Understanding OSS Project's Collaborative Dynamics: Core and Peripheral Interactions

Ikram El Asri, Mohammed Abdou Janati Idrissi

Abstract— The aim of this paper is to explore how open source software (OSS) development communities grow and fade. Sustaining the evolution of OSS communities and attracting volunteer contributors is crucial for the actual industry of the software. Therefore, it is important to figure out temporal patterns by which OSS contributors change their roles over time by shifting from peripheral contributors to core teams. We first formulate a time series clustering problem using SNA metrics to identify core-periphery structure. Then we characterize the changes made to the source code of five open source projects with respect to the amount of structural complexity introduced or alleviated by either core or peripheral contributors. Our analysis provides insights into common temporal patterns of the growing OSS communities on GitHub and broaden the understanding of project's collaboration dynamics. Our analysis reveals an interesting growth of core team members and they are substantially less likely to leave their social position. In addition, we identified the main characteristics of contributors that allow the transition from periphery to the core. Finally, we found that a drop in certain collaborative activity predicts who will leave the core team.

Index Terms— Collaboration, Core-Periphery, Socio-Technical Relationships, SNA.

1 Introduction

Open-source software development (OSS) communities grow and fade over time. Understanding how to sustain the growth of a community of free contributors is crucial for the survival and success of any open-source project [1]. For instance, more than 12,000 volunteers have contributed to Linux since 2005, from which more than 4,000 contributed in just the last 15 months since 2015 (50\% are first-time contributors); 3000 for Rails; and 1403 for AngularJs. Scaling from one person to thousands highly distributed free developers remains an interesting challenge of collaboration [2]. However, there is very little evidence as to how those virtual communities grow. Furthermore, how newcomers can navigate from the periphery (i.e., first-time contribution) to the core contributors leading the project (i.e., constantly committing, commenting, and participating in important decision-making)? The picture that emerged from this evidence- contribution of developers in OSS projects- has been taken to shape OSS organization structures. Contributors are often classified according to the dichotomy of core and peripheral roles [3]. 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.

Recent studies have shown that only small portion of contributors leads an OSS project making a large proportion of technical contribution [4], [5], [6], [3], [7], [8]. For instance, Mockus et al. [9] studied two open source projects: Apache and Mozilla and revealed that only 10 to 15 developers collaborate to carry out 80% of the contributions. Similarly, Dinh-Trong and Bieman [6] stated that only 28 to 42 contributors performed 80% of the development activity. Koch and Shneider [4] showed that 17% (51 out of 301 developers) provides core functionalities to the GNOME project. However, we lack basic knowledge about how peripheral contributors become core members, or even how they stick to the core team. The key idea in our work is to analyze temporal patterns by which newcomers to OSS project shift from the periphery to core teams. This shift remains largely uncovered even in the literature related to organizational theory. Understanding this phenomenon within open source projects can help gain insights on how to maintain virtual communities and how to attract new worldwide contributors in order to accelerate software development projects in both OSS and traditional commercial organizations [10].

To this end, we have undertaken a socio-technical analysis of five OSS communities aiming at uncovering the dynamics of growing and fading cycles of those communities over time. Particular attention has been paid to the migration of newcomers from the periphery to core teams. Thus, the research questions tackled in this work can be listed as follows:

- **RQ1.** How accurate is our approach to classify OSS contributors as core vs. peripheral?
- **RQ2**. How often contributors shift from periphery to the core team?
- **RQ3.** What are the main characteristics of those contributors that make the transition from periphery to the core?

- *Mohamed V University, National Higher School for Computer Science and System Analysis (ENSIAS) in Rabat, Morocco*
- *[ikram.Asri, a.Janati]@um5s.net.ma*

- **RQ3.1**. Does task type increase the chance for a newcomer to become a core member?
- **RQ3.2.** Do metrics related to activities help to predict whether core contributors will churn from the project?

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

2 Related Work

In this section, we briefly summarize the current state of research on structural organization of OSS contributors and its apparent limitations. Then we review several approaches for detecting Core/Peripheral contributors.

Open source software is built by a virtual structure of volunteers. A series of efforts in recent years have attempted to study the OSS development organization[1]. The literature reports on a Periphal/Core structure where Newcomers (i.e., Peripheral) are explorers of an OSS project who must orient themselves within an unfamiliar landscape to make a contribution. Few of them gain experience, and eventually settle in and create their own places within the landscape (i.e., the core members) [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 OSS project engagement and how contributors gain notoriety, and then fade over time from the core-periphery structure through the investigation of technical and social collaboration activities of developers.

