10-2010

Mining collaboration patterns from a large developer network

Didi SURIAN
Singapore Management University, didisurian@smu.edu.sg

David LO
Singapore Management University, davidlo@smu.edu.sg

Ee Peng LIM
Singapore Management University, eplim@smu.edu.sg

Citation

SURIAN, Didi; LO, David; and LIM, Ee Peng. Mining collaboration patterns from a large developer network. (2010). *WCRE 2010: 17th Working Conference on Reverse Engineering: 13-16 October 2010, Beverly, Massachusetts: Proceedings*. 269-273.
Available at: https://ink.library.smu.edu.sg/sis_research/1339

Mining Collaboration Patterns from a Large Developer Network

Didi Surian, David Lo and Ee-Peng Lim
School of Information Systems, Singapore Management University
didisurian,davidlo,eplim@smu.edu.sg

***Abstract*—In this study, we extract patterns from a large developer collaborations network extracted from SourceForge.Net at high and low level of details. At the high level of details, we extract various network-level statistics from the network. At the low level of details, we extract topological sub-graph patterns that are frequently seen among collaborating developers. Extracting sub-graph patterns from large graphs is a hard NP-complete problem. To address this challenge, we employ a novel combination of graph mining and graph matching by leveraging network-level properties of a developer network. With the approach, we successfully analyze a snapshot of SourceForge.Net data taken on September 2009. We present mined patterns and describe interesting observations.**

I. Introduction

With the advent of communication devices, distant locations do not hamper people to collaborate. Many projects involve participants from diverse locations. Some of which might be at the opposite ends of the globe. This is certainly the case with software development.

Both open-source and closed-source software development efforts benefit from this trend. Globally distributed software development is apparent in large corporations like Microsoft, IBM, etc that have centers all around the globe working together on various common projects. In the open-source settings, various people having common interests despite their location work together in various projects.

Recently, there has been a number of work that investigates globally distributed software development [9], [12], [4]. Other studies visualize the socio-technical relationships in software development [10], [6].

Extending the above studies, in this study we investigate properties of distributed developments and patterns of collaboration among large number of developers. We particularly focus on open source development community in SourceForge.Net.

By mining or reverse engineering patterns from historical collaboration data we could learn from the past and capture some interesting observations that could help in aiding open source development in the future. Collaboration patterns shed light to the dynamics of collaborations among developers and highlight some apparent properties. Some weaknesses of the current level or degree of collaborations among software developers could also be identified.

As subjects of our analysis, we take developers involved in software projects hosted within SourceForge.Net which is the most popular open-source software portal. These developer collaboration networks are modeled as a large graph or network whose nodes correspond to developers and edges correspond to collaborations among them. Based on this huge collaboration network, we investigate several research questions listed below:

RQ 1: How connected are the developers? Are all developers connected to every other developers in the-network? If no, how many clusters of connected developers are there in the network?

RQ 2: What are some characteristics of developer collaboration clusters?

RQ 3: What are some common topological collaboration patterns appearing in these developer collaboration clusters?

RQ 4: Within a connected collaboration cluster, are all developers connected to every other developers in 6 hops following the small world phenomenon?

To answer the above questions, we compute various counts or statistics from the collaboration network. In addition to the statistics, we also extract detailed topological patterns that represent the most frequent patterns of collaborations appearing in the network. Frequent topological patterns of collaborations shed light on factors affecting the formation of collaborations among developers.

Our contributions are as follows:

1) We investigate and describe various network-level statistics extracted from large scale software collaborations in SourceForge.Net. We report some interesting observations and answer some hypotheses.
2) We propose a novel solution that merges graph mining and graph matching that scales to mine frequent sub-graphs from a large software collaboration network.
3) We present top-30 topological collaboration patterns mined from SourceForge.Net. We also describe some interesting observations based on these patterns.

The structure of this paper is as follows. In Section II some definitions and concepts are described. In Section III, our approach to extract the dataset and mine patterns are elaborated. In Section IV, we describe our experiment and discuss our findings. We discuss some issues in Section V. Section VI describes related work. Finally, we conclude and mention future work in Section VII.

II. Definitions and Concepts

In this section, we describe some graph notations and introduce sub-graph isomorphism, pattern matching, and frequent graph mining problems.

