Received July 14, 2018, accepted August 22, 2018, date of publication September 3, 2018, date of current version October 8, 2018.

Digital Object Identifier 10.1109/ACCESS.2018.2868427

Empirical Study on the Evolution of Developer Social Networks

MOHAMED ABDELRAHMAN ALJEMABI[1,2] AND ZHONGJIE WANG[1], (Member, IEEE)
[1]School of Computer Science and Technology, Harbin Institute of Technology, Harbin 150001, China
[2]Faculty of Mathematical and Computer Science, University of Gezira, Wad Madani 21111, Sudan

Corresponding author: Zhongjie Wang (rainy@hit.edu.cn)

This work was supported in part by the National Key Research and Development Program of China under Grant 2017YFB1400604 and in part by the National Science Foundation of China under Grant 61772155, Grant 61832014, Grant 61832004, and Grant 61472106.

ABSTRACT Software development is incredibly complex. Specifically, open-source software (OSS) development requires developers to collaborate with each other to conduct their work. Because software systems are evolving with time, collaboration among software developers may affect the quality of evolved software. The OSS developer teams collaborate in various tasks, including communications, coordination, and making various social collaboration in the OSS projects (e.g., bug/issue report, discussion, code revisions, and so on) without access restriction, and all these activities are used to generate an implicit developer social network (DSN). The DSN that is based on a bug tracking system is one of the most important DSNs that reflect the real collaboration between developers. As the software system evolves, the DSN evolves. This paper describes an empirical study of the evolution of DSNs on OSS projects collected from GitHub. Four perspectives were used: social network analysis, DSN as an ecosystem, community evolution patterns, and the core-periphery structure. The results demonstrated the DSNs over time have followed the power law degree distribution with $+1\%$ or more as an increasing rate to be more fitting over time. DSNs over time are considered a small-world community. DSNs over time exhibits about 55% diversity with 75% of evenness between the developers to contribute in different OSS projects in the same environment. Moreover, DSNs over time have a few developers as core members and large developers as peripheral members. Finally, about 10% of developers changed their positions frequently over time.

INDEX TERMS Software engineering, open source software, software quality, software maintenance, bug tracking systems, developer social network, evolution.

I. INTRODUCTION

The complexity of software systems necessitates methods for effective software development [1], [2]. Over the last two decades, researchers have become increasingly interested in collaborative software development. Collaborative software development represents a successful model of software development, as developers' communities collaborate on a voluntary basis. For example, under a collaborative model, developers of the software and users can submit bug reports and requests for changes [1].

A huge amount of data is generated during software development, as every version of the project's code is stored in version archives; every reported defect is saved in bug tracking systems; and every piece of communication is kept in email and forum archives. The data of all developers' activities is stored in software repository systems, including email systems, configuration management tools, and bug tracking systems [3] such as GitHub, Source Forge, and others [4]. Because software systems are evolving with time, the quality of evolved software must be ensured during this evolution. Recently, researchers have started to recognize the complexities of software development activities such as work dependencies [5], daily work routines [6], and social networks [7] which impact the quality of a software product [8]. Traces of these activities can be found in the repositories used by developers on a daily basis, such as version control systems VCS, bug tracking systems BTS, and email communication archives [8]. Thus, the developer team structure (or collaboration structure) may influence the quality of evolved software systems. The OSS developer teams conduct several tasks, including communications and coordination, engaging in various forms of social collaboration in the open source software projects (e.g., Bug/issue report, discussion, code revisions, etc.). All these activities can be recorded on software repositories like GitHub and used to generate an implicit developer social network (DSN) [4], [9].

The developer social network (DSN) is a tool for researching the social collaboration features of developers in OSS projects that facilitate software engineering. Studies of the DSN concern the construction of a DSN, the analysis of a DSN, and the application of a DSN for improving software engineering tasks [4], [9]. Recently, various types of DSNs with different features and methods have been constructed. The common types of DSN are Project Participation-based DSN (PP-DSN), Version Control System-based DSN (VCS-DSN), and Bug Tracking System-based DSN (BTS-DSN) as well as other DSNs such as Email-based DSN, Follow-based social network, and forked-based sharing system. The composition of a DSN consists of developers as nodes and links between developers as edges [4], [9], [10]. Because the developer team structure may influence the quality of evolved software over time, and developer team structures has changed over time, it is worthwhile to investigate the evolution of developer social networks (DSNs) and how this has affected the software development. One of the most important DSNs is BTS-DSN because, unlike other types of DSN, it reflects the developers' activities of software maintenance. Also, BTS-DSNs focus on specific objectives related to the software development process [9].

This study empirically investigates four perspectives that are related to developer social networks (DSN): the Social Network Analysis perspective (SNA), the ecosystem perspective, the community perspective, and the core-periphery structure perspective. SNA uses several measures to study and analyze social networks. The ecosystem perspective suggests that the natural ecosystem and the software ecosystem are similar in several aspects and thus uses natural ecosystem evolution to describe the evolution of developer social networks (DSN). This perspective includes the use of biodiversity measurements to show how DSNs have evolved over time. The community perspective suggests that communities have a set of developers who work on similar issues and share similar interest in DSNs. Because these communities change over time, this approach studies how DSNs have evolved over time from the community's perspective. The core-periphery structure perspective conceptualizes a developer social network (DSN) as a core-periphery structure which is composed of a small set of core developers and a large set of peripheral developers. The core-periphery structure perspective also uses this model to explain how DSNs evolve over time. The following describes how the perspectives were applied to study DSNs:

(1) Social network analysis (SNA) perspective: From the social network analysis perspective, a comparison between constructions of DSNs in each period of time and between evolved DSNs was conducted to determine the differences and the extent of those differences as well as to show the evolution of DSNs over time using social network metrics.

(2) Ecosystem perspective: The ecosystem perspective views a developer social network (DSN) as an ecosystem. The relationship between developer social networks and natural ecosystems were investigated, and biodiversity measurements were used to assess how DSNs evolve over time.

(3) Community perspective: The communities in DSN change over time; some big communities are split into smaller ones, some smaller communities are merged into bigger ones, some communities disappear, and so on. These phenomena were studied to determine evolution patterns in DSNs.

(4) Core-periphery Structure perspective: In different DSNs and at different times within the same DSN, a developer's position changes between a peripheral developer and being a core team member. These developer position changes in DSNs over time were classified.

The conclusions of the study are summarized as follows:

* More than 70% of the new coming developers have contributed in DSNs over time, and 30% of old developers had continued their contributions.

* There is 55% or more of diversity between developers who have contributed to different OSS projects in the same environments over time. However, the evenness (equitability) shows (70% -80%) between developers of different OSS projects in the same environment could be influenced by the density of developers over time and.

* DSNs over time exhibited all five community patterns. The percentage of each pattern differed among DSNs, as it depended on the collaboration between developers in OSS projects.

* Through the evolution of DSNs, less than 20% of the developers were core members and about 80% of the developers were peripheral members over time. However, more than 90% of the developers kept their position as a core or peripheral member over time and less than 10% of developers changed their position frequently over time.

The remainder of this paper is organized as follows. Section 2 introduces the developer social networks (DSNs), Social Network Analysis (SNA), DSNs as ecosystems, community evolution patterns, and the core-periphery structure. Section 3 presents the components of the empirical study: project selection with data preparation, research questions (RQs), and extracting a BTS-DSN. Section 4 presents the evolution of DSNs using the social network analysis perspective, the ecosystem perspective, the community perspective, and the core-periphery structure perspective. Section 5 analyses the threats to validity, and Section 6 presents the conclusions.

II. BACKGROUND

A. DEVELOPER SOCIAL NETWORK (DSNs)

A developer social network (DSN) is a tool for researching the features of social collaboration among developers in OSS projects; such analysis can facilitate the performance of software engineering tasks. Studies of developer social networks (DSN) concern the construction of a DSN, the analysis of a DSN, and the application of the DSN for improving software engineering tasks [4]. In past studies, several researchers have constructed various types of DSNs.