Detecting the Core-Periphery Structure- The intuitive notion of core-periphery network, as a structure consisting of a densely-connected bunch of nodes (i.e., the Core) and low-degree nodes preferentially connected to the core (i.e., the Periphery) has been firstly formalized by Borgatti & Everett [14]. A series of efforts in recent years have focused on detecting the core-periphery structure and the characteristics of each group. For example, from Social Perspective- Dabbish et al. [15] 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 - communication and collaboration patterns - 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 [16] proposed a K-means classifier based on SNA metrics in order to detect the Core and Peripheral groups. We build upon these previous works to detect a further communication and collaboration patterns. Our approach is based on k-means classifier using three clusters instead of two: Core, Transitional, and Peripheral. We improve over the state of the art by considering a third cluster to represent a transitional state in-between Core and Peripheral, which gives us a higher accuracy (80\%) in identifying and dealing with OSS structures.
Amrit and van Hillegersberg [17] 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. [18] found that a few core members post the majority of messages and act as middlemen or brokers among other peripheral members. Nevertheless, there is an evidence that peripheral developers are just as critical to the project's success as core developers [19].
Our study, by contrast is a field study of the contributors' migration from the periphery to core team. We, therefore, aim to analyze and understand interactions and contributor's 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. [20] 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. [21] 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.

Social network analysis- Social network analysis is an essential element in social science research [22], [23]. Social network analysis has emerged due to the advances in information technology and promotes a better understanding of different research topics in engineering, business, economic and social sciences [24], [25]. The basic approach relies on representing communication events as links between the actors (nodes) in the network. Computing various global or node-specific metrics for a network is useful for making general statements about specific networks or classes of networks. Examples of such metrics are betweenness, diameter, distance, density, betweenness centrality, degree centrality, or eigenvector centrality [26], [27]. Sociologists have found that people's positions in the social networks are closely related to individual outcomes, e.g., better paid and getting faster promotions [28]. The connections in the social networks are an essential asset for people to gain access to vital information and resources to compete, to negotiate, and to innovate [29].
SNA has been exploited in several previous studies on OSS, Madey et al. looked at how projects are linked by individuals participating in more than one project [30]and suggested there are individuals who are important boundary-spanners between many projects. Crowston and Howison [31] looked at how developers are linked by working on the same artifacts in the defect tracking system. They found that it is less likely in a large project that one developer dominates the communication regarding a defect artifact. Bird et al. [32] considered five large OSS projects and looked at developers working together on the same files and reply-to relationships on the mailing-list. They found that (1) the communication network was modular, i.e. sub-groups could be identified, (2) that developers discussed product-centric topics in a smaller group, while other topics were discussed more broadly in the community, and (3) that people who interacted on the mailing-list also were much more likely to work together on the same files in the repository. De Souza et al. [33] extended this by static call graph analysis and followed the evolution of the situation over a long time. They found shifts in participation from the periphery to the core of a project and vice versa, as well as changes to the ownership of files over time.
This paper uses social network analysis to quantify contributors gain of reputation within the core periphery structure.

3 METHODOLOGY

3.1 Study Subjects

Our field study is GitHub, which is a collaborative coding environment that employs social media features. Github encourages software developers to perform collaborative software development by offering distributed version control and source code management services with social features (i.e., user profiles, comments, and broad-casting activity traces) [15]. By the end of 2017, GitHub is the most famous code hosting platform, with more than 28 million users distributed in 200 countries and more than 67 million hosted projects [34]. 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 projects from GitHub according to the following criteria:

- Projects should be active and highly stared since stars shows the popularity of the project.
- Projects should be long-lived, created at least two years prior to data collection. It ensures to explore the evolution for Core/Peripheral structure over time.
- Projects should have more than a thousand of contributors, with different lifespan, programming languages (PHP, Ruby, Python, C++, JavaScript, Rust and Go), and different domain in order to have diverse histories.

.

Table 1 and Table 2 show respectively projects descriptions and general information about the chosen open source projects.

Table 1. Study subject's description

Project name	description	Language	Created at
AngularJs	A JavaScript-based open-source front-end web application framework mainly maintained by Google and by a community of individuals and corporations to address many of the challenges encountered in developing single page applications.	JavaScript	01/2010
Moby	A collaborative project for the container ecosystem to assemble container-based systems	Go	01/2013
Rails	Aweb-application framework that includes everything needed to create database-backed web applications according to the Model-View-Controller (MVC) pattern.	Ruby	01/2005
Symfony	A PHP framework for web applications and a set of reusable PHP components.	PHP	01/2010
TensorFlow	An open source software library for numerical computation using data flow graphs.	C++	11/2015

Table 2. Overview of the Studied Systems

	Total Contributors	Total Commits	Total Commits comments	Total Reviews	Total Reviews Comments	Total Lines of Code
AngularJs	1,430	8,403	1,292	497	3,013	543,246
Moby	1,633	31,291	298	4,754	23,153	1,039,309
Rails	3,273	61,782	9,986	302	5,028	413,393
Symfony	1,474	30,106	2,309	3,226	17,014	744,619
TensorFlow	700	15,221	147	111	872	1,349, 495

3.2Data Collection

We used a REST (Representational state transfer) API 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; (3) Reviews comments were collected with timestamps and commenters.

