reviews comments**
AngularJs	0.76	0.77	0.70	0.72	0.28	0.60

Docker	0.73	0.81	0.75	0.72	0.43	0.06
Rails	0.68	0.74	0.62	0.60	0.71	0.31
Symfony	0.77	0.83	0.85	0.84	0.54	0.39
TensorFlow	0.69	0.72	0.73	0.79	0.68	0.27

RQ2.*What collaboration activities can influence how long it takes for a contributor to be a core team member?*

Motivation. We aim at identifying those collaborative activities that are more supporting newcomers gaining notoriety and being part of the core team.

Approach. Knowing the most correlated metric from RQ1, we calculated the number of commits for core contributors of each project as well as the amount of Line of code add to the project.

Results. Figure 4 shows the medians for four projects, (52 , 5465) for AngularJS, (109, 21787) for Docker, (154, 7975) for Rails, (145 , 6421) for Symfony, and (80, 36155) for Tenserflow. The results show that the number of commits and the amount of line of code add are both a statistically significant characteristics of core contributors.

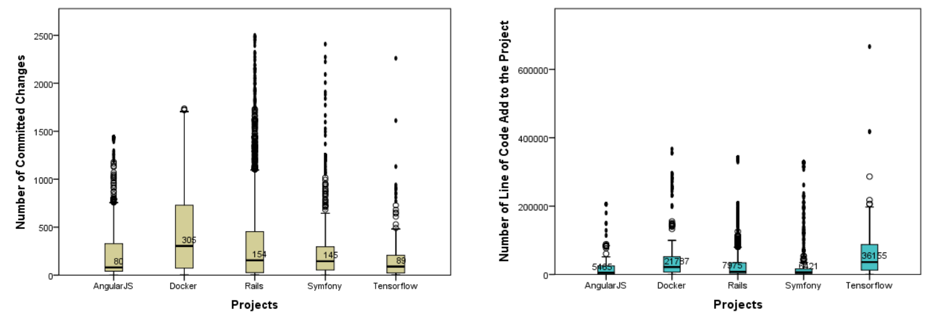


Fig 4. Number of commits (left) and LOC add from core contributors

RQ3. *Does the extent of involvement and environment predict whether a core contributor will churn from the project?*

Motivation. Enormous effort over the past decades was spent in attempts to understand factors that affect involvement and sustainability of OSS communities. We contribute to that body of knowledge through our predicting model.

Approach. To answer *RQ3* and predict who will leave the project, given the collaborative metrics, we applied the *J48* decision tree algorithm on the data clustered previously using k-means for all projects in WEKA[3]. To the purpose of this analysis, we filtered out only contributors that shift from the core to the gray area (CG) before the final transition to the periphery (GP), meaning leaving the project.

Results. Table 4 reports the results of our supervised machine learning approach regarding the four projects. For instance, we have 27 contributors within Angular project shift from the core to the gray area (CG). With a decision tree approach, we are able de predict 74.07% (.93 recall) the shift from C to G with only 7 out of 27 misclassified case. Interestingly, we found the root node of the decision tree to be the "EditedFiles" <= 871.39. This means that the amount of edited file is the most closely related metric to contributors' churn. The tree showed that if the number of edited files keep dropping then the contributor is likely to leave the core. However, the root activity of decision tree is not always the same for each project. We hypothesis that this difference is due to the progression stage of each project. For instance, it's easier to be part of the core team of Tenserflow, a relatively new project on Github, by just committing new changes.

Table 4. Machine Learning: Decision Tree Results (J48)

	Correctly Classified	Incorrectly Classified	Precision	Recall	Root Activity
AngularJs	20	7	74.07	.93	# Edited Files
Docker	43	0	100	1	# Commits
Rails	39	14	73.58	.84	CommentsOnReview
Symfony	19	13	59.37	.94	CommentsOnReview
TensorFlow	23	3	88.46	.70	# Commits

6. Discussion

6.1 Practical Implications

Understanding the involvement of contributors and their gain of notoriety can help the OSS communities to attract more valuable and highly motivated individuals. Moreover, analyzing the history of contributors 'activity may help to build a sustainable community of contributors around OSS.

[3] http://www.cs.waikato.ac.nz/ml/weka/downloading.html

Providing guidelines for whom want to lead future decisions of an OSS. We found the ascension of a newcomer becoming a core contributor to be associated mainly with technical contribution, especially the amount of code changes and interactions with existing source code. Most importantly, the number of commits and the amount of Line of code added rather than other activities such as commenting and reviewing others work. This may reflect some inherent differences between OSS projects and industrial ones in which we have other collaborative contributions such as requirement analysis, testing, etc.