We list the most common types of DSN. First, The Project–participation based DSN (PP-DSN) based on participation relationship between developers in OSS projects [4]. Second, the VCS-DSN based on the code co-change relations among developers; This network is built depending on contribution based on common file change managed by version control system such as CVS, SVN, and Git, called version control systems based DSNs (VCS-DSNs) [10], [11]. Third: the follow-based social network involves developers receiving "follow-up", or updates, regarding their activities rather than maintaining personal relationships [4]. Fourth: the BTS-DSN is based on the bug report/comment/fix relationships among developers [12]. Particularly, the studies in [3] and [13] construct DSNs based on interactions among developers involving comments regarding the same bug issues. In [14] and [15], the construction of a direct DSN is based on interactions of "reply-to" comments and assignee and reassignee bug reports among developers. Hong *et al.* [3] studied the evolution of DSNs and compared DSNs with popular GSNs such as Facebook and Twitter using the SNA approach (e.g., power law, the degree of separation, modularity, and community size). Kumar and Gupta [13] studied the evolution of DSNs using different measurements: the number of contributors, average degree, average path length, average distance, clique size, and clustering coefficient. Cataldo and Herbsleb [16] demonstrated the evolution of DSNs by mapping the geographical distribution of the projects. Lim and Bentley [17] studied the evolving relationships between social networks and stakeholder involvement in software projects. Sharma and Kaulgud [18] investigated team evolution during a project testing phase using SNA techniques. Tsay *et al.* [19] detected the influence developers in Apache projects through the social network activities, by investigating email based DSN and using SNA and prediction model, and conclude that the social communications are a better predictors than patching activity. Gharehyazie *et al.* [20] examined the contribution decisions by using the social and technical information of Github through building a statistical model analyzing the association of different pull request, submitter and repository measures of contributions. They found that projects managers when evaluating pull requests can used information of the technical contribution practices of a pull request and the strength of the social connection between submitter and project manager. Teixeira *et al.* [21] studied the OpenStack ecosystems by investigated the role of group, sub-communities and business models by combines qualitative analysis of archival data and SNA visualization in VCS-DSN. Results show the collaboration within the ecosystem does not necessary affected by the competition for the same revenue model. Addition to the expected collaboration between developers from same firm did not hold within the OpenStack ecosystem.

B. SOCIAL NETWORK ANALYSIS (SNA)

In our study, we follow a social network analysis approach that includes centrality measurement, global metrics, metrics of community structure, and properties metrics. (1) Centrality measurement is used to measure the central location of nodes in the network and includes factors of degree, closeness, and betweenness centrality. Degree centrality is used in an undirected network to measure node centrality. (2) Global metrics measure the network as a whole using factors such as size of the network, the number of nodes and edges, diameter, and graph density. (3) Properties metrics, including a power law probability distribution, are used to describe social phenomena with a diagram that depicts the existance of the significant leaders in a DSN, and average path length. (4) Community structure or modularity is used to quantify the strength of a community structure [4], [22]. We used the most important metrics in SNA analysis to capture the high properties and detailed properties of DSNs to show how DSNs evolve over the time.

C. DSNs AS ECOSYSTEMS

This study uses the following defition of a software ecosystem : "a collection of software projects which are developed and evolve together in the same environment." This understanding indicates several parallels and similarities between natural ecosystems and software ecosystems [1], [23]. For instance, in a natural ecosystem, nature is the environment, while in a software ecosystem, the environment refers to the development environment, or the software and hardware tools used during the development process [24]. There are also some similarities between software ecosystems and developer social networks (DSN), especially in the social collaboration between developers to achieve different tasks for OSS projects. For instance, the main parts of a software ecosystem are also those of the developer social network: vendors, OSS projects, developers, and the development environment. Moreover, the collaborative and social aspects of contributor communities (users and developers) are important for software ecosystem consideration, especially because, like software projects, the contributor communities evolve over the time, as there is a clear relationship between the sustainability of an ecosystem and its social aspects [25]. However, this study draws three parallels between the evolution of developer social networks (DSN) and that of natural ecosystems: (1) the developers as the equivalent of the "individuals"; (2) the contributors to the OSS projects as the equivalent of the "living species"; and (3) the collaboration between contributors to perform their tasks on OSS projects as the equivalent of the "environment". We used biodiversity measurements to show how DSNs evolve over time, including a diversity index, richness, and evenness. A diversity index[1] (Di) is a quantitative measure that reflects the number of different elements (such as OSS projects) in a dataset and simultaneously considers how evenly the basic entities (such as developers) are distributed among those types. Species richness[2] or abundance (S) is the number of different species

[1]https://en.wikipedia.org/wiki/Diversity_index
[2]https://en.wikipedia.org/wiki/Species_richness

represented in an ecological community, landscape, or region. In our study, we consider the richness measurement as developer density (DD_j). The Shannon diversity index[3] (H) which is commonly used to characterize species diversity in a community, is used. Shannon's index (H) accounts for both abundance and evenness of the species. Species evenness[4] or Equitability (E) refers to how close each species in an environment is in number. The equitability value is distributed between 0 and 1, with 1 indicating complete evenness.

D. COMMUNITY EVOLUTION

Proposed by Lin *et al.* [26], there are five categories for the patterns of community evolution: extinct, emerge, merge, split, and derivation. (1) Extinction occurs when the post community set of a community contains no community, which implies that the developers have left or completely scattered to other communities. (2) Emergence happens when the prior community set of a community contains no community, which may signify the emergence of a new interest or area of bugs. (3) Merge describes when a community has at least two communities in its prior community set, which indicates that at least two communities have shared bugs and, therefore, common interest. (4) Split happens when a community has at least two communities in its post community set, which shows that an interesting discrepancy occurred in the single community. (5) The derivation pattern is a one-to-one correspondence between the prior and post community, which includes the following; (i) Expandation occurs when a community increases in size and its prior community set comprises only one community, which indicates that newcoming developers are being attracted to this community. (ii) Shrinking happens when a community decreases in size and its prior community set comprises only one community, which is evidence that the community's developers are leaving the project or taking other communities [3], [26].

E. CORE-PERIPHERY STRUCTURE

The organization of OSS development communities can be described as a core-periphery structure [27], [28], with a set of tightly connected core nodes and a set of more loosely connected periphery nodes [29]. The small group of core developers in the OSS projects significantly contribute to the development and evolution of the project for a relatively long time [27], [30]. The larger group of peripheral developers occasionally contributes to the project [27], [30]. Peripheral developers mostly interact with the core developers and rarely communicate with other peripheral developers [27]. Due to their significant contributions and higher level among of interactions, the core developers are the most reputed contributors in an OSS community [27], [30]. However, the groups of core contributors and peripheral developers change over time. Core contributors can become peripheral contributors and vice versa. Because of these constant shifts, the core-periphery structure provides a useful tool for studying the evolution of DSNs. Several studies have focused on characterizing the core-periphery structures of OSS development communities. Fielding was the first to describe the core developer group and their roles in the Apache project [27]. Later studies found the core groups of multiple development projects contained only 3-25% of the developers, and these developers contribute 40-90% of the code [27], [31]. While the number of core developers in a project may not be high, the members belonging to the core change over time [27], [31].

[3]https://en.wikipedia.org/wiki/Diversity_index
[4]https://en.wikipedia.org/wiki/Species_evenness

III. EMPIRICAL STUDY SETUP

A. PROJECT SETELCTIONS AND DATA PREPRATION

In our empirical study of the evolution of DSNs, we used data collected from the GitHub community, which has a strong reputation and interest to join among the majority of developers [11]. The selection of these projects was based on the 89 root projects on the dataset offered by GHTorrent [32], which contained detailed information on the social coding activities of about 89 root projects and their forks with data of watchers, commits, issues, pull requests, and comments. Because this data was enormous (around 4000 GB), we selected the data between April 2009 to October 2013, and we extracted DSNs from a series of subsequent four-month periods to examine DSN evolution.

Table.1 shows the general statistics about root OSS projects and projects that used in this study.

TABLE 1. General statistics about root OSS projects in dataset.

OSS Project	*#Developers*	*#Forks*	*#Bugs*	*#Commits*	*#Versions*
All	517081	108629	150362	682451	-
Homebrew	8539	6786	23916	26257	16
Node.js	806	4598	2308	9154	83
Tornado	378	1342	903	1921	20
Hiphop-php	436	837	1070	6283	7
PHP-SDK	49	1232	66	135	17
facebook-android-sdk	45	1280	78	448	8
folly	15	522	39	417	1
flockdb	48	100	98	746	21
gizzard	27	123	103	1326	45
finagle	126	290	126	2527	42
zipkin	34	133	302	1145	3

B. RESEARCH QUESTIONS

RQ1: How does a DSN evolve over time?