A. Basic Notations

A collaboration graph G is a non-directed graph representing collaborations among a set of developers. The vertex/node set denoted by N includes all the developers appearing in the input dataset under study. The set of edges in G is denoted by E and corresponds to a set of vertex pairs. Each pair (v_i, v_j) corresponds to an edge between nodes v_i and v_j in G. Such an edge means that developer v_i collaborates with developer v_j before. For each node there is a label associated to it. This corresponds to the identifier of the developer. The label of v_i is denoted as $l(v_i)$ and the set of all labels in G is denoted as N_L. Hence, a graph could be represented as the triple (N,E,N_L).

A collaboration pattern is a set of nodes and a set of edges. It could be represented as a pair (N,E), where N is a set of nodes, and E is a set of edges.

B. Sub-Graph Isomorphism

Sub-graph matching or sub-graph isomorphism is a well known problem. The formal definition of subgraph isomorphism is given in Definition 1.

Definition 1 (Sub-Graph Isomorphism): Consider two graphs G=(N, E, N_L) and G'=(N', E', N'_L). *Sub-graph isomorphism* is an injective function $f : N \rightarrow N'$, s.t., (1), $\forall n \in N, l(n) = l'(f(n))$; and (2), $\forall (u, v) \in E, (f(u), f(v)) \in E'$. The function or mapping f is referred to as the embedding of G in G'.

A pattern P is said to match a graph G, iff ignoring the node labels in G, P is a sub-graph of G. Mathematically, this is defined in Definition 2.

Definition 2 (Pattern Match): A pattern P=(N, E) matches a graph G=(N', E', N'_L), if there exists an injective function $f : N \rightarrow N'$, s.t., $\forall (u, v) \in E, (f(u), f(v)) \in E'$.

C. Frequent Sub-Graph Mining

The formal definition of Frequent Sub-Graph Mining is given in Definition 3.

Definition 3 (Frequent Pattern Mining): Given a graph dataset $GSet$ and a threshold $msup$, find all patterns that appear in more than $msup$ graphs in $GSet$.

Past results of frequent graph mining (c.f., [15]) have shown that Property 1 defined below holds.

Property 1 (Anti-monotonicity of Support): If a graph pattern P is frequent, so is the graph pattern P', where P' is a sub-graph of P.

Definition 4 describes the concept of frequent *closed* sub-graph patterns. Among patterns having the same support, only maximal patterns are deemed as closed. The formal definition of closed graph mining is given in Definition 5.

Definition 4 (Closed Graphs): Given a set of graphs $GSet$, a graph pattern g is closed, if there does not exists another pattern g' where g' is a super-graph of g and both g and g' match the same set of graphs in $GSet$. If there exists such a g', we say that g is being subsumed by g'.

Definition 5 (Frequent Closed Pattern Mining): Given a graph dataset $GSet$ and a threshold $msup$, find all patterns that are frequent and closed.

III. Proposed Approach

A. Overall Framework

We start with the construction of a collaboration network from SourceForge.Net dataset. We then extract developer collaboration clusters which corresponds to the various connected components in the network. This set of connected components are treated as a graph database which is the basis of our analysis. We refer to this database as the collaboration graph database CGD. Each member of the CGD is a collaboration cluster and is denoted as CC.

In this work, our goal is to mine for two types of patterns: high-level and detailed topological patterns. High-level patterns correspond to the various statistics or counts that we could extract from the collaboration network.

Detailed topological patterns correspond to patterns that appear in many graphs in the CGD. Our goal is to get the top-k most frequent graph patterns from the CGD.

B. Topological Pattern Mining

In this step, we extract top-k graph patterns from the collaboration graph database CGD. We propose an approach that merges graph mining and graph matching. The resultant approach utilizes the strengths and limitations of each technique. The steps are described below:

1) Divide the database CGD into two sub-databases: One containing the large graphs CGD_L and another containing the small graphs CGD_S.
2) As most patterns are small all frequent graphs minable from CGD at minimum support $msup$, could be mined from CGD_S with minimum support $msup'$, where $msup' = (msup - |CGD_L|) \approx msup$.
3) The mining algorithm outputs frequent closed patterns along with their support in CGD_S. Let us denote the support of a pattern P in CGD_S as sup(P,CGD_S).
4) For each graph pattern P that is mined, we try to find graphs in CGD_L that could be matched with P.
5) For each graph pattern P, we store the number of graphs in CGD_L containing it. We denote this value as sup(P,CGD_L).
6) For each graph pattern P, we compute the total support of the pattern. This is equal to sup(P,CGD_S) + sup(P,CGD_L). This number is denoted as sup(P,CGD) or sup(P), which corresponds to the support of the pattern P in the the graph database CGD.
7) We sort the mined patterns in a descending order of their support in CGD.

