Heterogeneous Network Analysis of Developer Contribution in Bug Repositories

Wen Zhang[1, 3], Song Wang[1,4], Ye Yang[1, 2], Qing Wang[1, 2]
[1]Laboratory for Internet Software Technologies, Institute of Software, Chinese Academy of Sciences
[2]State Key Laboratory of Computer Science, Institute of Software, Chinese Academy of Sciences
[3]State Key Laboratory of Software Engineering of Wuhan University
[4]University of Chinese Academy of Sciences
{zhangwen, wangsong, yangye, wq}@nfs.iscas.ac.cn

***Abstract*—Using a bug repository, developers contribute to improve the quality of software incrementally by creating and updating bug reports. All the software artifacts in bug repositories are derived from developer contribution. Most prior studies on developer contribution in bug repositories bias on one particular form, e.g., commenting bug reports. However, in real practice of bug repositories, developers participate in and contribute to software projects via multiple ways, e.g., reporting new bugs, reopening incorrectly fixed bugs, commenting unfixed bug reports, and fixing unsolved bugs. In this paper, we exploit recent advances in analysis of heterogeneous network to avoid biased aspects in measuring developer contribution and explore multiple types of developer contribution in bug repositories. Further, we consider leveraging such multiple types of developer contribution to assist a typical prediction problem in bug repositories, i.e., bug triage. Empirical studies on bug repositories of Eclipse and Mozilla show that our approach can provide enriched knowledge of developer contribution to improve the resolution of bug triage. This study strongly suggests using the promising aspects of heterogeneous network can open many actionable insights in analyzing software repositories.**

Keywords-developer contribution; heterogeneous network; bug triage

I. INTRODUCTION

Bug report tracking systems (e.g., Bugzilla[1]) also known as bug repositories are widely adopted in software development to manage and track bugs efficiently. Within a bug repository, developers and users work together and make contribution continuously to improve the quality of software projects by reporting encountered problems, commenting bug reports to provide more detail information, and fixing bugs [1]. All the software artifacts in bug repositories are derived from developer contribution. Thus, understanding the structure of developer contribution could be helpful for fixing bugs and building success software.

Traditionally, metric LOC (lines of code) is used for measuring developer contribution in source code repositories [3], [6], [10]. Recently, as developers' social behaviors are more and more explicit in software development [5], [7], some social network metrics such as degree, betweenness, and closeness [9] are used to leverage developer or community contribution to improve software practice in software repositories. For example, [11], [12] build developer networks in change log repositories to improve software failure prediction; [13] prioritizes developers, [8] investigate developer communities via social network analysis in bug repositories.

Most of above studies employ social network techniques to construct developer networks bias on a particular form of developer contribution (e.g., in developer network of [8], a link between two developers denotes developers have commented a common bug). However, in real practice of software repositories, e.g., in a bug repository, developers contribute to software projects via multiple ways, e.g., reporting new bugs, reopening incorrectly fixed bugs, commenting unfixed bugs, and fixing unsolved bugs.

Since typical social network techniques cannot deal with multi-relation among nodes [4]. In this paper, we exploit recent advances in analysis of heterogeneous network [4] to explore multiple types of developer contribution in bug repositories. Heterogeneous network analysis has open many new insights in knowledge mining [22], social multi-relation analysis [23].

We introduce heterogeneous developer contribution network to model the multiple types of contribution from developers to components in software bug repositories. Different from traditional developer contribution network, e.g., Fig. 1 (b), a heterogeneous developer contribution network contains multiple types of objects, such as developers, bugs, comments, and components, as well as multiple types of links denoting different semantic relations among these objects. Moreover, objects in heterogeneous developer contribution network may connect with each other via different paths which denote different semantic relations between objects. For example, in Fig. 1 (a), a path between developer "*Bob*" and *Component1* as "*Bob- Comment1-Bug Report1- Component1*" means *Bob* comments a bug report of *Component1*. The two nodes also have another path as "*Bob-Bug Report2 - Component1*" denotes *Bob* creates a bug report in *Component1*. Both of these two paths denote *Bob* contributes to *Component1*, to distinguish the means of different paths between two objects, we propose to use meta path [4] (a path that connects objects via a sequence of relations) to denote multi-relation between two objects.

Further, we study 4 Research Questions (RQs) to explore the properties of multiple types of developer contribution and its application in bug repositories, i.e., improving the result of bug triage.

[1]http://www.bugzilla.org/

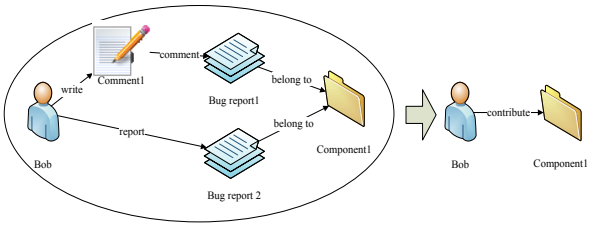


Figure 1. Examples of a simply heterogeneous developer contribution network (a) and a traditional developer contribution network (b).

Experiments on bug repositories of Eclipse and Mozilla show that our approach can assist bug triage in bug repositories, the average accuracy is improved by 43% on Eclipse and 28% on Mozilla by combining multiple types of developer contribution.

Our contribution can be summarized as follows:

1. We introduce heterogeneous network analysis to model multiple types of contribution behaviors of developers in bug repositories.

2. Using meta path-based approach in heterogeneous network, we measure and analyze multiple types of developer contribution in bug repositories.

3. We investigate how to improve a typical prediction problem, i.e., bug triage, by combining multiple types of developer contribution with machine learning techniques.

The remainder of this paper is organized as follows. Section II presents the background. Section III describes the methodology of extracting heterogeneous networks from bug repositories. Section IV analyzes multiple types of developer contribution in bug repositories. Section V presents the results of assisting bug triage by leveraging multiple types of developer contribution. Section VI states the threats. Section VII presents the related work. Section VIII summarizes this paper.