Thus, we perform HTTP GET requests to the 'api.github.com' server following the syntax '/repos/:owner/:repo/commits?page=:page' to get all of the commits for a specific repository, where: owner is the owner of the repository, :repo is the repository ; :page is the commits' page number. This request sends back a list of JSON objects, which contain all the information related to each commit, such as its Id (called sha in GitHub language), the title, the text body, the author, the creation and modification dates, the number of comments received, a link to details of modified files, etc.

3.3Data Processing: Building Dynamic Temporal Networks

We build a social network as a graph G = [V, E] which consists of a set of agents V and a set of edges E connecting them. A dynamic social network consists of a series of observations of social networks at different time steps [G1, G2,..., Gn]. A dynamic social network contains not only a set of relationships between agents, but also information on how these relationships change over time. For our case we leverage on information of co-edited files to construct our dynamic collaborative networks similar to [35]. The data sets have been processed and sliced per month to provide time frames (TF) for dynamic data analysis. For each time frame (for instance, 79 TF for AngularJS), we constructed a cumulative co-edited file Network (CFN), by progressively adding one-month activity after another. The CFN can be modeled as a graph Gt = (Nt, Et) where Nt represents OSS contributors and Et the set of interactions among them at time frames t. Co-editing the same file is the dependent variable indicating whether or not an interaction between two developers happened. Hence, for every node i and j, an undirected link is drawn between i and j when i and j has edited the same file (see Fig. 1).

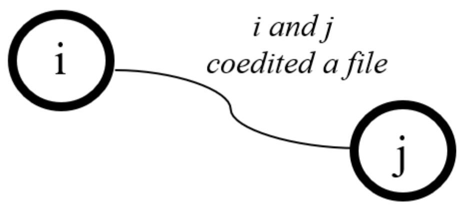

Fig. 1 Temporal files co-edition network.

We consider Gt an undirected weighted graph where the weight link is an aggregate of the quantity of interactions between those contributors based on the number of files, they both edited. 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.

4 Core-Peripheral Contributors

RQ1. How accurate is our approach to classify OSS contributors as core vs. peripheral?

Motivation: The existing literature provides a number of theories and approaches that may help identifying Core-Peripheral structures in OSS projects. We could classify the OSS structure, as most of these previous works, in two classes core and peripheral. However, we found that classification in three groups provides a more accurate model as reported in **Error! Reference source not found.**. For instance, for AngularJS project the precision of classifying contributors into two clusters is 67% while the precision relying on three clusters perform better 80.1%. Thus, we answer our first research question related to accuracy of clustering OSS contributors as Core vs. Peripheral.

Table 3. k-means Classifier Precision

		2 Classes		3 Classes	
	#Sub-Graphs	Precision %	#Nodes by Cluster)	Precision %	#Nodes by Cluster)
AngularJs	79	67.0	(50, 1379)	80.1	(39, 169, 1221)
Moby	50	64.3	(92, 1540)	83.7	(22, 156, 1425)
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)

Approach: In our proposed method, we used k-mean algorithm to cluster OSS contributors based on historical data sliced per month. We used Elbow method [36]to find the optimum number of clusters K as input to our k-means algorithm. This algorithm looks at the percentage of variance explained as a function of the number of clusters: the principle here is to choose a number of clusters so that adding another cluster doesn't improve the model anymore. Fig. 2 shows the results after running k-means clustering for k going from 1 to 5 on randomly chosen temporal entries for AngularJS project. One can see a pretty clear elbow at k = 3, indicating that 3 is the best number of clusters. Once the K is fixed, we followed the three steps bellow to classify contributors in three groups according to their SNA metrics and see whether they belong to Core, Transitional, or Peripheral groups

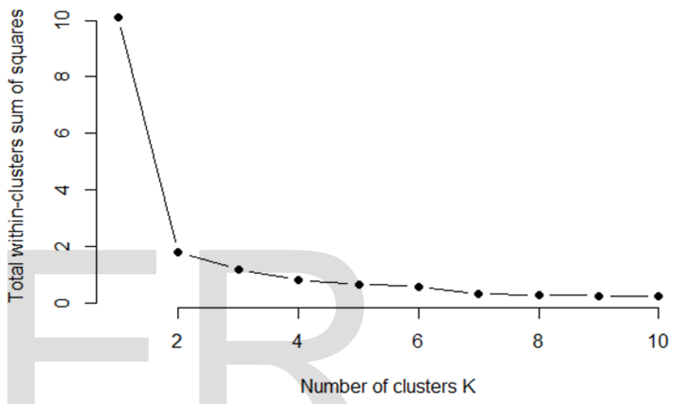

Fig. 2 Assessing the Optimal Number of Clusters with the Elbow Method.