Predicting who will churn along and who will stay is important. Allows project owners to find potential long-term contributors earlier and helps newcomers to improve their behaviors. Figure 5 illustrates tracking one contributor from Docker project. This contributor has belonged to the core team and then churned from the project at the end of 2013. If we could predict contributors 'turnover according to some aspects of behavior that we are able to model and quantify then we have the ingredients to build a sustainable long-lived community of contributors.

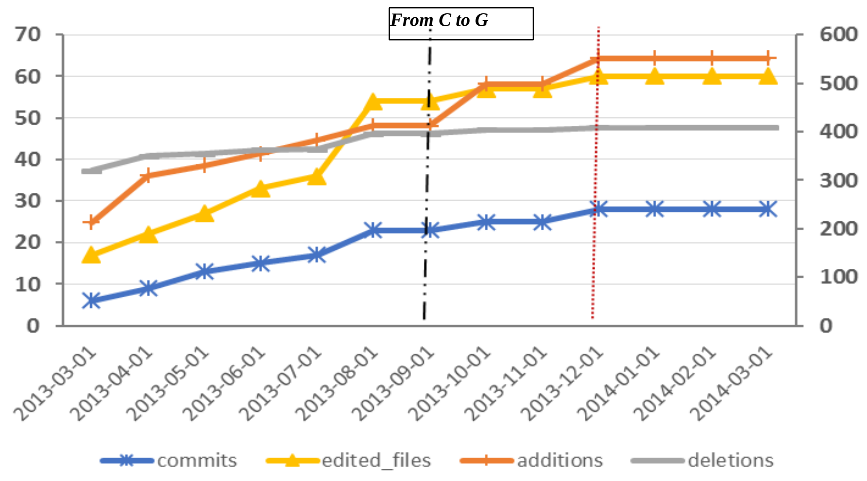


Fig 5. Contributor Turnover

In summary, understanding the shift from peripheral to core contributors and then sustainable core team in OSS projects requires an understanding of "Hidden" collaborative activities as well as the motivation behind observable developers' commitment. Therefore, while our results may be statistically valid, more care must be taken in interpreting their meaning to draw, for instance, a recipe to guide newcomer's behavior within OSS projects. Much work remains to be done in studying sustainable collaboration in open source projects.

6.2 Threats to Validity

We recognize few threats to our reported results. First, we did not check the bug database to assess the quality of contributions; instead we rely on crowd comments and code reviews that OSS communities use to enhance the software quality. Our choice was deliberate since we assume that core teams have gained their notoriety by performing in the quality also. However, such choice possesses threats of over estimation the quality of contributions. To mitigate the threat, we evaluate our classification against manual annotation. Second, we make certain assumption based on how we slice and build our co-edition, comments, and reviews graphs. For instance, we consider cumulative data for the co-edition network per month, and thus, building networks from the beginning of the projects up to the studied month. We consider source code collaboration as a sustainable activity in the sense that contributors leverage on the previous work of each other.

Finally, our study is the subject of statistical conclusion validity which refers to the ability to make an accurate assessment of the strength of the relationship between our independent and dependent variables. For instance, in section 3, we used k-means to cluster and segregate contributors in three categories (core, peripheral, and gray area) according to a series of metrics. To gain more confidence on our classification approach, we triangulate our results using different methods such as SNA metrics, O(m) Algorithm, GitHub information, and visual inspection of collaborative networks.

7. Conclusion

In this paper, we study the fine-grained evolution of structural collaborative networks of five OSS projects. We analyzed the evolution of communication patterns over time, we presented a dynamic visualization based on time series analysis by dividing the long period of the project into several consecutive time frames (one month). We were also interested to quantify the amount of contributors' transition from the periphery to the core crossing what we call a gray area. We have observed, for all projects, a monthly evolution of core contributors ranging between [0.4 and 10.4].

Information on developer roles is crucial to understanding the collaborative dynamics of software projects. Ultimately, the most important collaborative activity to join and stay in the core team of an OSS is the amount of submitted changes (#commits). The more source code changes a new contributor submits, the faster and more likely he will make it the shift to the core team. Finally, depending on progression stage of the project, a drop in certain collaborative activity predicts who will leave the core. Our future work will focus further on community building and how to build a sustainable OSS ecosystem.