II. BACKGROUND

A. Bug Repositories

A bug repository is an issue tracking system adopted in software development and maintenance. Most software projects use bug repositories to manage their historical and incoming bugs. Bugs are recorded and managed in bug repositories in formation of bug reports. Most bugs share a common lifecycle as described by Anvik et al. [1]. A developer finds a problem of the software and then creates a bug report in a bug repository. Next, the bug report will be assigned to a developer who is responsible for resolving this bug by a triager. Other developers can provide potential solutions for resolving this bug by commenting the bug report. Finally, after the fixer resolves the bug, other developers verify his/her solution for the bug. If correctly fixed, the bug will be closed and marked with "resolved" or the bug will be reopened for future resolution. Following prior work Jeong et al. [14], we refer developers to all the people worked in bug repositories, including reporters, triagers, fixers, and active users.

B. Heterogeneous Network

A heterogeneous network [4] contains multiple types of objects and multiple types of links, and the definition is presented as follows.

Heterogeneous Network: A heterogeneous network is defined as a directed graph $G = (V, E)$. *V* is the set of nodes, including *n* types of objects. $E \subseteq V \times V$ is the set of links between nodes in *V*, which involves multiple types of links.

Objects in a heterogeneous network can be connected via different paths with different relations. Formally, these paths are called *meta path* [4], and can be defined as below.

Meta Path: A meta path *P* is a path defined on heterogeneous network $G = (V, E)$, and is represented in form of $O_1 \xrightarrow{R_1} O_2 \xrightarrow{R_2} \ldots \xrightarrow{R_{n-1}} O_n$(*n* this the number of objects in *P*). Here, O_i $(1 \le i \le n)$ is an object in *V* and R_i is a relation between two objects in *V*. Meta path *P* can be abbreviated as $O_1 - O_2 - \ldots - O_n$, if there is no ambiguity in either the meaning or the order of the relations.

In heterogeneous network, *path count* [4] is frequently used to measure topological features of meta path.

Path count: the number of path instances *p* between two objects *x* and *y* following meta path *P* and denoted as $PC_P(x, y)$.

III. EXTRACTING HETEROGENEOUS NETWORK

In this paper we use the bug repositories of open source projects Eclipse and Mozilla as examples to explore the heterogeneous network. In a typical bug repository, a bug is reported and submitted to the bug repository as a bug report, which contains full information of the bug. Developers can suggest potential solution for fixing bugs via comments [16], and each bug belongs to a component.

To construct heterogeneous networks from bug repositories, we empirically consider 4 types of objects, namely **developers** (denoted as D), **bugs** (denoted as B), **comments** (denoted as S), and **components** (denoted as C), and their interactions during the process of fixing bugs in bug repositories. Relations between developers and comments are represented by "write" and "written by" (denoted as write^{-1}). Relations between bugs and comments are represented by "comment" and "commented by" (denoted as comment^{-1}). Relations between components and bugs are represented by "contain" and "belong to". Relations between bugs are represented by "duplicate" and "duplicate of" (denoted as duplicate^{-1}). Interactions between developers and bug reports have multiple types of relations. That is, a developer can report a new bug, assign a bug to another developer, toss a bug to other developer, fix a bug, close a bug, and reopen a bug. Here we use "report/assign/toss/fix/close/ reopen" to represent the multiple relations from developers to bugs and use "report^{-1}/assign^{-1}/toss^{-1}/fix^{-1}/close^{-1}/reopen^{-1}" to represent the

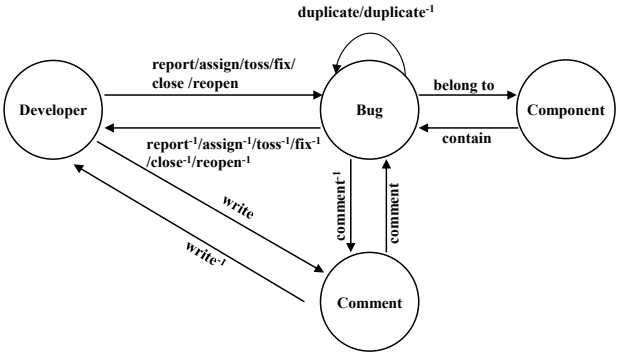


Figure 2. Schema for heterogeneous network in a bug repository.

multiple relations from bugs to developers, which means a bug "reported by", "assigned by", "tossed by", "fixed by", "closed by", or "reopened by" a developer.

Further, we employ the concept of **network schema** [4] shown in Fig. 2 to summarize the meta structure of the constructed heterogeneous network defined for a bug repository. In the network schema, nodes denote the types of objects, and links (also called meta path) denote the relations between the two objects.

With the network schema, we can extract a heterogeneous network from a bug repository. Typical bug repositories maintain a bug report and an activity log for each bug. A bug report records full information of a bug such as the reporter, fixer, commenters and their comments, and the component this bug belongs to. An activity log records the change histories of a bug report, e.g., the change histories of the bug status, the fixer, and tossing histories of this bug report [14].

In bug repositories, bug reports and their activity logs are typically presented as HTML web pages. By crawling and parsing these HTML web pages, we can obtain meta paths directly between two objects shown in network schema to build a heterogeneous network. Algorithm 1 shows how to construct a heterogeneous network by extracting different types of meta paths from bug reports and their activity logs in a bug repository with network schema in Fig. 2.