Step 1: We compute[1] the social networks metrics for each project. It is worth noting that the historical collaboration data have been proceeded to create one Network per month. **Error! Reference source not found.** shows (#sub-Graphs), which represent the amount of generated sub-graphs for each project. For instance, we generated 79 sub graphs - collaboration networks - for AngularJs project. We computed SNA metrics for each node in each sub graph out of the 79.

Step 2: For each sub network, we used k-mean to cluster contributors in our three classes based on SNA metrics from step one. **Error! Reference source not found.** presents the results in column ("#Nodes by Cluster"). For instance, the last analyzed month of AngularJs project embodies (Core=39, Transitional=169, Peripheral=1221). One can see the monthly evolution of Core team size in section 5 Fig. 7. The organizational structure of OSS projects is like the peels of an onion as described previously by Oezbek .al [37].

[1] Using Networkx SNA package: https://networkx.github.io/

Step 3: We cross validated the resulting k-means clusters for core-periphery using two methods. First, we compared k-means classification against the result of O(m) Algorithm for Cores decomposition of Networks [38]. The algorithm takes as input a graph and provides as an output a certain amount of partitions. Second, we manually inspected the visualization of a random sample of graphs using Cytoscape tool.

Results: The cross validation of our k-mean results against O(m) Algorithm showed an agreement ranging from 60\% to 100\% between the two approaches as depicted in Fig. 3. For instance, on January 2015 our k-mean cluster 27 contributors as core while O(m) Algorithm detects 17 (out of 27) as Core members.

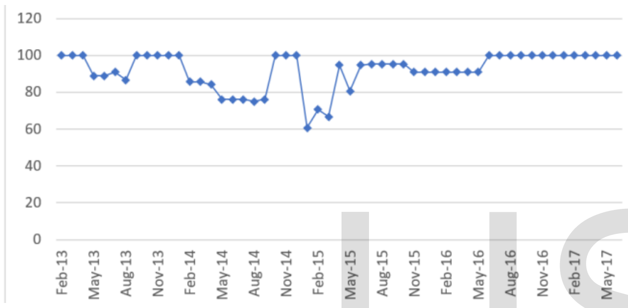

Fig. 3 Agreement between our k-mean approach and o(m) for core contributors clustering (Moby project).

On the other hand, Fig. 4 visually inspect the position of core contributor's into collaboration networks. Thus, we validated visually that the core contributors belong to a dense and cohesive bloc showing core members in the network (color Yellow).

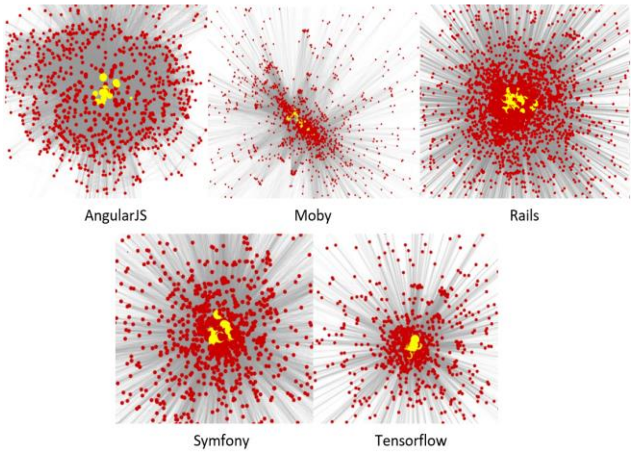

Fig. 4 Visualization of Core Contributors (Yellow) within co-edition Networks.

This result has provided evidence that our clustering approach produces consistent classifications of OSS contributors into Core/peripheral structure and that developer networks provide specific characteristics of the content of each cluster. Next, we present the results of the transitions patterns between clusters.

5 Results

RQ2. How often contributors shift from the periphery to the core team?