RQ2: Is it possible to study the evolution of DSNs as the evolution of a natural ecosystem? If yes, how do DSNs evolve over time, based on biodiversity measurements?

RQ3: Are there any evolution patterns in DSNs? How can these patterns be discovered?

RQ4: Is it possible to classify developers based on how their position in the DSN changes over time?

C. RESEARCH VARIABLES

This empirical study uses two types of variables: independent variables and dependent variables, as shown in Table 2.

TABLE 2. Dependent-independent variables.

Type	*used*	*Characteristic (Variable)*	*Formula*
Independent variables		*Project (P)*	$P=\{p_1,p_2,\ldots p_z\}$
		Issue (ISU)	$ISU_t=\{b_1,b_2,..,b_{is}\}$
		Issue Comments (IC)	$IC_t=\{c_1,c_2,..,c_{cm}\}$
		Number of nodes (V)	$V=\{v_1,v_2,\ldots.v_{\|V\|}\}$
		Number of edge (E)	$E=\{e_1,e_2,\ldots,e_{\|E\|}\}$
Dependent variables	RQ1	*Average Degree (AV)*	$AV=\frac{1}{n}\sum_{i=1}^{n} k_i$, k_i *is a degree of* v_i
		Av. Weight degree (WD)	$WD=\frac{1}{n}\sum_{i=1}^{n} w_i$, w_i *is weighted degree of* v_i
		Graph Density (GD)	$GD=2\|E\|/(\|V\|(\|V\|-1))$
		Modularity (Q)	$Q=\sum_i(e_{ii}-a_i^2)$
		Average Path Length (PL)	$PL=\sum_{vi>vj} L(vi,vj)/\frac{n(n-1)}{2}$
		Power law	$P(degree \geq k) = Ck^{-\theta}$
	RQ2	*Ecosystems (Ec)*	$Ec(P^{Ec},V^{Ec})$, $P^{Ec}=\{p_1,p_2,\ldots p_s\}$, $V^{Ec}=\{v_1,v_2,\ldots.v_N\}$
		Developer Density (DDj)	$DD_j=\frac{n_j}{N}, j=\{1,2,\ldots,s\}$
		Diversity Index (DI(Ec))	$DI(Ec)=1-\frac{\sum_{i=1}^{s} n_j(n_j-1)}{N(N-1)}$
		Shannon Index (H(Ec))	$H(Ec)=-\sum_{i=1}^{S}(DD_j)_i\ln(DD_j)_i$
		Evenness (E_H(Ec))	$E_H(\mathrm{Ec})=\frac{H(Ec)}{\ln S}$
	RQ3	*Set of Community*	$SC_k(G_t)$, G_t *is a DSN at t period of time, k is the number of community*
		Set of the prior community	$SC_k(G_{t-1})$, G_{t-1} *is a DSN at t - 1*
		Set of the post community	$SC_k(G_{t+1})$, G_{t+1} *is a DSN at t +1*
	RQ4	*Set of Core developers*	$V^{CD}=\{v_1,v_2,\ldots.v_{CD}\}$
		Set of Peripheral developer	$V^{PD}=\{v_1,v_2,\ldots.v_{PD}\}$
		Classified Developers	$CKI\ (G_t)=\ V^{CD}\ U\ V^{PD}$

D. EXTRACTING DSNs

In our empirical study, we chose the BTS-DSN as an example to examine the evolution of DSNs for several reasons. First, the relationship between developers is focused on a particular technical topic (e.g., bug report, comment, and bug fixing). Second, the BTS-DSN reflects the real collaboration between developers, unlike other DSNs. Third, the relationship between developers in a BTS-DSN are short-lived, but strong temporal relationship [9]. We used the following rules to construct the BTS-DSN: if two developers commented on the same bug issues, there is a link between them in the DSN; in a Bug report, if a reporter developer assigns a bug issue to an assignee developer for fixing, there is a link between the two developers in the DSN; if developer A replies on a bug report to developer B's comment, there is a connection from A to B in the DSN [4], [33].

Figure1 provides an illustrative example of BTS-DSN construction and evolution over time: (a) Suppose we have four developers (A, B, C, and D) who contributed to each bug issue (e.g., bug report, comment, bug fixing). Based on their collaboration relationship, we extract a developer social network, as shown in the (b) side of the figure, which indicates developers as nodes and links between developers as edges; edges have different thicknesses depending on the frequency of collaboration between the two developers regarding the various bug issues (weighted edge). (c) Shows how the DSNs, the communities, and the developer positions evolve over time.

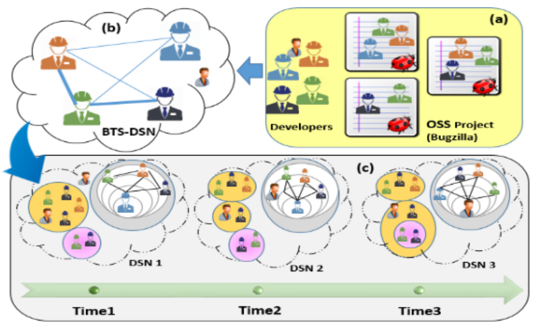

FIGURE 1. BTS-DSN construction and evolution.

IV. EMPIRICAL STUDY

A. SNA PERSPECTIVE

We constructed DSNs $G_{bts} = (V, E)$, directly using the data of OSS projects, P, collected from the dataset offered by GHTorrent [32], and constructed DSNs for each period of time: $G_{bts} = (V, E) = \{G_1, G_2, \ldots G_t\}$. The DSNs for each period of time were constructed using the sequence of issue data *ISU*; and the sequence of issue comments data is *IC*. Due to the large volume of data, we selected four of the successful OSS projects in the dataset that had different technology and extracted DSNs for each period of time. These OSS projects were Homebrew,[5] Hiphop-PHP,[6] Tornado,[7] and Node.[8] Table 3 provides the general information and statistics of the evolving DSNs in four OSS projects over time using SNA.

As shown in Table 3, the DSNs of our OSS projects evolve in the structure of the network over time, which reflects in the number of issues (ISU), the number of nodes (V), and the number of edges (E). All these parameters have changed over time and lead to an effect on the degree of developers (AV), the weighted developers' degree (WD), and other metrics. However, not all metrics behaved in this way. First, the degree of developers changed, as did the average degree of developers (AV) and the weighted degree and its average (WD) for all DSNs of the OSS projects.

The average degree (AV) and the average weighted degree (WD) can increase or decrease over time depending on the volume of the collaboration between developers, which is impacted by the release of new versions of the OSS project. Second, the modularity (*Q*) of the projects' evolved DSNs over time exhibits fluctuating increases, but at all periods of time the value of the modularity is above 0.3, which indicates a strong community structure [34]. Third, the Graph Density (GD) of DSNs for the four OSS projects over time ranged from 0.007 to 0.3. The sequence of all the OSS projects' graph

[5]https://github.com/mxcl/homebrew
[6]https://github.com/facebook/hiphop-php
[7]https://github.com/facebook/tornado
[8] https://github.com//joyent/node

TABLE 3. SNA ststistics and information for evolution of DSNs of four OSS projects.