Algorithm 1. Extracting a heterogeneous network from a bug repository

Input: B: bug report set, L: activity log set, T: a set of directly link between objects (shown in Fig. 2)

Output: Heterogeneous network HN

```
1. HeterogeneousNetworkExtracting(B, L, T);
2. Procedure HeterogeneousNetworkExtracting (B, L, T)
3.     for link t_i in T do
4.         for bug report b_i in B do
5.             if b_i contains objects in t_i then
6.                 add instance of t_i to HN;
7.             end if;
8.         end for;
9.         for activity log l_i in L do
10.            if l_i contains objects in t_i then
11.                add instance of t_i to HN;
12.            end if;
13.        end for;
14.    end for;
```

TABLE I. THE DETAILS OF DATASET

Porjects	#Developers	#Components	#Bug reports	#Comments
Eclipse	32,722	1005	300,191	1,569,403
Mozilla	150,326	709	488,053	4,093,368

IV. MULTIPLE TYPES OF DEVELOPER CONTRIBUTION IN HETEROGENEOUS NETWORKS OF BUG REPOSITORIES

In this section, we investigate multiple types of developer contribution and its applications in heterogeneous networks extracted from bug repositories of two large scale and open source projects namely Eclipse and Mozilla.

A. Research Questions

We propose four Research Questions (RQs) to explore developer contribution in bug repositories. The first three questions are analysis of multiple types of developer contribution. The fourth question is application of multiple types of developer contribution in prediction problems. We answer these two categories of RQs in section IV (RQ1, RQ2, and RQ3) and section V (RQ4), respectively.

RQ1. How to represent multiple types of developer contribution in a heterogeneous network extracted from a bug repository?

RQ2. What is the proportion of each type of contribution?

RQ3. What is the distribution and evolution of multiple types of developer contribution in bug repositories?

For RQ1, RQ2, and RQ3, we use four types of meta paths to explore developer contribution in heterogeneous networks extracted from bug repositories.

RQ4. Can we use multiple types of developer contribution to assist prediction problems in bug repositories?

In RQ4, we investigate how to leverage meta path-based developer contribution to improve a typical problem in bug repositories, i.e., bug triage.

B. Data Collection

To investigate the answers of the above four RQs, we conducted experiments on bug repositories of Eclipse and Mozilla. For Eclipse we collected bug reports from 2001/10/10 to 2010/01/31 including 300,191 bug reports; for Mozilla we collected bug reports from 1998/04/07 to 2009/06/31 including 488,053 bug reports. The details of our dataset are shown in Table I. With Algorithm 1, we extract heterogeneous networks from Eclipse and Mozilla projects based on the above dataset, respectively.

C. Multiple Types of Developer Contribution in Heterogeneous Network

In software bug repositories, developers can contribute to a component via multiple ways including reporting new bugs, reopening incorrectly fixed bugs, fixing unsolved bugs, and commenting on bug reports. Prior studies [8], [13] consider developer contribution based on purely developers' comments. The comment-based approaches ignore other

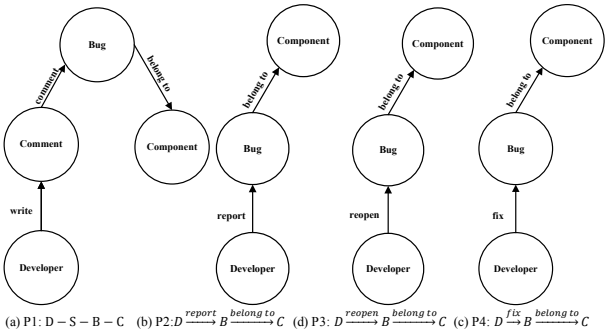
kinds of developer contribution. For example, according to our observation, developer "use leaf" has fixed 128 bugs during 1998/09/14-2003/03/19 in Mozilla project, however, he made only 1 comment on bug reports; developer "Rick Osborne" reported 22 bugs in the year 1998 without any comment. Intuitively, many developers like "use leaf" and "Rick Osborne" may focus on fixing or reporting bugs. Thus, they seldom make comments on bug reports. So, simply applying comment-based approaches to measure developer contribution in bug repositories might be inappropriate.

This shortcoming motivates us to propose a new developer contribution measure that captures multiple types of developer contribution from different behaviors of developers in bug repositories. In this work, we mainly consider component-level developer contribution (contribution from developers to a software component) on fixing bugs via four types of developer behaviors. These four types of developer contribution are listed as below:

- ***Developer contribution via making comments***. Fixing bug is a team work, many developers give their suggestions or supplying related information by making comments. We use meta path $\mathrm{D}-\mathrm{S}-\mathrm{B}-\mathrm{C}$ to address this kind of developer contribution. This meta path means a developer (D) makes a comment (S) on a bug report (B) of a component (C).
- ***Developer contribution via reporting bugs***. Developers find bugs and report these bugs (here we only consider non-duplicate bugs), we address this kind of contribution by meta path $\mathrm{D} \xrightarrow{\text{report}} \mathrm{B} \xrightarrow{\text{belong to}} \mathrm{C}$, which denotes a developer (D) reports a new bug report (B) of a component (C).
- ***Developer contribution via reopening bugs***. Some bug reports labeled with "FIXED" may be incorrectly fixed, developers may find these bugs and reopen them for correctly fixing in the future. We address this kind of contribution by meta path $\mathrm{D} \xrightarrow{\text{reopen}} \mathrm{B} \xrightarrow{\text{belong to}} \mathrm{C}$, which means a developer (D) reopens an incorrectly fixed bug report (B) of a component (C).
- ***Developer contribution via fixing bugs***. A vital reason for using a bug repository is improving software quality by fixing bugs. Thus, fixing bug is a direct way to contribute to a bug repository. In this work, we study this kind of developer contribution via meta path $\mathrm{D} \xrightarrow{\text{fix}} \mathrm{B} \xrightarrow{\text{belong to}} \mathrm{C}$, which means a developer (D) fixes a bug (B) of a component (C).