Motivation: Understanding the roles contributors play in an OSS project is crucial to figure out the project's collaborative dynamics. Previous work [39] has shown that core developers typically attain their credibility through consistent involvement and often have accumulated knowledge in particular areas of the system over substantial time periods. Although peripheral contributors might be considered as a risk for an OSS project's success considering their volatile nature of commitment and the known problems of knowledge loss and inadequate changes [40], without them there is a limited opportunity for a screening process to identify and promote appropriate developers. Peripheral contributors are also crucial for an OSS project's success in many ways for a high-quality software product. As stated by Mockus et al. [41], it is important to maintain a balanced composition of a structure of core/peripheral in a community, otherwise an OSS community is not sustainable. Fig. 5 points out an example of a role change from periphery to core for one AngularJs developer. Understanding the stability patterns of developer roles can help practitioners to understand the potential risks associated with each role. If stability is uniform across roles, then it may be hard to implement strategies or organizational structures that mitigate risk. However, if stability is substantially greater in one group than in another, it would be sensible to mitigate risk by delegating responsibilities that demand long-term involvement to those individuals that are most likely to be stable [42].

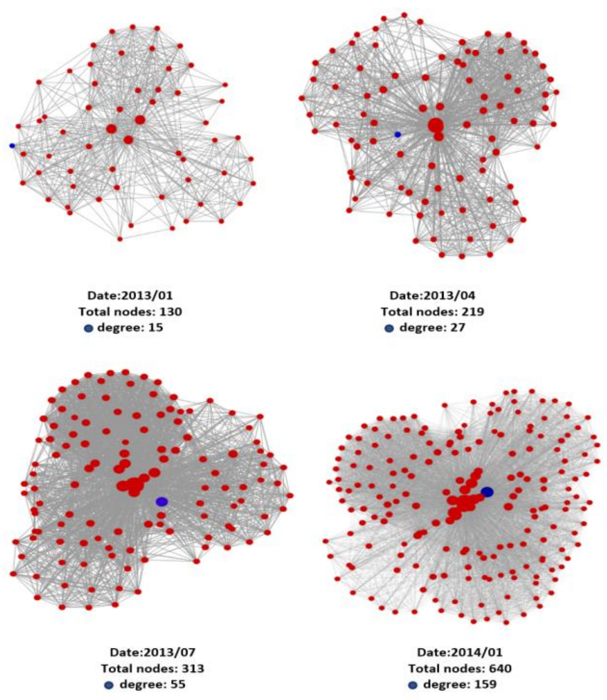

Fig. 5 Snapshots of progression from periphery to core within collaboration network (Example of a chosen contributor from AngularJS project).

Approach: we investigate transitions patterns of developers through different roles (i.e., Core, Peripheral) by tracking changes in the corresponding dynamic developer network over time. We calculated developer stability by exploring the monthly probability of developers' transition from one role to another. Thus, for each developer, using RQ1 results we are able to identify time ordered sequence of roles change during his involvement in the project. Using this assumption, we are able to represent developer transitions from state to state as an N * N transition matrix, in which each element indicates the average probability of transitioning (presented in percentage) from any state to any other state during the entire project's evolution.

Results: Fig. 6 shows the transition probabilities between developer states are shown in the form of a Markov chain. The primary observation is that developers in a core role are substantially less likely to transition to the Transitional zone and do never make a transition back to periphery. Based on this result, we can confirm that the core developers represent a stable group. For instance, for AngularJs project we observed that the core developers stay in this area with a 97% probability, transit to the Transitional state with 3% probability and with 0% probability transit directly to the peripheral state.

Our monthly analysis of the evolution of core team size in studied OSS projects reveals an interesting growth of the core teams as depicted in Fig. 7. This finding illustrates how attractive is the project in terms of its capacity to gain a large number of faithful and engaged contributors. The core team of AngularJS started with only one contributor and we counted 26 developers in core team by Jun 2017.

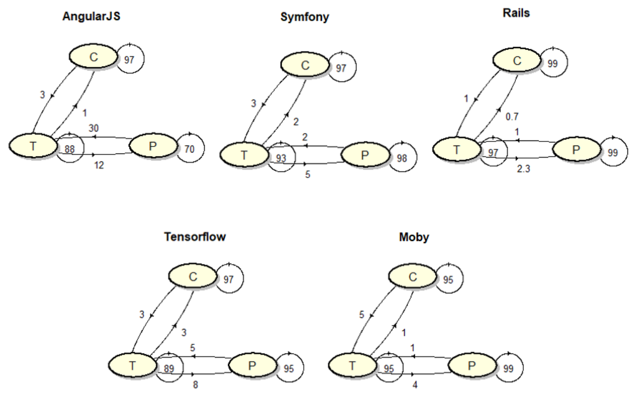

Fig. 6 Developers role stability for studied projects shown in the form of a Markov chain.