OSS Project	Variable	DSNs over time (G_t)																
		$G_{3\text{-}200}$	$G_{4\text{-}200}$	$G_{1\text{-}201}$	$G_{2\text{-}201}$	$G_{3\text{-}201}$	$G_{4\text{-}201}$	$G_{1\text{-}201}$	$G_{2\text{-}201}$	$G_{3\text{-}201}$	$G_{4\text{-}201}$	$G_{1\text{-}201}$	$G_{2\text{-}201}$	$G_{3\text{-}201}$	$G_{4\text{-}201}$	$G_{1\text{-}201}$	$G_{2\text{-}201}$	$\mathbf{G_{3\text{-}2013}}$
Tornado	*ISU*	21	26	31	25	38	56	40	52	77	57	59	68	46	60	47	136	62
	V	19	22	28	17	13	26	24	27	47	30	27	33	24	32	33	57	25
	E	63	35	49	22	11	29	45	39	69	41	26	51	32	36	42	90	29
	AV	6.632	3.182	3.5	2.588	1.692	2.231	3.75	2.889	2.936	2.733	1.926	3.091	2.667	2.25	2.545	3.158	2.32
	WD	14.32	6.091	12.43	6.118	6.769	6.385	6.667	13.1	18.98	8.4	5.704	17.636	8.917	7.562	8.97	21.965	7.44
	Q	0.48	0.41	0.45	0.30	0.31	0.42	0.44	0.36	0.42	0.30	0.39	0.30	0.32	0.30	0.31	0.42	0.32
	GD	0.368	0.152	0.13	0.162	0.141	0.089	0.163	0.111	0.064	0.094	0.074	0.097	0.116	0.073	0.08	0.056	0.097
	PL	1.865	2.257	1.826	1.838	1.946	1.986	1.837	1.86	2.009	2.084	1.914	1.891	1.884	1.979	1.92	1.976	1.903
Homebrew	*ISU*	71	270	734	644	924	1096	1183	1284	1686	1476	2140	1952	2142	1639	2088	2143	2240
	V	69	196	363	306	463	529	614	754	1203	838	1082	1050	1220	798	965	1067	792
	E	246	577	1052	718	933	1264	1740	3561	6455	3577	3977	4268	4920	2843	2826	4998	2180
	AV	7.13	5.888	5.796	4.693	4.03	4.779	5.668	9.446	10.73	8.537	7.351	8.13	8.066	7.125	5.875	9.368	5.51
	WD	65.07	48.96	44.09	32.62	30.06	32.19	37.74	48.27	45.11	105.8	103.9	74.88	98.28	91.98	53.11	74.02	69.21
	Q	0.36	0.44	0.53	0.52	0.50	0.49	0.56	0.61	0.65	0.54	0.46	0.53	0.49	0.47	0.47	0.54	0.45
	GD	0.105	0.03	0.016	0.015	0.009	0.009	0.009	0.013	0.009	0.01	0.007	0.008	0.007	0.009	0.006	0.009	0.007
	PL	1.922	2.127	2.178	2.009	2.129	2.167	2.159	2.084	2.321	2.244	2.259	2.276	2.325	2.31	2.311	2.303	2.28
Hiphop-PHP	*ISU*	-	-	68	24	117	49	44	54	45	43	44	25	82	53	66	112	237
	V	-	-	55	21	28	28	25	40	62	44	29	29	59	50	55	59	82
	E	-	-	135	31	64	57	45	89	231	109	47	95	203	134	153	139	169
	AV	-	-	4.909	3.048	4.571	4.071	3.6	4.45	7.452	4.955	3.241	6.552	6.881	5.36	5.564	4.712	4.122
	WD	-	-	11.38	7.619	30	33.42	8.56	8.65	28.09	9.091	6.069	20.138	30.40	26.04	25.636	38.271	65.415
	Q	-	-	0.51	0.53	0.31	0.45	0.35	0.39	0.47	0.48	0.35	0.42	0.35	0.43	0.46	0.35	0.33
	GD	-	-	0.091	0.152	0.169	0.151	0.15	0.114	0.122	0.115	0.116	0.234	0.119	0.109	0.103	0.081	0.051
	PL	-	-	2.302	1.66	2.026	2.127	2.093	2.108	1.907	1.981	2.007	2.01	2.094	2.126	2.331	2.33	2.046
Node	*ISU*	-	-	-	-	42	119	91	150	181	207	247	200	147	168	294	235	206
	V	-	-	-	-	14	36	62	97	158	115	176	100	96	94	112	102	72
	E	-	-	-	-	15	67	133	270	1955	397	2146	272	231	245	318	366	236
	AV	-	-	-	-	2.143	3.722	4.29	5.567	24.75	6.904	24.38	5.44	4.812	5.213	5.679	7.176	6.556
	WD	-	-	-	-	16.14	10	11.19	22.35	197.3	50.47	83.08	33.62	31.23	32.27	54.16	45.65	53.11
	Q	-	-	-	-	0.31	0.47	0.46	0.39	0.32	0.37	0.41	0.39	0.33	0.33	0.31	0.34	0.31
	GD	-	-	-	-	0.165	0.106	0.07	0.058	0.158	0.061	0.139	0.055	0.051	0.056	0.051	0.071	0.092
	PL	-	-	-	-	1.881	2.079	2.332	2.386	2.103	2.131	2.169	2.166	2.079	2.07	2.179	2.04	2.021

density (GD) values decreased over time, thus making the DSNs more dense and complex over time. Fourth, the average path length (PL) evolved from 1.8 to 2.2 in Tornado, 1.9 to 2.3 in Homebrew, 1.6 to 2.3 in Hiphop-PHP, and 1.8 to 2.3 in Node. The average of the average path length (PL) was 1.94 in the Tornado project, 2.20 in the Homebrew project, 2.07 in the Hiphop-PHP project, and 2.12 in the Node project. Thus, the average path length (PL) of all four DSNs can be considered as relatively constant over the long term. The average path length slowly increases over time and most of the pairs of developers remain connected within two hopes, leading to evolved DSNs forming a small-scale community over time.

1) POWER-LAW

The power law distribution typically features a one-sided long tail with many large-valued outliers. In our study, we investigate the cumulative degree distribution of $(G = (V, E), P\,(degree \geq k) = Ck^{-\theta}$, where C is a positive constant and θ is an exponential parameter) DSNs for each period of time. Figure2 shows the evolution of the degree distribution for the DSNs of the four OSS projects, (a) Homebrew, (b) Node, (c) Hiphop-PHP, and (d) Tornado, in log-log scale axis. As shown in Figure2, the degree distribution of the projects' evolved DSNs followed the power law distribution over time on $\alpha = 0.01$. The degree distribution of developers in DSNs over time followed the power law distribution and evolved to fit more with the power law over time. This indicates that the DSNs of all four OSS projects have a large proportion of low-degree nodes and a very small proportion of high-degree nodes.

2) DEVELOPER CHANGES

The new developers participate in projects; the old developers still contribute to project; and other developers leave the projects or participate in other projects. Figure 3 shows the

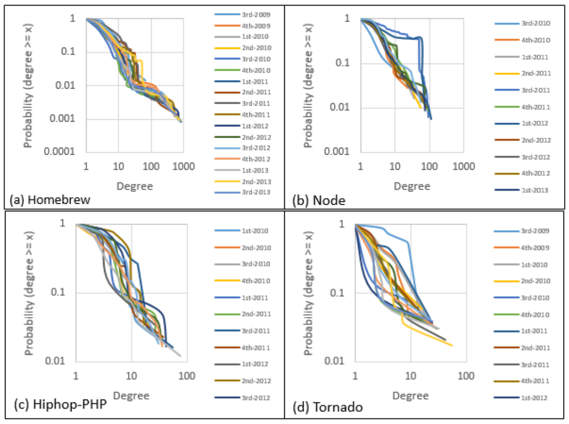

FIGURE 2. Cumulative Degree distribution evolution.

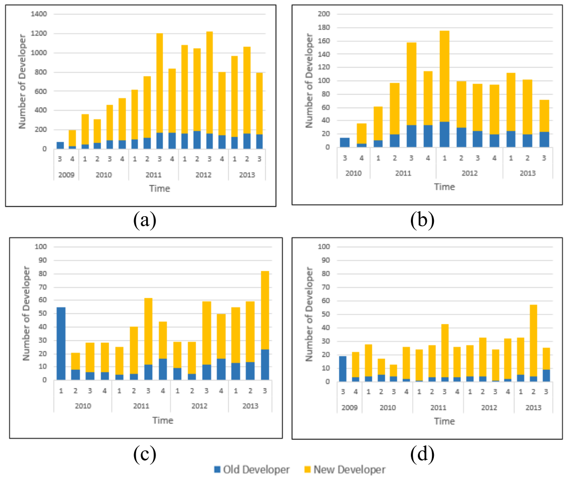

FIGURE 3. Number of active developers over time. (a) Homebrew. (b) Node. (c) Hiphop-PHP. (d) Tornado.

developer changes in the DSNs of the OSS projects over time in (a) Homebrew, (b) Node, (c) Hiphop-PHP, and (d) Tornado. For each column in this figure, the yellow bar indicates the number of new developers in that time period and the blue bar indicates the old developers who were active in that time period. We observed that the percentage of new coming developers is high in each period of time.