Fig. 3 shows the four types of meta paths used to address multiple types of developer contribution in this work. Meta path instances of these meta paths can be produced by traversing on the network schema using breadth-first search.

We propose a metric to measure the contribution from a developer to a component in a bug repository based on four types of meta paths.



(a) P1: $\mathrm{D}-\mathrm{S}-\mathrm{B}-\mathrm{C}$ (b) P2: $D \xrightarrow{report} B \xrightarrow{belong\ to} C$ (d) P3: $D \xrightarrow{reopen} B \xrightarrow{belong\ to} C$ (c) P4: $D \xrightarrow{fix} B \xrightarrow{belong\ to} C$

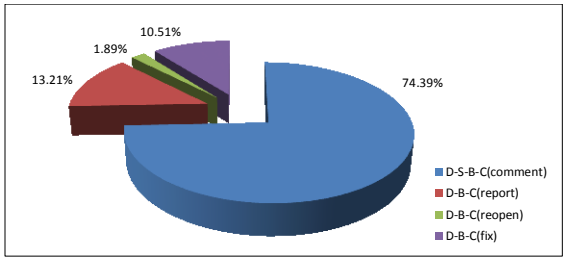
Figure 3. Meta paths used to represent developer contribution.

Given a developer d_i ($1 \le i \le n$, n is the number of developers) and a component c, we use *path count* of these four meta paths namely $PC_{P1}(d_i, c)$, $PC_{P2}(d_i, c)$, $PC_{P3}(d_i, c)$, and $PC_{P4}(d_i, c)$ to represent the contribution that d_i made to c via meta paths *P1*, *P2*, *P3*, and *P4*, respectively. We denote $Con(d_i, c)$ as contribution that d_i made on c, and it can be calculated by the following equation.

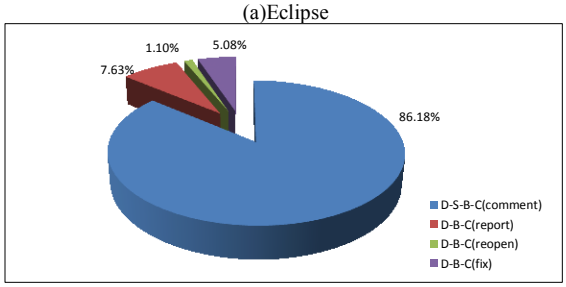
$$Con(d_i,c) = \frac{PC_{P1}(d_i,c)}{\mathrm{M1}} + \frac{PC_{P2}(d_i,c)}{\mathrm{M2}} + \frac{PC_{P3}(d_i,c)}{\mathrm{M3}} + \frac{PC_{P4}(d_i,c)}{\mathrm{M4}} \quad (1)$$

In equation 1, M1, M2, M3, and M4 are normalized parameters, by which we normalize each kind of contribution to the range from 0 to 1. We regulate $\mathrm{M1} = max_{1 \le i \le n} PC_{P1}(d_i, c)$, $\mathrm{M2} = max_{1 \le i \le n} PC_{P2}(d_i, c)$, $\mathrm{M3} = max_{1 \le i \le n} PC_{P3}(d_i, c)$, and $\mathrm{M4} = max_{1 \le i \le n} PC_{P4}(d_i, c)$. Thus the range of $Con(d_i, c)$ is [0 4]. Moreover, we can calculate the whole developer contribution from all developers to component c by $Con(c) = \sum_{1 \le i \le n} Con(d_i, c)$.

Answer to RQ1. *Employing meta path-based analysis, we can represent and measure four types of developer contribution in bug repositories.*



(a)Eclipse



(b)Mozilla

Figure 4. The proportion of four types of meta paths for measuring developer contribution in bug repositories of Eclipse and Mozilla.

Proportion of four types of developer contribution. Understanding the proportion of each types of contribution could provide valuable insights into the process of software development and maintenance. We investigate the proportion of four types of developer contribution in the whole projects.

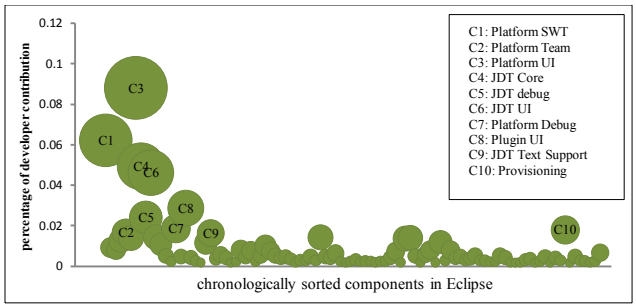
Fig. 4 presents the ratio of four types of meta paths (listed in Fig. 3) extracted from bug repositories of Eclipse and Mozilla. In both projects, the ratios of meta paths $D \xrightarrow{\text{reopen}} B \xrightarrow{\text{belong to}} C$ and $D \xrightarrow{\text{fix}} B \xrightarrow{\text{belong to}} C$ are lower than the other two types of meta paths, and the ratio of meta path $D - S - B - C$ is significant in both projects (over 70%). This implies that most developers contribute to these two projects by commenting bugs. We also note that in Eclipse the ratio of comment-based developer contribution is about 74%, in Mozilla this ratio is up to 86.18%, one possible reason for this is the number of developers in Mozilla is larger than that in Eclipse (150,326 developers in Mozilla, and 32,722 developers in Eclipse) and typically a bug report is assigned to a single developer. Thus, many developers in Mozilla do not fix bugs and they may suggest potential solutions for fixing bugs via commenting bugs.