An end-to-end pseudo-code of our approach is shown in Figure 1.

Procedure Mine Collaboration Patterns
Inputs: SFN: SourceForge.Net Database;
R: Desired range for k;
$msup$: Initial minimum support threshold;
Outputs: Top-k Frequent Graph Collaboration Patterns in SFN;
Method:
1: Let CGD = Extract connected components from developer and project tables in SFN
2: Let CGD_L = {g|g∈CGD∧(|$V(g)$|>254∨|$E(g)$|>254)};
3: Let CGD_S = CGD - CGD_L;
4: Let FPS = {};
5: Do
6: FPS = Run CloseGraph on CGD_S with minimum support set at $msup$
7: If $|FPS|$ is within range R
8: break;
9: Else
10: $msup$ = Reduce the value of $msup$;
11: While $|FPS| < R$
12: For each pattern P in FPS
13: For each graph g in CGD_L
14: If P is contained in g
15: P.support++;
16: Sort P in FPS according to its support
17: Output top-k patterns in FPS

Fig. 1. Mining Topological Collaboration Patterns - Mine-Match

IV. Experiments

We extract a collaboration network from database dumps of SourceForge.Net collected by Madey *et al.* described in [2]. Madey *et al.* dump the database monthly and we take the snapshot extracted at September 2009. There are in total 73 tables with various relationships among them. From the database we investigate 192,706 projects.

We are not interested in inactive projects, thus as an initial step, we filter out projects with no developers and only consider projects which have 100 or more number of downloads. We find that 1,898 projects are not assigned to any developer in the snapshot. Thus, we only process the remaining 190,808 projects. The snapshot only contains information on the number of downloads for 64,487 out of the 190,808 projects. From these 64,487 projects, we find that there are 28,087 projects with 100 or more downloads. From these 28,087 projects, we extract a collaboration network. This collaboration network consists of 55,694 developers.

A. How connected are the developers?

We first investigate whether the collaboration network is a connected one. We find that this is not the case. Indeed the graph is made of many disjoint connected components or collaboration clusters (i.e., CCs). In total there are 6,744 collaboration clusters in the collaboration network.

The number of developers that work alone is very low. There are about 838 developers among 55,694 developers (1.5%) that do not work with anyone else. We also note that there is a very large collaboration cluster (CC) of 30,111 developers (54.07%) which represents the core community of developers. Other CCs are of much smaller size (the second largest CC only contains 117 developers).

B. What are some statistical characteristics of collaboration clusters?

We plot the size of the CCs (x-axis) versus the count or frequency of the CCs having that size (y-axis) in Figure 2. We consider two notions of size: (a) the number of nodes in a CC and (b) the number of edges in a CC. The left graph in Figure 2 observes a power law like behavior, while the one on the right follows a similar downward trend but is "noisy".

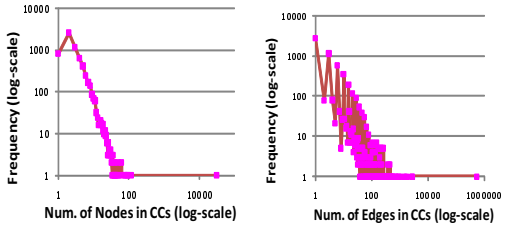

Fig. 2. **All CCs: Size (Num. of Nodes) vs. Frequency (Left), Size (Num. of Edges) vs. Frequency (Right)**

We also compute a measure characterizing the connectivity of a graph. We define connectivity as the number of edges divided by the number of nodes.