3) ANALYSIS SUMMARY

The results of analyzing the evolution of the DSNs from an SNA perspective are be summarized as follows:

(1) In the studied DSNs, less than 4% of developers with a high degree and 96% or more of developers with a low degree over time. Also, more than 70% of the new coming developers had contributed in DSNs over time, and 30% of old developers had continued their contributions. However, the evolved DSNs over time demonstrated tight-knit, and strong community structures.

(2) In the studied DSNs, the ratio of average path length increased slowly over time, as demonstrated by the mean of the average path length for the DSNs. The mean of the average path length was 2.20 in Homebrew, 2.07 in Hiphop-PHP, 2.12 in Tornado, and 1.97 in Node project. Thus, the DSNs over time are a small-scale community or ‘‘small-world’’.

B. ECOSYSTEMS PERSPECTIVE

In this section, we show the evolution of DSNs from an ecosystem perspective. Under this assumption, the evolution of DSNs parallels the evolution of natural ecosystems. So, we selected from our dataset all the developers who were working in the same environments. We chose Facebook and Twitter vendor repositories as an example to conduct our empirical study which contained five different OSS projects in Facebook and four OSS projects in Twitter as root projects and other forked projects. We used the original projects to show how DSNs can evolve over time.

Figure 4. (a) depicts how Facebook joined into the Github repository in April 2009 and established their projects in 2009 with the Tornado[9] project using the Python programming language. In 2010, the Tornado project grew; in that same year, three new OSS projects appeared in the Facebook environment. These projects were Hiphop-PHP[10] using C++ programming language, PHP-SDK[11] using PHP programming language, and Facebook-android-SDK[12] using JAVA programming language. In 2011, interactions among developers on each OSS project grew, and some developers collaborated on two projects. In 2012, Folly[13] which used C++ programming language appeared. The number of developers on each project was growing and, in 2013, some projects had less contributors, and some of the developers began to collaborate with others on different projects. Figure4. (b) shows how Twitter joined into the Github repository in January 2009 and established their projects with Flockdb[14] and Gizzard[15] in April 2010, Finagle[16] on October 2010, and all OSS projects with the Scala programming language.

[9]https://github.com/facebook/tornado
[10]https://github.com/facebook/hiphop-php
[11]https://github.com/facebook/php-sdk
[12]https://github.com/facebook/ facebook-android-sdk
[13]https://github.com/facebook/folly
[14]https://github.com/twitter/flockdb
[15]https://github.com/twitter/gizzard
[16]https://github.com/twitter/finagle

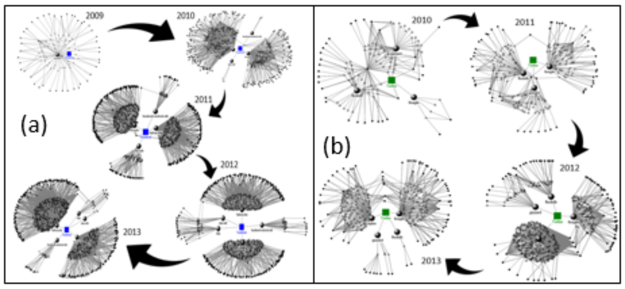

FIGURE 4. Evolution of DSNs in (a) Facebook, and (b) Twitter environment.

In 2011, the developers were working together on different OSS projects. In 2012, the Zipkin[17] project also appeared, resulting from the growth of its developers through collaborations on the Finagle project. In 2013, the collaboration between developers of the Zipkin and Finagle projects increased, but the collaboration between developers on other projects decreased, thus rendering these projects inactive.

1) BIODIVERSITY MEASUREMENTS

In this section, we investigate the evolved DSNs in Facebook and Twitter environments through biodiversity measurements, and we examine the developer density in each OSS project over time in addition to studying the evenness and diversity of developers over time. Figure 5 shows the evolution of developer density (DD_j) of OSS projects in Facebook and Twitter environments over time. Each column provides the percentage of the density of developers (i.e., the abundance of developers) of each project in each period. Figure 5. (a) shows the developer densities of OSS projects in the Facebook environment over time. The highest concentration of the developers' density was between the Tornado and the Hiphop-PHP OSS projects, with lower percentages of developer density in other OSS projects in the Facebook environment.

Also, in some columns, some OSS projects (e.g., PHP-SDK) did not exist, and after investigating, we found that these projects are inactive, deprecated, and have no contributors (i.e., extinct). Figure5.(b) shows the developer densities of OSS projects in the Twitter environment over time. The highest concentration of developer density was firstly distributed between the Flockdb and Gizzard projects, secondly between the Flockdb and Finagle projects, and thirdly between the Finagle and Zipkin projects, with lower percentages of developer densitites of other OSS projects in the Twitter environment.

Table 4 shows the results of the biodiversity measures of DSNs in the Facebook and Twitter environments in each period of time. These measures include the diversity index ($D_I(EC)$), the Shannon diversity index ($H(EC)$), evenness ($E_H(EC)$), the number of OSS projects in the ecosystem (P^{EC}), and the number of developers in the ecosystem (V^{EC}).

[17]https://github.com/twitter/zipkin

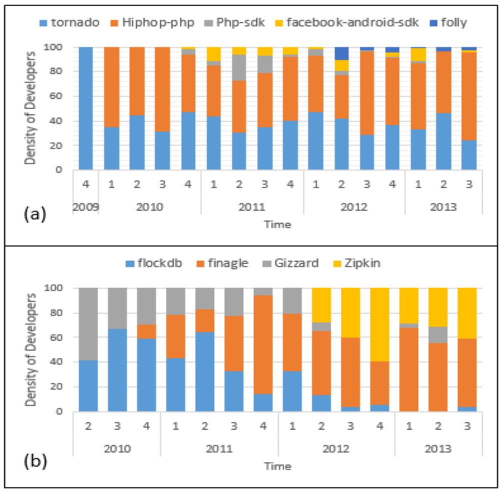

FIGURE 5. Evolution of developers' density in (a) Facebook, and (b) Twitter environment.

Using these measures, we traced changes in values to demonstrate the evolution of DSNs over time. First, the diversity index ($D_I(EC)$) of developers in the Facebook environment changed from 0.44 to 0.69 over time, and in the Twitter environment, the diversity index changed from 0.34 to 0.67 over time. Second, the evenness value ($E_H(EC)$) of developers in the Facebook environment changed from 0.54 to 0.99. In the Twitter environment, the evenness value changed from 0.56 to 0.97. The highest value of evenness ($E_H(EC)$) indicated complete evenness between developers in each OSS project (the equality distribution of developers' density in each OSS project). The reason for these changes was that most developers focused on particular OSS projects over others. For example, in the second quarter of 2010, the DSNs of the Facebook environment had owned two OSS projects, the value of $D_I(EC)$ was 0.51, and the value of $E_H(EC)$ was 0.99, meaning these projects had the same chance for developers to contribute, and these developers had the same density in each OSS project.

Figure 6 details the evolution of the diversity and evenness indices (actual and predict) to explain the changing trends of these measurements over time. When we traced the change in evenness trends in the Twitter environment and compared

TABLE 4. Biodiversity measurements for developers in the Facebook environment.