Answer to RQ2. *In both projects, ratios of comment-based developer contribution are over 74%. Other kinds of developer contribution are also significant, e.g., over 13% developer contribution in Eclipse is made by reporting bugs. This fact supports our argument of the need of capturing multiple developer contribution ignored in prior work.*

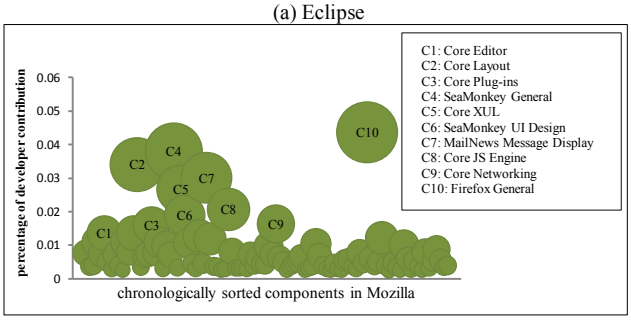
Distribution of Developer Contribution. In Fig. 5, we present a full view of the distribution of component-level developer contribution in the whole projects. Two indicators of a component are used, one is the sum of *path count* of four types of meta paths in this component, another is the percentage of developer contribution on this component. We label the top 10 components which occupy highest developer contribution with their names.

Fig. 5 lists the detailed distribution of developer contribution among components of Eclipse and Mozilla. Obviously, a component with a large sum of *path count* leads to occupying high percentage of developer contribution. We note that the distribution of developer contribution in Eclipse is not as even as that in Mozilla. In Eclipse, most of the components that occupy high developer contribution emerge in the very beginning of the project, e.g., 9 of the top 10 components which occupy high developer contribution emerge before the year 2001. Moreover, the top 10 components occupy 36.99% developer contribution. However, in Mozilla, these important components emerge along with the development of the project gradually. And these components occupy 25.80% developer contribution. One possible reason for this result is Eclipse is a corporation leaded open source project that supported by IBM [17], and is well designed at the beginning. However, Mozilla is driven by open source community [18], so its architecture may not as mature as that of Eclipse at the beginning.

To observe the distribution of contribution of dominant developers (developers with significant contribution), we

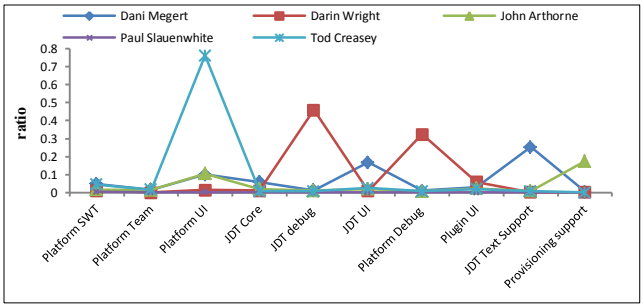


(a) Eclipse

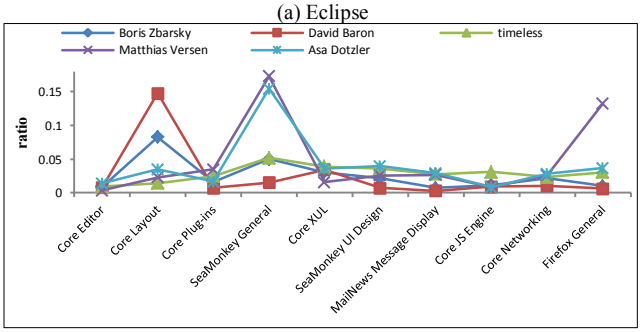


(b)Mozilla

Figure 5. The percentage of developer contribuion on components of Eclipse and Mozilla. The diameter of a circle denotes the sum of path count of four types of meta paths.



(a) Eclipse



(b) Mozilla

Figure 6. Contribution on 10 components for 5 dominant developers in the whole project. For each projcet, the selected 10 components are labled in Fig. 5, which occupying higest developer contribution. In vertical axis, the ratio denotes the percentage of developer contribution on components.

present the contribution of top 5 developers (who make the highest contribution in the whole project) on 10 active components (i.e., top 10 components in Fig. 5) in Fig. 6. For most developers, they mainly contribute to one or two components, while contribute a little to other components. e.g., in Eclipse, for developer "Tod Creasey", 76.24%

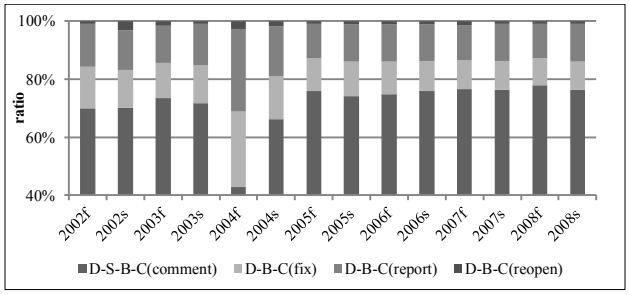
contribution is on component **Platform UI**, while only 0.4%contribution is on component **JDT Debug**; in Mozilla, developer "Matthias Versen" mainly contributes to components **SeaMonkey General** and **Firefox General**.

The reason for this fact might be that developers have different expertise and responsibilities in the development and maintenance of software projects.

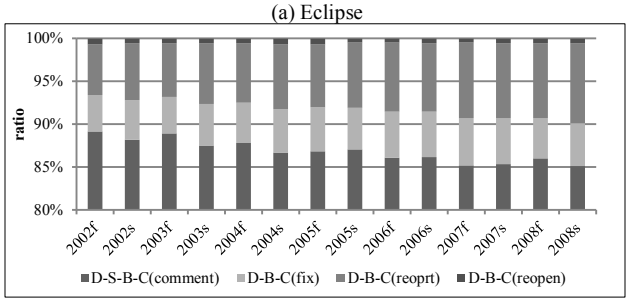
Evolution of Developer Contribution. Both Eclipse and Mozilla have been developed for many years. To investigate the evolution of four types of developer contribution, we select bug reports in 7 continuous years, from Jan. 2002 to Dec. 2008 in Eclipse and Mozilla. We choose half a year as an interval. We denote the first half year with "f" and the second half year with "s". For each interval, we calculate the ratios of four types of developer contribution.