We round-off the connectivity value to the nearest 0.5 (e.g, 2.31 is rounded off to 2.5). The plot of connectivity value (x-axis) versus frequency of CCs having that connectivity value (y-axis) is shown in Figure 3 (left). Next, we zoom in to the largest CC of 30,111 nodes representing the core community of developers. We plot the distribution of the degree of the nodes/developers in the large CC in Figure 3 (right). The degree of a node in a graph is the number of neighbors that it has. We observe that Figure 3 (left) observes a power law like behavior, however this is not the case with Figure 3 (right).

We perform a hypothesis test to investigate if the mean of the connectivity values of the CCs of large sizes is substantially different from that of small size CCs. As a null hypothesis we assume that both means are equal. We find that we could reject the null hypothesis as the P value is less than 0.0001 (statistically significant at α= 0.0001). As the connectivity values are generally larger for larger graphs, it suggests that having indirect "neighbor" helps in fostering more collaborations among developers.

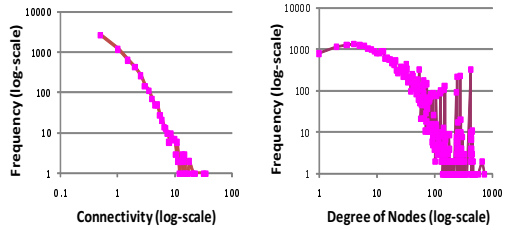

Fig. 3. **Connectivity vs. Frequency (Left), Degree of Nodes vs. Frequency (Right)**

C. What are some common topological collaboration patterns?

We separate graphs having more than 254 nodes or 254 edges to a separate dataset. There are a total of 36 graphs

min. support	$\|P\|$	runtime (CG)	runtime (VF)
990	30	10.22 hours	0.71 hours
980	36	15.54 hours	0.84 hours
970	36	15.49 hours	0.87 hours
960	37	15.87 hours	0.90 hours
950	39	16.69 hours	0.95 hours

TABLE I
NUMBER OF PATTERNS ($|P|$) AND RUNTIME AT VARIOUS MIN. SUPPORT THRESHOLDS FOR CLOSEGRAPH (CG) AND VFLIB (VF)

having more than 254 nodes or 254 edges. There is only one graph with more than 254 nodes *and* 254 edges. This graph has in total 30,111 nodes and 543,559 edges. We also separate developers who only work with one other developer or no other developer. For these very small graphs, we simply keep their counts and would use them in the calculation of the support of the pattern having only two nodes with one edge connecting them[1]. Hence, there are 3 sets of graphs: large, small, and very small.

We run CloseGraph [16] for the small graphs to produce an initial set of patterns. We run VFLib [7] on the large graphs to update the support count of the patterns. The final set of patterns are then reported.

The runtime for CloseGraph and VFLib and the number of patterns mined for different minimum support thresholds are shown in Table I. The result of the experiment extracting the top 30 most common topological collaboration patterns is shown in Figure 4. The 30 patterns are mined with the minimum support threshold set at 990. Our approach is able to scale for the large dataset.

Performing a deeper analysis based on the general observations above, we note the following findings:

1) The frequent collaboration patterns are of relatively small sizes. Many developers are only "professionally" linked to a few other developers. Among the 30 patterns, we only see patterns with at most 6 nodes.
2) The mined patterns suggest that the collaborations between developers follow triadic closure principle [8]. The principle states that if two people in a social network have a friend in common then in the future, there is a likelihood that they will become friends themselves. We observe this when we compare G2 with G3, G5 with G6, G6 with G7, G7 with G8, etc. We could then compute the likelihood of clusters exhibiting G2 to also exhibit G3 (similarly for G2 and G7, and G7 and G8). We notice that the likelihoods are high which are 96.5%, 94.2%, and 99.6% for G2 and G3, G6 and G7, and G7 and G8 respectively.
3) The least dense patterns are the chain like patterns i.e., G1, G2, G4, and G10. The most dense collaboration patterns are the complete graph patterns i.e., G1, G3, G8 and G20. It is interesting to note that the likelihoods of clusters observing a chain like pattern to also observe the complete graph pattern of the same number of nodes are high. They are 96.5%, 92.9% and 90.0% for G2 and G3, G4 and G8, and G10 and G20 respectively. The result suggests that indirect links (i.e., collaborators' collaborators) are likely to realize into direct links (i.e., new collaborations).
4) Some patterns of a lower rank could be derived from patterns of a higher rank by expanding its hub. For example, G2 could be derived by expanding the hub of G1, G6 could be derived by expanding the hub of G3, etc. These suggest that in many collaboration clusters there are one or more developers who actively recruit other developers to join in.