Environmen	Biodiversit	2009	2010				2011				2012				2013		
		4	1	2	3	4	1	2	3	4	1	2	3	4	1	2	3
Facebook	$D_I(EC)$	0	0.46	0.51	0.44	0.57	0.63	0.69	0.67	0.57	0.58	0.69	0.46	0.57	0.59	0.53	0.44
	$E_H(EC)$	0	0.93	0.99	0.90	0.66	0.79	0.89	0.86	0.69	0.68	0.81	0.55	0.62	0.64	0.74	0.54
	$H(EC)$	0	0.65	0.69	0.63	0.92	1.09	1.23	1.19	0.96	0.94	1.30	0.76	0.99	1.03	0.81	0.75
	P^{EC}	1	2	2	2	4	4	4	4	4	4	5	4	5	5	3	4
	V^{EC}	45	86	38	44	64	60	98	142	87	68	83	87	93	102	123	115
Twitter	$D_I(EC)$	-	-	0.53	0.48	0.58	0.67	0.55	0.67	0.34	0.65	0.65	0.54	0.55	0.47	0.58	0.54
	$E_H(EC)$	-	-	0.97	0.91	0.84	0.96	0.81	0.96	0.56	0.95	0.83	0.74	0.77	0.67	0.87	0.74
	$H(EC)$	-	-	0.67	0.63	0.92	1.06	0.89	1.06	0.62	1.05	1.15	0.82	0.84	0.74	0.95	0.81
	P^{EC}	-	-	2	2	3	3	3	3	3	3	4	3	3	3	3	3
	V^{EC}	-	-	12	12	17	23	17	18	54	33	29	25	17	28	45	27

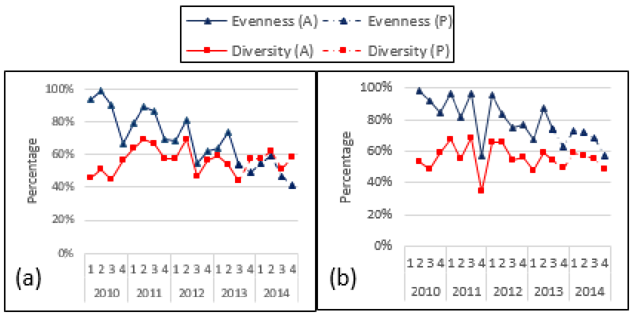

FIGURE 6. **Evolution of diversity and evenness indices of evolved DSNs in (a) Facebook, and (b) Twitter environments.**

this change to the Facebook environment, we found that the evenness between developers who collaborate through OSS projects in the Twitter environment is greater than the evenness in the Facebook environment.

Also, the collaboration between developers through OSS projects in the Facebook environment has resulted in in greater developer strength to work on specific projects over time. In the Twitter environment, the collaboration between developers through OSS projects has resulted in more collaboration between developers to work on OSS projects over time.

2) ANALYSIS SUMMARY

The results of analyzing the evolution of DSNs from the ecosystem perspective are as follows:

(1) The ratio of developer density changed over time. In the Facebook environment, there was a high percentage of developers' density between the Tornado and Hiphop-PHP projects over time, while the other projects demonstrated a low percentage over time. In the Twitter environment, the percentage of developer density was distributed between projects over time.

(2) The ratio of the diversity index over time exhibited convergences of 56% and 55% in the Facebook and Twitter environments, respectively. However, the ratio of the evenness index over time differs between developers in the two environments, as the Facebook environment exhibits 70% evenness while the Twitter environment exhibits 83% evenness. This difference is influenced by the ratio of the developer density in each environment over time.

C. COMMUNITY PERSPECTIVE

1) COMMUNITY EVOLUTION PATTERNS

As mentioned in Section 2.4, there are five categories of the community evolution: derivation, merge, split, extinct, and emerge [3], [26]. In our empirical study of the evolution of DSNs, we all these community evolution patterns were detected.

Figure 7 represents the evolution of the communities of all the OSS projects' DSNs between 2nd 2009 to 3rd 2013. In Figure 7, each node represents a community, and each edge

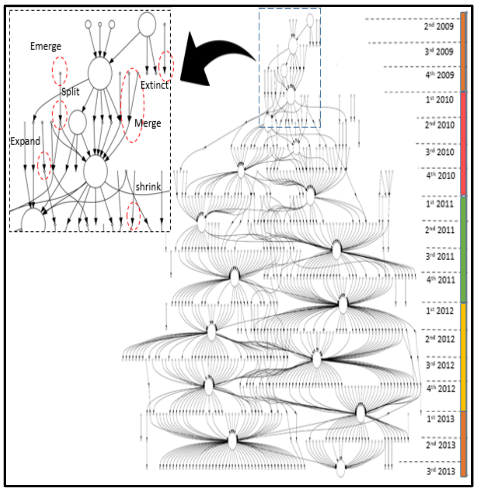

FIGURE 7. **Evolution of DSNs community.**

represents an evolving relationship between two communities. In each node, the radius is proportional to the number of developers. The edge starts horizontally from top to down communities. Also, Figure7, presents a zooming piece of the evolved communities which clearly describes the community evolution patterns. We investigated these patterns to show changes in the DSN communities of the four OSS projects (Hiphop PHP, Tornado, Node, and Homebrew).

Figure 8 shows the ratios of community evolution patterns over time for these OSS projects. As noted in the figure, all five patterns appeared in the DSNs of the four OSS projects. As shown in Figure 8. (a), the most frequent pattern in the Homebrew projects was "extinct", which may indicate that a lot of bug issues are found, and the developers are quick to fix these issues. Addition to some of the newcomer developers left the project. Figure 8. (b) shows that the most frequent pattern in the Node project is "derivation", with a

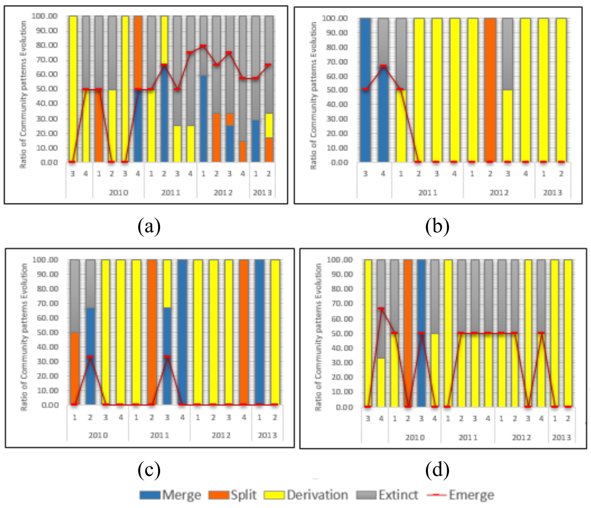

FIGURE 8. **Community evolution patterns for four OSS projects. (a) Homebrew. (b) Node. (c) Hiphop-PHP. (d) Tornado.**

low percentage of the "emerge" pattern over time, which may describe the interaction between developers who are working together over time. Figure 8. (c) shows that the "split" and "merge" patterns exist between developers in the community, which indicates that these developers can interact with bug issues as a specialist working group. Figure 8. (d) depicts how the developers of Tornado and communities interact as working groups and then work together as one group of developers to interact with bug issues. Moreover, some of the newcomer developers left the project.

2) ANALYSIS SUMMARY

The results of analyzing the evolution of the DSNs evolution using the community perspective are summarized as follows:

(1) The DSNs exhibited all five community evolution patterns as well as the three sub-types of the derivation pattern. The studied DSNs displayed different results for each pattern. The ratio of the emerge pattern was 49.64% in Homebrew, 29.17% in Tornado, 13.89% in Node, and 4.76% in Hiphop-PHP. The ratio of the extinct pattern was 49.30% in Homebrew, 29.17% in Tornado, 11.11% in Node, and 5.95% in Hiphop-PHP. The ratio of the merge pattern was 23.81% in Hiphop-PHP, 14.39% in Homebrew, 13.89% in Node, and 6.25% in Tornado. The ratio of the split pattern was 17.86% in Hiphop-PHP, 10.79% in Homebrew, 8.33% in Node, and 6.25% in Tornado projects. The ratio of the derivation pattern was 66.67% in Node, 58.33% in Tornado, 52.38% in Hiphop-PHP, and 28.13% in Homebrew.

D. CORE-PERIPHERY STRUCTURE PERSPECTIVE

In general, a social network that has a core-periphery structure is composed of a set of heavily connected core nodes and a set of more weakly connected periphery nodes [29]. Here, we show how DSNs can be portrayed in a core-periphery structure and demonstrate how DSNs evolve over time using the core-periphery structure perspective.

1) CORE-PERIPHERY STRUCTURE

First, we classified each OSS project in each period of time to the core-peripheral structure through Core Identification using the k-means (CKI) approach [27], which is based on six centrality scores (degree, betweenness, closeness, eigenvector, PageRank, and eccentricity). Second, in each period of time we investigated the percentage of the core developers and the peripheral developers of the evolved DSNs.