As shown in Fig. 7, the ratios in Mozilla are more stable than these in Eclipse. In Mozilla, the ratio of meta path $D-S-B-C$, which denotes comment-based developer contribution, are over 85% in each interval. In Eclipse the ratio of each types of developer contribution fluctuates drastic around the year 2004. Moreover, in both projects, the ratios of each types of developer contribution are kept stable after 2004.

In both Eclipse and Mozilla projects, the ratios of four types of developer contribution change over time. This result might be caused by two possible reasons. One is the increasing complexity of the two projects; another is the changing developers of these two projects.



(a) Eclipse



(b)Mozilla

Figure 7. Evolution of normalized percentage for each type of developer contribution in Eclipse and Mozilla over time.

Answer to RQ3. *Several dominant components occupy high developer contribution. Most developers mainly contribute to one or two components. The ratios of four types of developer contribution are changing along with time.*

V. ASSIST BUG TRIAGE IN BUG REPOSITORIES

In this section, we investigate the results of leveraging meta path-based multiple types of developer contribution to assist a typical prediction problem in bug repositories. Driven by **RQ4**, we exam our approach on bug triage which has been widely studied in existing researches.

Bug triage is a widely known problem during software development and maintenance, which aims to predict a potential developer for a new coming bug [1]. Many machine learning based automatic bug assignment algorithms have been proposed to reduce time and labor cost of manual ways. Most of prior work models bug triage as a text classification problem based on knowledge extracted from bug reports [1], [13], [14], [15], [19]. In this paper, we consider improving bug triage by combining machine learning methods with multiple types of developer contribution revealed in heterogeneous networks extracted from bug repositories.

Following Bhattacharya et al. [15], we employ the incremental learning framework to evaluate the accuracy of bug triage. We sort bug reports in chronological order and divide them into 11 folds and execute 10 rounds to investigate accuracies of all the folds. In each round, we calculate four types of developer contribution in the heterogeneous network extracted from the training set, and then combine it with the prediction set of a classifier to generate a new prediction set.

Formally, we define our approach as follows.

1. Given a new coming bug report *B*, we predict a set of potential developers called *PD* by a machine learning classifier, e.g., Supporting Vector Machine (SVM) and Naïve Bayes (NB).

2. We extract the component c of the new coming bug report, and calculate four types of developer contribution in component c of each developer by equation 1.

3. We combine developer contribution with the prediction results of a machine learning method. For each developer d_i, the final score $F_i = p_i + con(d_i, c)$ where p_i is the predicted probability for d_i by a classifier.

4. We rank developers in *PD* by their final scores and select developers with highest scores as the final prediction set. In our work we examine the results of top-5 developers in the final prediction set.

5. We evaluate the accuracies of top 1, top 3, and top 5 developers in the final prediction set. The accuracy is computed by $\text{Accuracy} = \frac{\#\text{correctly predicted bug reports}}{\#\text{ all the bug repots}}$, where the number of correctly predicted bug reports is computed by matching developers in the final prediction sets and developers who fixed the bugs.

TABLE II. PREDICATION ACCURACY OF BUG TRIAGE ON ECLIPSE AND MOZILLA

Project	Classifier	Selection	Approach	Accuracy for each fold (%)										Average Accuracy	Improvement
				2	3	4	5	6	7	8	9	10	11		
Eclipse	SVM	Top1	**SVM**	16.14	16.50	18.67	22.22	24.36	19.72	21.80	25.78	26.60	26.51	21.83	22.15
			SVM+MD	38.04	41.90	39.45	43.62	44.49	48.53	44.07	45.02	47.47	47.16	43.98	
		Top3	**SVM**	25.52	29.68	33.90	37.97	38.63	35.10	41.98	43.38	44.29	44.60	37.51	36.87
			SVM+ MD	68.40	74.05	70.85	75.71	75.84	78.73	72.62	74.33	76.35	76.87	74.38	
		Top5	**SVM**	30.93	37.26	41.16	45.88	47.47	43.72	52.22	52.42	52.93	54.44	45.84	40.17
			SVM+ MD	80.95	87.06	82.46	87.03	87.37	**89.39**	85.06	86.03	86.68	88.11	86.01	
	Naïve Bayes	Top1	**NB**	26.12	28.64	29.13	31.95	34.77	29.57	31.97	30.71	32.86	33.44	30.92	11.37
			NB+ MD	37.02	39.41	38.94	41.46	45.91	44.76	43.41	41.79	44.52	45.67	42.29	
		Top3	**NB**	34.98	36.32	37.19	40.71	43.03	37.21	39.94	41.27	43.29	44.40	39.83	32.15
			NB+ MD	65.47	71.14	68.92	72.26	74.93	75.67	71.20	71.78	73.93	74.53	71.98	
		Top5	**NB**	36.53	37.01	38.32	41.95	44.23	38.30	40.72	42.98	44.85	46.22	41.11	43.82
			NB+ MD	79.35	84.95	80.82	85.90	87.14	88.92	84.04	85.24	85.68	87.22	84.93	
Mozilla	SVM	Top1	**SVM**	12.89	16.52	15.59	14.19	17.55	18.06	18.64	20.29	23.57	22.35	17.97	11.62
			SVM+ MD	32.11	31.43	29.27	28.51	30.47	30.88	31.39	27.69	26.55	27.56	29.59	
		Top3	**SVM**	22.79	29.71	28.72	28.34	33.65	35.61	38.14	38.07	41.43	39.01	33.55	21.59
			SVM+ MD	55.69	55.24	53.70	52.84	55.04	53.70	56.55	56.44	56.03	56.15	55.14	
		Top5	**SVM**	29.78	37.77	37.59	37.83	43.35	46.68	48.63	46.57	50.71	48.10	42.70	23.99
			SVM+ MD	66.45	67.48	67.58	64.53	65.52	66.14	66.93	69.71	66.21	66.30	66.69	
	Naïve Bayes	Top1	**NB**	25.43	27.03	25.50	24.95	26.35	26.94	30.16	30.23	30.57	28.35	27.55	5.75
			NB+ MD	34.41	33.65	31.80	31.91	31.70	33.10	35.98	34.13	34.17	32.19	33.30	
		Top3	**NB**	35.98	38.59	36.77	35.23	38.21	38.90	41.74	41.30	40.88	39.49	38.71	17.95
			NB+ MD	57.23	57.88	56.37	54.52	56.20	55.38	58.40	56.51	57.95	56.19	56.66	
		Top5	**NB**	38.86	41.19	38.93	37.25	40.61	41.60	43.76	43.15	43.04	41.06	40.95	28.00
			NB+ MD	68.61	69.12	68.47	66.28	67.68	68.68	71.21	**71.45**	70.25	67.74	68.95	