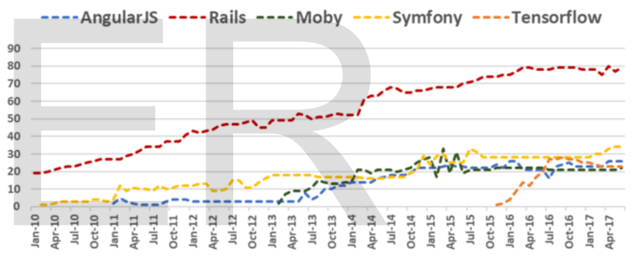

Fig. 7 Monthly Evolution of Core Contributors.

Furthermore, the percentage of T to C (form Transitional zone to core group) transition remains very important. ***On average 1% of contributors in Transitional zone join the core team.*** New peripheral contributors are as important as core members for any OSS project. Some of them are aspiring to become core members through continuous contributions. Our next research question RQ3.1 will identify the most prevalent activity supporting this type of transition.

Also, transitions from one cluster to another are also interesting to study, especially contributors churning from the core team. Shifts away from the core are clearly not good since it can be considered as an indication of the instability of the project [17]. Thus, our RQ3.2 will examine if the extent to which the involvement and environment can predict whether a core contributor will churn from the project.

RQ3. What are the main characteristics of those contributors that make the transition from periphery to the core?

We are interested in analyzing properties of developers' roles change within OSS communities. Hence, we formulate the following research questions:

RQ3.1. Does task type increase the chance for a newcomer to become a core member?

Motivation: As seen in RQ2, few newcomers in OSS end up being into the core team suggesting that somehow, they are gaining reputation due to their participation in specific collaborative activities such as changing source code, reviewing contributions from others, and commenting on commits and reviews. This research question aims 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 project. Next, we trace back the history of contributions aiming at quantifying collaboration activities. We considered contributors' activities under five types of contributions detailed bellow:

- Count Commits: The number of commits a developer has authored (merged to the master branch). A commit represents a single unit of effort for making a logically related set of changes to the source code.
- Lines of code (LOC): The sum of added and deleted lines of code a developer has authored (merged to the master branch).
- Count edited files (distinct count): Each commit's merge is modifying a set of files.
- Count comments on commits: The number of contributor's interventions in commits discussions.
- Count comments on reviews: The number of contributor's interventions on reviews request discussions

Finally, we correlated collaborative activity with respect to the contributors' social network metrics.

Results: Table 4 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 position 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. 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.

Table 4. 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

For a further exploration, we calculated the distribution of commits count as well as the amount of line of code add by core contributors. Fig. 8 shows the medians for the five studied projects, (52 , 5465) for AngularJS, (109, 21787) for Moby, (154, 7975) for Rails, (145 , 6421) for Symfony, and (80, 36155) for Tensorflow. The results show that the number of commits and the amount of line of code add are both statistically significant to characterize core contributors.

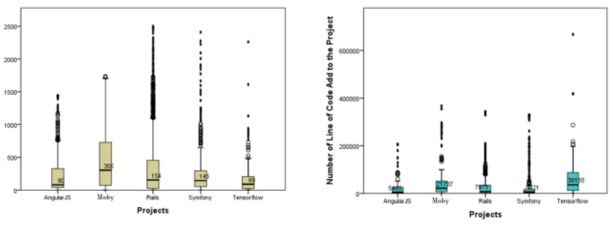

Fig. 8 Number of commits (left) and LOC add from Core Contributors.

RQ3.2. Do metrics related to activities help to predict whether core contributors will churn from the project?

Motivation: A lot of 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 predictive model.

Approach: To answer this research question and predict who will leave the project and who will stay, given the collaborative metrics, we applied the J48 decision tree algorithm on the data clustered previously using k-means. To the purpose of this analysis, we filtered out only contributors that shift from Core to Transitional area (C-T) before the final transition back to the Periphery (T-P), which means they are churning from the project.

Results: Table 5 reports the results of our supervised machine learning approach regarding the four projects. For instance, we have 27 contributors within Angular project that shift from the Core to the Transitional area (C-T). With a decision tree approach, we are able to predict 74.07% (.93 recall) the shift from C to T 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 Tensorflow, a relatively new project on GitHub, by just committing new changes.

Table 5. 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

6.1Practical Implications

Understanding the involvement of contributors and their gain of reputation 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.

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. It allows project owners to find potential long-term contributors earlier and helps newcomers to improve their behaviors. Fig. 9 illustrates tracking one contributor from Moby 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.