[1]We ignore trivial graphs with only one node as they do not correspond to collaborations.

D. Does six-degree-of-separation exist?

We first analyze the largest CC with 30,111 nodes. We use Java Universal Network/Graph Framework (JUNG)[2] to calculate the diameter and the average value of the shortest paths between two nodes in a graph. The result shows that the diameter of the largest graph is 19 and the average of the shortest paths is 6.55 (≈ 6.6). Thus, our study shows that the six-degree-of-separation exists among developers in the core community of developers.

V. DISCUSSION

In our study, two developers are considered to collaborate on a software project if both of them are listed as contributors of the project in SourceForge.Net. A developer could be listed as a contributor of a project, but does not actually contribute anything to the project. These cases could be viewed as noise in the dataset.

In our initial experiment, we have experimented with a collaboration network formed by developers working on the top-100 projects (based on the number of downloads) in SourceForge.Net. We find 1,902 unique SVN/CVS committer identifiers extracted from this dataset. However, there are several issues with the identifiers. First, the same developer might use different committer identifiers in different projects. Second, different developers might use the same committer identifier in different projects.

Due to the scalability issue and the issue with the uniqueness of committer identifiers, we do not use CVS/SVN logs to construct the collaboration network. Rather, we use the contributor information in SourceForge.Net. Although some developers listed as contributors might not eventually commit code to the project, it is possible that they are involved in the initial planning of the project or in other tasks which are not recorded in the CVS or SVN but are yet valuable to the success of the project.

VI. RELATED WORK

Several researchers have studied social or expertise networks among developers. One of the early work is by Bird *et al.* in [3] that extracts a social network from developers' email communications. This work complements past studies by recovering collaboration patterns, in the form of statistical

[2]Available at: http://jung.sourceforge.net/index.html

Rank	ID	Pattern	N	E	Sup	Dia	Den	Con	Rank	ID	Pattern	N	E	Sup	Dia	Den	Con
1	G1		2	1	5906	1	1	0.5	16	G16		5	6	1370	2	0.6	1.2
2	G2		3	2	3250	2	0.67	0.67	17	G17		5	7	1369	2	0.7	1.4
3	G3		3	3	3135	1	1	1	18	G18		5	6	1349	2	0.6	1.2
4	G4		4	3	2061	3	0.5	0.75	19	G19		5	8	1276	2	0.8	1.6
5	G5		4	3	2061	2	0.5	0.75	20	G20		5	10	1272	1	1	2
6	G6		4	4	2040	2	0.67	1	21	G21		6	5	994	4	0.33	0.83
7	G7		4	5	1922	2	0.83	1.25	22	G22		6	5	994	4	0.33	0.83
8	G8		4	6	1915	1	1	1.5	23	G23		6	5	994	3	0.33	0.83
9	G9		5	4	1416	3	0.4	0.8	24	G24		6	5	993	3	0.33	0.83
10	G10		5	4	1414	4	0.4	0.8	25	G25		6	6	991	3	0.4	1
11	G11		5	5	1400	3	0.5	1	26	G26		6	6	991	4	0.4	1
12	G12		5	5	1398	2	0.5	1	27	G27		6	6	990	4	0.4	1
13	G13		5	5	1379	3	0.5	1	28	G28		6	6	990	3	0.4	1
14	G14		5	5	1371	2	0.5	1	29	G29		6	6	990	2	0.4	1
15	G15		5	6	1370	3	0.6	1.2	30	G30		6	6	990	3	0.4	1

N = Number of nodes
E = Number of edges
Sup = Number of support
Dia = Diameter
Den = Density
Con = Connectivity

Fig. 4. **Collaboration Patterns among Developers in SourceForge.Net**

characteristics and topological patterns, among developers in open-source projects, namely SourceForge.Net.

Lungu *et al.* propose an approach to visualize a super-repository [10]. A related visualization study is also performed by Sarma *et al.* [13]. Our approach is complementary to past studies by Lungu *et al.*'s and Sarma *et al.*'s by mining for patterns from a super-repository containing tens of thousands of diversified projects.