We validate our approach on bug reports of Eclipse and Mozilla. We choose bug reports after the year 2005 since in both projects the four types of developer contribution are stable. For Eclipse, we select bug reports from 200001 to 300000(Aug 2007 to Jan 2010). For Mozilla, we consider bug reports from 400001 to 500000(Oct 2007 to Jun 2009). We remove the non-fixed bug reports (the resolution of bug reports not marked as "FIXED") and inactive developers (developers who have fixed less than 50 bug reports) as prior work [1], [13]. After this, 49,539 bug reports from Eclipse and 32,097 bug reports from Mozilla are selected. We use the techniques of tf-idf [25], removing stop words, and stemming to extract string vectors from the title and description of a bug report. For each bug report, its finally fixer is extracted as a label for the classifier. We use Weka's [25] built-in Supporting Vector Machine (SVM) and Naïve Bayes (NB) classifiers in this work.

Table II shows the results of bug triage in Eclipse and Mozilla by combining the multiple types of developer contribution with the output of the classifiers (MD denotes the multiple types of developer contribution. SVM+MD and NB+MD denote the results of our approach based on SVM and NB classifiers, respectively).

It obviously that, for both SVM and NB, the accuracies are improved when combed with developer contribution. The maximum prediction accuracy for Eclipse is 89.39% and 71.45% for Mozilla. For both classifiers, when combined with developer contribution, the average improvement for Eclipse is better than that of Mozilla. One possible reason for this fact is the contribution of most developers in Mozilla are more uniform than that in Eclipse since the number of developers in Mozilla is larger than that in Eclipse (150,326 developers in Mozilla, and 32,722 developers in Eclipse). Thus, for Mozilla, the ranks for developers may not change too much when combining with developer contribution.

As shown in Table IV, by mixing the original vector with multiple types of developer contribution, the performance is improved.

Answer to RQ4. *By examining a typical prediction problem in bug repositories, we conclude that meta path-based multiple developer contribution can provide enriched knowledge of developer contribution to improve results of bug triage in bug repositories.*

VI. THREATS TO VALIDITY

In this work, we investigate multiple types of developer contribution based on heterogeneous networks extracted from bug repositories of two large open source software projects namely Eclipse and Mozilla. However, it is possible that our approach may not work well on some closed-source software (e.g., commercial software) or small scale open source software projects. Where, developer contribution pattern in those projects may be different with Eclipse and Mozilla. Whether our heterogeneous network-based approach is feasible for these software projects should be further investigated.

VII. RELATED WORK

A. Developer Contribution in Software Developerment

Gousios et al. [3] propose an approach for evaluating developer contribution during the process of software development by a set of predefined developer actions. Their work also considers positive and negative developer contribution. Pinzger et al. [11] study the correlation between developer contribution and the number of post-release failures by building a developer-module network. Xu et al. [20] study the development community at SourceForge.net and empirically classifier developers into project leader, core developer, co-developer, and active user based on developer contribution in software projects. Hong et al. [8] analyze the

difference between developer social network and general social network (e.g., Facebook) based on developers' comments in bug repository of Mozilla. Xuan et al. [13] investigate developer prioritization and its applications in bug repositories by building comment-based developer networks. J. Eyolfson et al. [24] investigate the relation between developers' contribution and the quality of their commits.

B. Bug triage

Čubranić et al. [2] first model bug triage as a text classification problem to semi-automate bug assignment. Anvik et al. [1] improve the above work with filtering out unfixed bug reports and inactive developers. Jeong et al. [14] and Bhattacharya et al. [15] improve bug triage using tossing graph based on developers tossing behaviors in bug repositories. Our prior work [21] proposes an approach to improve bug triage based on developer collaboration in heterogeneous bug repositories. Other work also refers to this topic, such as developer prioritization based approach [13], the fuzzy-set and cache-based approach [19].

VIII. CONCLUSION

In this paper, we explore multiple types of developer contribution revealed by meta path-based analysis in a heterogeneous network extracted from a bug repository. We consider leveraging our approach to assist a typical prediction problem in bug repositories, i.e., bug triage. Our study strongly suggests using the heterogeneous network-based analysis can open many actionable insights in analyzing software repositories.

ACKNOWLEDGMENT

This research was supported in part by National Natural Science Foundation of China under Grant Nos. 61073044, 71101138, 61003028 and 61379046; National Science and Technology Major Project under Grant Nos. 2012ZX01039-004; Beijing Natural Science Fund under Grant No.4122087; State Key Laboratory of Software Engineering of Wuhan University.

REFERENCES

[1] J. Anvik, L. Hiew, and G.C. Murphy, "Who Should Fix This Bug?," Proc. 28th Intl. Conf. Software Engineering (ICSE '06), May 2006,pp. 361-370.