Figure 9 displays the percentages of the core developers and the peripheral developers in each period of time for the DSNs of the four OSS projects: (a) Homebrew, (b) Node, (c) Hiphop-PHP, and (d) Tornado. As we noted in Figure 9, the percentage of core developers was between 1% and 19% of developers over time, while the percentage of peripheral developers was between 99% and 81% of developers over time. In general, in the DSNs of all the projects, the percentage of the core developers was low, while the percentage of the periphery developers was high percentage over time. All results of the individual DSNs in the specific time period are

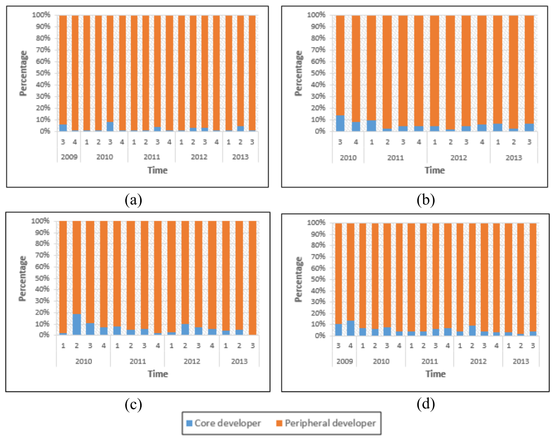

FIGURE 9. **Evolution of core-periphery structure percentage over time. (a) Homebrew. (b) Node. (c) Hiphop-PHP. (d) Tornado.**

consistent with other previous studies. Thus, we conclude that the evolved DSNs over time have a set of a few developers as the core members and a set of several developers as the peripheral developers.

2) THE CHANGE OF DEVELOPERS' POSITION

In this section, we used the results of the previous section to investigate the change in developers' positions. We examined the change in developers' positions deeply using the CKI approach [27] in each evolved DSN over time ($G_t = (V_t, E_t)$, $CKI\,(G_t) = V^{CD}UV^{PD}$). Figure10 presents the heat chart of a sample of developers in the DSNs of (a) Homebrew, (b) Node, (c) Hiphop-PHP, and (d) Tornado. In Figure10, the bold red indicates the developers at the core position, and the light red shows the developer in the peripheral position, but the white space accounts for the absent contribution of developers in a specific period of time.

As noted in figure 10, some developers changed their position from a peripheral member to a core member in the project and vice versa; moreover, some developers left the project and contributed again as a peripheral or core member, and some developers contributed to the project as a core member the entire time. Thus, the developers in each DSN have changed their position from core to peripheral members and vice versa over time. Also, we observed that some developers remained stable as a core member or a peripheral member and some changed their position between the two consecutive periods. We noted that in the Homebrew project, the average ratio of keeping stable developers was 96.14%, and the average ratio of the frequently changing developers was 3.86%. In the Tornado project, the average ratio of keeping stable developers was 94.20%, and the average ratio of the frequently changing developers was 5.80%. In the Hiphop-PHP project, the average ratio of the keeping stable developers was 91.34%, and the average ratio of the frequently changing developers was 8.66%. In the Node project, the average ratio of the keep stable developers was 87.35%, and the average ratio of the frequently changing developers was 12.65%.

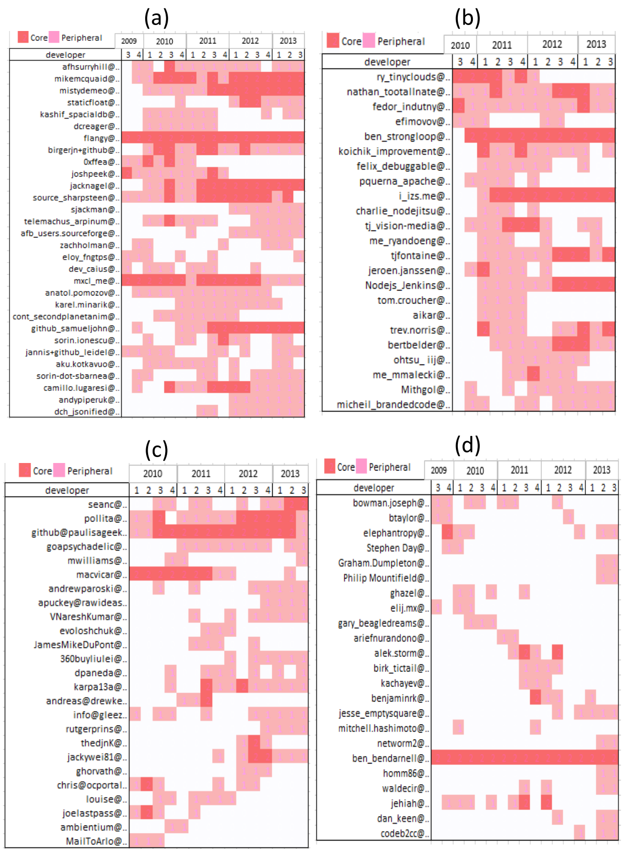

FIGURE 10. Change of developers' position in the projects. (a) Homebrew. (b) Node. (c) Hiphop-php. (d) Tornado.

3) ANALYSIS SUMMARY

The results of analyzing DSN's evolution using the core-periphery structure perspective is summarized as follows:

(1) In the DSNs, less than 20% of the developers were core members and about 80% of the developers were peripheral members over time.

(2) In the DSNs, more than 90% of the developers kept their position as a core or peripheral member over time and less than 10% of developers changed their position frequently over time.

V. THREATS TO VALIDITY

(1) Construct validity: In this work, we conducted analysis only to study the evolution of DSNs, and we used BTS-DSN/ All independent variables were extracted from bug reports with comments on bug issues by developers. However, besides bug reports, developers have other types of contributions such as documentation, commenting on others' work, project management, and so on. Performing the analysis solely on bug reports with comments ignores these contributions, thus threating the validity of the study's results.

(2) Internal validity: In our work, to construct DSNs over time, we used the Github dataset directly, and there were no significant threats to internal validity. In the SNA analysis to show the evolution of DSNs, the results may have changed depending on the type of DSN. Also, in studying the evolution of DSNs from the ecosystem perspective, there were changes depending on the type of DSN. In the community perspective, there were also changes that depended on the type of DSN. In the core-periphery structure perspective, there were no significant threats to internal validity.

(3) External validity: In this work, we conducted analysis only on the GitHub project, which has been well studied in prior research. We chose 11 of 89 OSS root projects that were considered famous, mature, and successful; these projects have relatively differentiated functionality scales as well as diversified structures, team sizes, and contribution forms, so they are adequately representative of other projects. We used the GHTorrent dataset to extract necessary data for constructing the BTS-DSN. It is possible that this study's results may not generalize to other projects. However, our methodology for analysis can be easily applied to analyse other OSS projects.

VI. CONCLUSIONS AND FUTURE WORK

We conducted an empirical study of the evolution of DSNs to show how DSNs evolve over time using four different perspectives: Social Network Analysis perspective (SNA), the ecosystem perspective, the community evolution patterns perspective, and the core-periphery structure perspective. In the SNA perspective, we examined the evolution of DSNs in different periods of time. We show that in each period of time, the DSNs follow a power law degree distribution; the community structure for each DSN in each period of time exhibited a strong community, and the DSNs in each period of time were "small-world". Using the ecosystem perspective, we examined the evolution of DSNs using the biodiversity measurements, such as the density of developers, a diversity index, and the evenness index of DSNs in each period of time, as a novel approach to studying the evolution of DSNs. Using the community perspective, we found that the communities of the DSNs manifested all five types of evolution pattern (emerge, split, merge, extinct, and derivation. Finally, using the core-periphery structure perspective, we investigated the changes in developers' positions over time. In future work, we will use the results of the current study as well as other references to develop the notion of "social collaboration patterns mining".

REFERENCES

[1] T. Mens, M. Claes, P. Grosjean, and A. Serebrenik, "Studying evolving software ecosystems based on ecological models," in *Evolving Software Systems*. Berlin, Germany: Springer, 2014, pp. 297–326.

[2] I. Kwan, A. Schroter, and D. Damian, "Does socio-technical congruence have an effect on software build success? a study of coordination in a software project," *IEEE Trans. Softw. Eng.*, vol. 37, no. 3, pp. 307–324, May/Jun. 2011.

[3] Q. Hong, S. Kim, S. C. Cheung, and C. Bird, "Understanding a developer social network and its evolution," in *Proc. 27th IEEE Int. Conf. Softw. Maintenance (ICSM)*, Sep. 2011, pp. 323–332.