There have been recent studies that investigate the applicability of power-law in software engineering code bases [5], [17]. Our study also shows that power law holds in a number of cases. However, rather than investigating code elements, we investigate developers that work on various code bases.

There have also been several work on the analysis of open source developer communities [11], [14]. None of the studies extract topological collaboration patterns. Also, we enrich result reported in these past studies as described in our technical report [1].

There are a number of studies analyzing globally distributed software development activities [9], [12], [4]. In this study, we complement those studies by analyzing both high-level statistical patterns and detailed graph topological patterns describing collaborations among developers working across tens of thousands of diversified projects in open source settings.

VII. Conclusion and Future Work

In this work, we extract high-level counts and detailed topological graph patterns from a large super-repository, namely SourceForge.Net. We show that not all developers are connected to every other developers. There are many collaboration clusters. The number of nodes of the clusters and connectivity follow power law. But this is not the case for the number of edges and node degrees in the core cluster. We find that the small-world phenomenon also exists where each developers in a connected network is separated on the average by approximately 6 hops. We also mine top-30 topological patterns. We note the likely importance of hubs in fostering collaborations as many mined patterns contain one or even more hubs. The patterns also suggest that "weak" links resulting from transitive relationships between developers in the network are likely to evolve to direct collaborations.

In the future, we plan to develop a recommendation system and study the evolution of network-level statistical counts and detailed collaboration topological pattern as SourceForge.Net evolves over time. We also plan to find "bad" collaboration patterns that correlate to unsuccessful projects.

Acknowledgement. We would like to thank Greg Madey for sharing with us the SourceForge.Net dataset, Xifeng Yan for providing us the binary of CloseGraph, and National Research Foundation (NRF) (NRF2008IDM-IDM004-036) for funding the work.

References

[1] http://www.mysmu.edu/staff/didisurian/papers/devcollabpattern.pdf.
[2] M. Antwerp and G. Madey, "Advancess in the sourceforge research data archive (SRDA)," in *OSS*, 2008.
[3] C. Bird, A. Gourley, P. Devanbu, M. Gertz, and A. Swaminathan, "Mining email social networks," in *MSR*, 2006.
[4] M. Cataldo and J. Herbsleb, "Communication networks in geographically distributed software development," in *CSCW*, 2008.
[5] G. Concas, M. Marchesi, S. Pinna, and N. Serra, "Power-laws in a large object-oriented software system," *IEEE TSE*.
[6] C. R. B. de Souza, S. Quirk, E. Trainer, and D. F. Redmiles, "Supporting collaborative software development through the visualization of socio-technical dependencies," in *ACM SIGGROUP*, 2007.
[7] P. Foggia, "The vflib graph matching library, version 2.0." http://amalfi.dis.unina.it/graph/db/vflib-2.0/doc/vflib.html, 2001.
[8] M. Granovetter, "The strength of weak ties," *American Journal of Sociology*.
[9] J. Herbsleb, D. Paulish, and M. Bass, "Global software development at Siemens: Experience from nine projects," in *ICSE*, 2005.
[10] M. Lungu, M. Lanza, T. Girba, and R. Heeck, "Reverse engineering super-repositories," in *WCRE*, 2007.
[11] G. Madey, V. Freeh, and R. Tynan, "The open source software development phenomenon: An analysis based on social network theory," in *AMCIS*, 2002.
[12] N. Ramasubbu and R. Balan, "Globally distributed software development project performance: An empirical analysis," in *FSE*, 2007.
[13] A. Sarma, L. Maccherone, P. Wagstrom, and J. Herbsleb, "Tesseract: Interactive visual exploration of socio-technical relationships in software development," in *ICSE*, 2009.
[14] J. Xu, Y. Gao, S. Christley, and G. Madey, "A topological analysis of the open source software development community," in *HICSS*, 2005.
[15] X. Yan and J. Han, "gSpan: Graph-based substructure pattern mining," in *Proc. of ICDM*.
[16] ——, "Closegraph: Mining closed frequent graph patterns," in *KDD*, 2003.
[17] H. Zhang, "Power laws in computer programs," *Information Procs. and Management*.