[2] D. Čubranić and G.C. Murphy, "Automatic Bug Triage Using Text Categorization," Proc. 16th Intl. Conf. Software Engineering & Knowledge Engineering (SEKE '04), Jun. 2004, pp. 92-97.

[3] G. Gousios, E. Kalliamvakou, and D. Spinellis, "Measuring Developer Contribution from Software Repository Data," Proc. 5th IEEE Working Conf. Mining Software Repositories (MSR '08), May 2008, pp. 129-132.

[4] Y. Sun and J. Han: Mining Heterogeneous Information Networks: Principles and Methodologies. Synthesis Lectures on Data Mining and Knowledge Discovery, Morgan & Claypool Publishers 2012.

[5] K. Crowston and J. Howison, "The social structure of free and open source software development," First Monday, vol. 10, no. 2, 2005.

[6] S.H. Kan, Metrics and Models in Software Quality Engineering. Addison Wesley, 1995.

[7] C. Bird, D. Pattison, R. D'Souza, V. Filkov and P. Devanbu, "Latent Social Structure in Open Source Projects," Proc. 16th ACM SIGSOFT Intl. Symp. Foundations of software engineering (FSE '08), Nov. 2008, pp.24-35.

[8] Q. Hong, S. Kim, S.C. Cheung, and C. Bird, "Understanding a Developer Social Network and its Evolution," Proc. 27th IEEE Intl. Conf. Software Maintenance (ICSM '11), Sept. 2011, pp. 323-332.

[9] L. C. Freeman. Centrality in social networks: Conceptual clarification. Social Networks, 1(3), pp. 215-239, 1979.

[10] J. Asundi, "The need for effort estimation models for open source software projects", 5-WOSSE: Proc. 5th workshop on Open source software engineering, May, 2005, pp.1-3.

[11] M. Pinzger, N. Nagappan, and B. Murphy, "Can Developer-Module Networks Predict Failures?," Proc. 16th ACM SIGSOFT Intl. Symp. Foundations of Software Engineering (FSE '08), Nov. 2008, pp. 2-12.

[12] A. Meneely, L. Williams, W. Snipes, and J. Osborne, "Predicting Failures with Deverloper Networks and Social Network Analysis," Proc. 16th ACM SIGSOFT Intl. Symp. Foundations of Software Engineering (FSE '08), Nov. 2008, pp. 13-23.

[13] J. Xuan, H. Jiang, Z. Ren, and W. Zou, "Developer Prioritization in Bug Repositories," Proc. 34st Intl. Conf. Software Engineering (ICSE '12), June.2012, pp. 25-35

[14] G. Jeong, S. Kim, and T. Zimmermann, "Improving Bug Triage with Tossing Graphs," Proc. 17th ACM SIGSOFT Symp. Foundations of Software Engineering (FSE'09), Aug. 2009, pp. 111-120.

[15] P. Bhattacharya and I. Neamtiu, "Fine-Grained Incremental Learning and Multi-Feature Tossing Graphs to Improve Bug Triaging," Proc. 26th IEEE Intl. Conf. Software Maintenance (ICSM '10), Sept. 2010, pp. 1-10.

[16] T. Zimmermann, R. Premraj, N. Bettenburg, S. Just, A. Schröter, and C. Weiss, "What Makes a Good Bug Report?," IEEE Trans. Software Engineering, vol. 36, no.5, Oct. 2010, pp. 618-643.

[17] T. Mens, J. Fernandez-Ramil, and S. Degrandsart, "The Evolution of Eclipse" Proc. 24th IEEE Intl. Conf. Software Maintenance (ICSM '08), Sept. 2008, pp. 386 - 395.

[18] A. Mockus, R. T. Fielding, and J. D. Herbsleb, "Two case studies of open source software development: Apache and Mozilla" ACM Trans. Software Engineering and Methodology, vol. 11, issue 3, July 2002, pp 309 – 346.

[19] A. Tamrawi, T.T. Nguyen, J.M. Al-Kofahi, and T.N. Nguyen, "Fuzzy Set and Cache-Based Approach for Bug Triaging," Proc. 19th ACM SIGSOFT Symp. Foundations of Software Engineering (FSE '11), Sept. 2011, pp. 365-375.

[20] 20J. Xu, Y. Gao, S. Christley, and G. Madey, "A topological analysis of the open souce software development community," Proc. 38th IEEE Hawaii International Conference on System Sciences(HICSS '05), Jan. 2005, pp. 198a

[21] S. Wang, W. Zhang, Y. Yang, and Q. Wang, "DevNet: Exploring Developer Collaboration in Heterogeneous Network of Bug Repositories," Proc. 7th ACM / IEEE Intl. Symp. Empirical Software Engineering and Measurement (ESEM'13), Oct. 2013.

[22] Y. Sun, B. Norick, J. Han, X. Yan, P. Yu, and X. Yu, "Integrating meta-path selection with user-guided object clustering in heterogeneous information networks," Proc. 18th ACM SIGKDD Intl. Conf. Knowledge Discovery and Data Mining (KDD'2012), Aug.2012, pp. 1348-1356.

[23] Y. Yang, N. Chawla, Y. Sun, and J. Hani, "Predicting Links in Multi-relational and Heterogeneous Networks," Proc. 12th Int. Conf. Data Mining (ICDM' 12), Dec.2012, pp. 755-764.

[24] J. Eyolfson, L. Tan, and P. Lam, "Do time of day and developer experience affect commit bugginess?," Proc. 8th IEEE Working Conf. Mining Software Repositories (MSR '11), May 2011, pp. 153-162.

[25] I.H. Witten, E. Frank, and M.A. Hall, Data Mining: Practical Machine Learning Tools and Techniques, 3rd ed. Morgan Kaufmann, Burlington, MA, 2011.