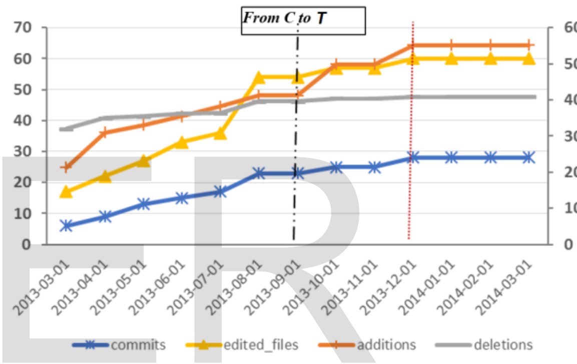

Fig. 9 Contributor Turnover.

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.

6.2Threats to Validity

Internal 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 reputation by performing in the quality also. However, such choice rises threats of overestimation the quality of contributions. To mitigate the threat, we evaluated our classification against manual annotation. Second, we made an assumption related to our approach of slicing and building 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 considered 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 5, we used k-means to cluster and segregate contributors in three categories (core, peripheral, and transitional 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.

External Validity: The main threat to the external validity of our findings is the problem that our subject projects might not be representative of the entirety of OSS projects. Although, the projects do represent a broad spectrum in several dimensions (from different domains, written in different programming languages and have different time spanning), they are still limited to relatively successful, mature, and large projects. Thus, our results may not be relevant for less mature or very small OSS projects. Prior research indicates that OSS may not be seen as one universal phenomenon but that considerable differences between the projects exist [31]. Hence, the findings we achieved for our five case studies may not be representative of the entirety of OSS development. We mitigate this problem by choosing a very diverse set of OSS projects, as shown in Section 3.1, the projects vary considerably in size, source-code activity, commits activity, and reviews activity. Thus, our findings should not be significantly biased.

7 Conclusion

In this paper, we study the fine-grained evolution of projects' collaborative dynamics of five OSS projects. We analyzed the evolvement of contributors from periphery to core teams over time, we presented a dynamic visualization based on time series analysis by slicing the long period of the project into several consecutive time frames (one per month). Then we proposed a k-mean clustering approach based on SNA centrality metrics to dynamically classify contributors in monthly collaboration networks in three classes Core, Transitional and Peripheral. Our approach has shown a good agreement with the O(m) core decomposition algorithm. Moreover, visual inspections validated that classified core contributors effectively belong to a dense and cohesive bloc physically centered in the network.

We were also interested to quantify the number of contributors' transition from the periphery to core teams. We have observed a monthly evolution of core contributors ranging between [0.4 and 10.4]. Also core team has shown a probability ranging between [97% and 99%] for staying in the core.

Information on developer roles is crucial to understanding the project's collaborative dynamics. Our results suggest that the most important collaborative activities to join and stay with the core team of an OSS is activities related to source code changes (#commits, #LOC). 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, such as \#commits predicts who will leave the core. Our future work will focus further on a qualitative study to get more insights from contributors who made the transition and have become Core members.

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, 2012, pp. 518-528.

[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, 2006, pp. 343-352.

[3] J. Geldenhuys, "Finding the Core Developers," in *36th Euromicro Conf. on Soft. Eng. and Advanced Applications*, Lille, France, 2010, pp. 447-450.

[4] S. Koch and G. Schneider, "Effort, cooperation and coordination in an open source software project: Gnome," *Information Systems Journal,* vol. 12, no. 1, pp. 27-42, 2002.

[5] G. Robles, S. Koch, J. M. Gonzalez-Barahona, and J. Carlos, "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*, 2004, pp. 51-55.

[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,* vol. 31, no. 6, pp. 481-494, 2005.

[7] 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, 2011, pp. 74-82.

[8] Y. Tian, P. S. Kochhar, E.-P. Lim, F. Zhu, and D. Lo, "Predicting Best Answerers for New Questions: An Approach Leveraging Topic Modeling and Collaborative Voting," in *Proc. of the Int'l Workshops, QMC and Histoinformatics*: Springer, 2014, pp. 55-68.

[9] A. Mockus, R. T. Fielding, and J. D. Herbsleb, "Two case studies of open source software development: Apache and Mozilla," *ACM Transactions on Software Engineering and Methodology,* vol. 11, no. 3, pp. 309-346, 2002.

[10] S. Lussier, "New Tricks: How Open Souce Changed the Way My Team Works," *IEEE Software,* vol. 21, no. 1, pp. 68-72, 2004.

[11] B. Dagenais, H. Ossher, R. K. E. Bellamy, M. P. Robillard, and J. P. deVries, "Moving into a new software project landscape," in *Proc. of the 32 Int'l Conf. on Software Engineering (ICSE '12)*, ape Town, South Africa, 2010, vol. 1, pp. 275-284.

[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)*, 2003, pp. 419-429.

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

[14] S. P. Borgatti and M. G. Everett, "Models of core/periphery structures," *Social Networks,* vol. 21, no. 4, pp. 375-395, 2000.