[4] W. Zhang, L. Nie, H. Jiang, Z. Chen, and J. Liu, "Developer social networks in software engineering: Construction, analysis, and applications," *Sci. China Inf. Sci.*, vol. 57, no. 12, pp. 1–23, 2014.

[5] M. Cataldo, A. Mockus, J. A. Roberts, and J. D. Herbsleb, "Software dependencies, work dependencies, and their impact on failures," *IEEE Trans. Softw. Eng.*, vol. 35, no. 6, pp. 864–878, Nov./Dec. 2009.

[6] J. Śliwerski, T. Zimmermann, and A. Zeller, "When do changes induce fixes?" in *ACM SIGSOFT Softw. Eng. Notes*, vol. 30, no. 4, pp. 1–5, 2005.

[7] T. Wolf, A. Schroter, D. Damian, and T. Nguyen, "Predicting build failures using social network analysis on developer communication," in *Proc. 31st Int. Conf. Softw. Eng.*, 2009, pp. 1–11.

[8] N. Bettenburg and A. E. Hassan, "Studying the impact of social interactions on software quality," *Empirical Softw. Eng.*, vol. 18, no. 2, pp. 375–431, 2013.

[9] M. A. Aljemabi and Z. Wang, "Empirical study on the similarity and difference between VCS-DSN and BTS-DSN," in *Proc. Int. Conf. Manage. Eng., Softw. Eng. Service Sci.*, Wuhan, China, 2017, pp. 30–37.

[10] Y. Tymchuk, A. Mocci, and M. Lanza, "Collaboration in open-source projects: Myth or reality?" in *Proc. 11th Work. Conf. Mining Softw. Repositories*, 2014, pp. 304–307.

[11] A. Jermakovics, A. Sillitti, and G. Succi, "Exploring collaboration networks in open-source projects," in *Open Source Software: Quality Verification—OSS*. Berlin, Germany: Springer, 2013.

[12] T. Zhang and B. Lee, "An automated bug triage approach: A concept profile and social network based developer recommendation," in *Intelligent Computing Technology*. Berlin, Germany: Springer, 2012, pp. 505–512.

[13] A. Kumar and A. Gupta, "Evolution of developer social network and its impact on bug fixing process," in *Proc. 6th India Softw. Eng. Conf.*, 2013, pp. 63–72.

[14] J. Xuan, H. Jiang, Z. Ren, and W. Zou, "Developer prioritization in bug repositories," in *Proc. 34th Int. Conf. Softw. Eng. (ICSE)*, Jun. 2012, pp. 25–35.

[15] K. Crowston and J. Howison, "The social structure of free and open source software development," *First Monday*, vol. 10, no. 2, 2005. [Online]. Available: http://journals.uic.edu/ojs/index.php/fm/article/view/1478, doi: 10.5210/fm.v0i0.1478.

[16] M. Cataldo and J. D. Herbsleb, "Communication networks in geographically distributed software development," in *Proc. ACM Conf. Comput. Supported Cooperat. Work*, 2008, pp. 579–588.

[17] S. L. Lim and P. J. Bentley, "Evolving relationships between social networks and stakeholder involvement in software projects," in *Proc. 13th Annu. Conf. Genet. Evol. Comput.*, 2011, pp. 1899–1906.

[18] V. S. Sharma and V. Kaulgud, "Studying team evolution during software testing," in *Proc. 4th Int. Workshop Cooperat. Hum. Aspects Softw. Eng.*, 2011, pp. 72–75.

[19] J. Tsay, L. Dabbish, and J. Herbsleb, "Influence of social and technical factors for evaluating contribution in GitHub," in *Proc. 36th Int. Conf. Softw. Eng.*, 2014, pp. 356–366.

[20] M. Gharehyazie, D. Posnett, B. Vasilescu, and V. Filkov, "Developer initiation and social interactions in OSS: A case study of the apache software foundation," *Empirical Softw. Eng.*, vol. 20, no. 5, pp. 1318–1353, 2015.

[21] J. Teixeira, G. Robles, and J. M. González-Barahona, "Lessons learned from applying social network analysis on an industrial free/libre/open source software ecosystem," *J. Internet Services Appl.*, vol. 6, p. 14, Aug. 2015.

[22] M. S. Zanetti, I. Scholtes, C. J. Tessone, and F. Schweitzer, "Categorizing bugs with social networks: A case study on four open source software communities," in *Proc. Int. Conf. Softw. Eng.*, 2013, pp. 1032–1041.

[23] M. Lungu, "Towards reverse engineering software ecosystems," in *Proc. IEEE Int. Conf. Softw. Maintenance (ICSM)*, Sep./Oct. 2008, pp. 428–431.

[24] G. Bavota, G. Canfora, M. Di Penta, R. Oliveto, and S. Panichella, "The evolution of project inter-dependencies in a software ecosystem: The case of apache," in *Proc. ICSM*, Sep. 2013, pp. 280–289.

[25] C. R. B. de Souza, F. F. Filho, M. Miranda, R. P. Ferreira, C. Treude, and L. Singer, "The social side of software platform ecosystems," in *Proc. CHI Conf. Hum. Factors Comput. Syst.*, 2016, pp. 3204–3214.

[26] Y.-R. Lin, H. Sundaram, Y. Chi, J. Tatemura, and B. L. Tseng, "Blog community discovery and evolution based on mutual awareness expansion," in *Proc. IEEE/WIC/ACM Int. Conf. Web Intell.*, 2007, pp. 48–56.

[27] A. Bosu and J. C. Carver, "Impact of developer reputation on code review outcomes in OSS projects: An empirical investigation," in *Proc. 8th ACM/IEEE Int. Symp. Empirical Softw. Eng. Meas.*, 2014, Art. no. 33.

[28] T. T. Dinh-Trong and J. M. Bieman, "The FreeBSD project: A replication case study of open source development," *IEEE Trans. Softw. Eng.*, vol. 31, no. 6, pp. 481–494, Jun. 2005.

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

[30] Y. Ye and K. Kishida, "Toward an understanding of the motivation open source software developers," in *Proc. 25th Int. Conf. Softw. Eng.*, 2003, pp. 419–429.

[31] G. Robles, J. M. Gonzalez-Barahona, and I. Herraiz, "Evolution of the core team of developers in libre software projects," in *Proc. 6th IEEE Int. Work. Conf. Mining Softw. Repositories*, 2009, pp. 167–170.

[32] G. Gousios, B. Vasilescu, A. Serebrenik, and A. Zaidman, "Lean GHTorrent: GitHub data on demand," in *Proc. 11th Work. Conf. Mining Softw. Repositories*, 2014, pp. 384–387.

[33] A. Sureka, A. Goyal, and A. Rastogi, "Using social network analysis for mining collaboration data in a defect tracking system for risk and vulnerability analysis," in *Proc. 4th India Softw. Eng. Conf.*, 2011, pp. 195–204.

[34] H. Kwak, C. Lee, H. Park, and S. Moon, "What is Twitter, a social network or a news media?" in *Proc. 19th Int. Conf. World Wide Web*, 2010, pp. 591–600.

MOHAMED ABDELRAHMAN ALJEMABI received the B.S. degree in statistical and computer science and the M.S. degree in computer science from the University of Gezira, Wadmadani, Sudan, in 2007 and 2010, respectively. He is currently pursuing the Ph.D. degree in computer science with the School of Computer Science and Technology, Harbin Institute of Technology, Harbin, China.

From 2012 to 2014, he was a Lecturer with the Faculty of Mathematical and Computer Science, University of Gezira. His research interests include information systems, software engineering, and mining software repository.

ZHONGJIE WANG (M'08) received the B.S., M.S., and Ph.D. degrees in computer science from the School of Computer Science and Technology, Harbin Institute of Technology. He is currently a Professor with the School of Computer Science and Technology, Harbin Institute of Technology. He has authored two books more than 50 articles. His research interests include services computing, service engineering, Internet services and cloud, social network service, software engineering, software architecture, software evolution, social software engineering and crowdsourcing, and mining software repositories.

Dr. Wang is a Senior Member of the China Computer Federation (CCF), the Deputy Secretary of the Technical Committee of Services Computing (CCF), and a member of the Technical Committee of Software Engineering (CFF).