References

[1] M. Zhou and A. Mockus, What make long term contributors: willingness and opportunity in OSS community. in Proc. of the 34th Int'l Conf. on Software Engineering (ICSE '12), (Zurich, Switzerland), 518-528, 2012.

[2] P. Hinds and C. McGrath, Structures that work: social structure, work structure and coordination ease in geographically distributed teams. in Proc. the 20th Int'l Conf. on Computer Supported Cooperative Work, (Banff, Alberta, Canada), 343-352, 2006.

[3] S. Koch and G. Schneider Effort, cooperation and coordination in an open source software project: Gnome. Information Systems Journal, 12 (1): 27-42, 2002.

[4] M. Goeminne and T. Mens, Evidence for the pareto principle in open source software activity. in the 1st Int'l Workshop on Model Driven Software Maintenance and 5th Int'l Workshop on Software Quality and Maintainability, 74-82, 2011.

[5] J. Geldenhuys, Finding the Core Developers. in 36th Euromicro Conf. on Soft. Eng. and Advanced Applications, (Lille, France), 447-450, 2010.

[6] T.T. Dinh-Trong and J.M. Bieman The FreeBSD project: a replication case study of open source development. IEEE Transactions on Software Engineering, 31 (6): 481-494, 2005.

[7] G. Robles,S. Koch,J.M. Gonzalez-Barahona, et al., Remote analysis and measurement of libre software systems by means of the cvsanaly tool. in Proc. the 2nd ICSE Workshop on Remote Analysis and Measurement of Software Systems, 51-55, 2004.

[8] A. Mockus,R.T. Fielding and J.D. Herbsleb Two Case Studies of Open Source Software Development: Apache and Mozilla. ACM Trans. Softw. Eng. Methodol., 11 (3): 309--346, 2002.

[9] T. Dinh-Trong and J. Bieman The freebsd project: a replication case study of open source development. IEEE Trans. on Software Engineering, 31 (6): 481-494, 2005.

[10] S. Lussier New Tricks: How Open Souce Changed the Way My Team Works. IEEE Software, 21 (1): 68-72, 2004.

[11] B. Dagenais,H. Ossher,R.K.E. Bellamy, et al., Moving into a new software project landscape. in Proc. of the 32 Int'l Conf. on Software Engineering (ICSE '12), (ape Town, South Africa), 275-284, 2010.

[12] Y. Ye and K. Kishida, Toward an understanding of the motivation open source software developers. in Proc of the Int'l Conf. on Software Engineering (ICSE'03), 419-429, 2003.

[13] A. Hars and O. Shaosong Working for free? Motivations of participating in open source projects. 9, 2001.

[14] L. Dabbish,C. Stuart,J. Tsay, et al., Social coding in GitHub: transparency and collaboration in an open software repository. in the conference on Computer Supported Cooperative Work, (Seattle, WA, USA), 1277-1286, 2012.

[15] A. Bosu and J.C. Carver, Impact of developer reputation on code review outcomes in OSS projects: An Empirical Investigation. in Proceedings of the 8th International Symposium on Empirical Software Engineering and Measurement, (Torino, Italy), 1-10, 2014.

[16] C. Amrit and J. van Hillegersberg Exploring the impact of socio-technical core-periphery structures in open source software development. Journal of Information Technology, 25 (2): 216-229, 2010.

[17] S.L. Toral,M.R. Martínez-Torres and F. Barrero Analysis of virtual communities supporting OSS projects using social network analysis. Information and Software Technology, 52 (3): 296-303, 2010.

[18] M. Zhou and A. Mockus Who Will Stay in the FLOSS Community? Modeling Participant's Initial Behavior. IEEE Trans. on Soft. Eng., 41 (1): 82-99, 2015.

[19] J. Gamalielsson and B. Lundell Sustainability of Open Source software communities beyond a fork: How and why has the LibreOffice project evolved? Journal of Systems and Software, 89: 128-145, 2014.
[20] V. Batagelj and M. Zaversnik An O(m) Algorithm for Cores Decomposition of Networks. Adv. in Data Analysis and Classification, 5 (2): 129-145, 2003.
[21] V.H.P. Louzada,F. Daolio,H.J. Herrmann, et al. Smart rewiring for network robustness. Journal of Complex networks, 1 (2): 150-159, 2013.
[22] N. McDonald,K. Blincoe and E.V.A. Petakovic Modeling Distributed Collaboration on Github. Advances in Complex Systems, 7 (8)2014.