[15] L. Dabbish, C. Stuart, J. Tsay, and J. Herbsleb, "Social coding in GitHub: transparency and collaboration in an open software repository," in *the conference on Computer Supported Cooperative Work*, Seattle, WA, USA, 2012, pp. 1277-1286.

[16] 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, 2014, pp. 1-10.

[17] C. Amrit and J. van Hillegersberg, "Exploring the impact of socio-technical core-periphery structures in open source software development," *Journal of Information Technology,* vol. 25, no. 2, pp. 216-229, 2010.

[18] 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,* vol. 52, no. 3, pp. 296-303, 2010.

[19] E. S. Raymond, "The cathedral and the bazaar," ed, 1999.

[20] M. Zhou and A. Mockus, "Who Will Stay in the FLOSS Community? Modeling Participant's Initial Behavior," *IEEE Transactions on Software Engineering,* vol. 41, no. 1, pp. 82-99, 2015.

[21] 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,* vol. 89, pp. 128-145, 2014.

[22] S. Wasserman and K. Faust, *Social network analysis: Methods and applications*. Cambridge university press, 1994.

[23] S. P. Borgatti, M. G. Everett, and J. C. Johnson, *Analyzing social networks*. SAGE Publications Limited, 2013.

[24] M. McPherson, L. Smith-Lovin, and J. M. Cook, "Birds of a feather: Homophily in social networks," *Annual review of sociology,* vol. 27, no. 1, pp. 415-444 %@ 0360-0572, 2001.

[25] G. C. Kane, M. Alavi, G. J. Labianca, and S. Borgatti, "What's different about social media networks? A framework and research agenda," 2012.

[26] M. E. J. Newman, "The structure and function of complex networks," *SIAM review,* vol. 45, no. 2, pp. 167-256 %@ 0036-1445, 2003.

[27] M. E. J. Newman, "Analysis of weighted networks," *Physical review E,* vol. 70, no. 5, p. 056131, 2004.

[28] R. S. Burt, "Positions in networks," *Social forces,* vol. 55, no. 1, pp. 93-122 %@ 1534-7605, 1976.

[29] R. S. Burt, *Structural holes: The social structure of competition*. Harvard university press, 2009.

[30] G. Madey, V. Freeh, and R. Tynan, "The open source software development phenomenon: An analysis based on social network theory," *AMCIS 2002 Proceedings,* p. 247, 2002.

[31] K. Crowston and J. Howison, "The Social Structure of Free and Open Source Software Development," *First Monday,* vol. 10, no. 2, 2005.

[32] C. Bird, D. Pattison, R. D'Souza, V. Filkov, and P. Devanbu, "Latent Social Structure in Open Source Projects," in *Proc. of the 16th Int'l Symp. on Foundations of Soft. Eng. (FSE' 08)*, Atlanta, Georgia, 2008, pp. 24-35.

[33] C. De Souza, J. Froehlich, and P. Dourish, "Seeking the source: software source code as a social and technical artifact," 2005, pp. 197-206 %@ 1595932232: ACM.

[34] GitHub. (2017, 21-06-2018). *The State of the Octoverse 2017*. Available: https://octoverse.github.com/

[35] N. Kerzazi and I. El Asri, "Who Can Help to Review This Piece of Code?," 2016, pp. 289-301: Springer.

[36] P. Bholowalia and A. Kumar, "EBK-means: A clustering technique based on elbow method and k-means in WSN," *International Journal of Computer Applications,* vol. 105, no. 9 %@ 0975-8887, 2014.

[37] C. Oezbek, L. Prechelt, and F. Thiel, "The onion has cancer: Some social network analysis visualizations of open source project communication," 2010, pp. 5-10 %@ 1605589780: ACM.

[38] V. Batagelj and M. Zaversnik, "An O (m) algorithm for cores decomposition of networks," *arXiv preprint cs/0310049,* 2003.

[39] M. Joblin, "Structural and Evolutionary Analysis of Developer Networks," University of Passau, 2017.

[40] A. Terceiro, L. R. Rios, and C. Chavez, "An empirical study on the structural complexity introduced by core and peripheral developers in free software projects," 2010, pp. 21-29 %@ 1424489172: IEEE.

[41] A. Mockus, R. T. Fielding, and J. Herbsleb, "A case study of open source software development: the Apache server," 2000, pp. 263-272 %@ 1581132069: Acm.

[42] M. Joblin, S. Apel, C. Hunsen, and W. Mauerer, "Classifying developers into core and peripheral: An empirical study on count and network metrics," 2017, pp. 164-174 %@ 1538638681: IEEE Press.