1-2018

Collaboration patterns in software developer network

Didi SURIAN

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

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

Collaboration Patterns in Software Developer Network

Didi Surian[1,3], David Lo[2] and Ee-Peng Lim[2]

1 School of Information Technologies, University of Sydney, Sydney, NSW, Australia

2 School of Information Systems, Singapore Management University, Singapore, Singapore

3 Centre for Health Informatics, Australian Institute of Health Innovation, Macquarie University, Sydney, NSW, Australia

Published in Encyclopedia of social network analysis and mining, 2019, 2nd ed, Cham: Springer, pp. 224-229.

DOI: 10.1007/978-1-4939-7131-2_292

Accepted version

Synonyms

Collaboration patterns; Developer collaboration network; Graph pattern mining

Glossary

Closed graphs

A graph pattern p is closed with respect to a graph database GDB if there is no pattern p′ where p′ is a supergraph of p and both are subgraphs of the same set of graphs in GDB

Collaboration graph

An undirected graph whose nodes represent all developers appearing in the input dataset under study and edges represent a collaboration among the developers. We denote a graph by (N, E, N_L), where N is a set of nodes, E is a set of edges, and N_L is a set of node labels

Frequent subgraph patterns

A graph pattern p is frequent in a graph database GDB (i.e., a set of graphs) with respect to a minimum support threshold msup if p is a subgraph of at least msup graphs in GDB. The number of graphs where p is a subgraph is referred to as the support of p

Subgraph isomorphism

Consider two graphs $G_1 = (N_1, E_1, N_{L1})$ and $G_2 = (N_2, E_2, N_2)$ and two functions $L_1: N_1 \rightarrow N_{L1}$ and $L_2: N_2 \rightarrow N_{L2}$ that map nodes to labels. Subgraph isomorphism is an injective function $f: N_1 \rightarrow N_2$ such that (1) $\forall n \in N_1, L_1(n) = L_2(f(n))$ and (2) $\forall (u, v) \in E_1, (f(u), f(v)) \in E_2$. The function f is referred to as the embedding of G_1 in G_2

Super repository

Logged information of developers working on various projects. One of the largest super repository capturing developers working on various projects around the globe is SourceForge.Net

Definition

In this entry, we mine collaboration patterns from a large software developer network (Surian et al. 2010). We consider high- and low-level patterns. High-level patterns correspond to various network-level statistics that we observe to hold in this network. Low-level patterns are topological subgraph patterns that are frequently observed among developers collaborating in the network. Mining topological subgraph patterns are difficult as it is an NP-hard problem. To address this issue, we use a combination of frequent subgraph mining and graph matching by leveraging the power law property exhibited by a large collaboration graph. The technique is applicable to any software developer network that could be represented as a large graph. As a case study, we experiment with a developer collaboration network extracted from SourceForge.Net, which is the most popular open-source software portal.

Introduction

Nowadays collaborations among people could be done without being hampered by distant locations. This phenomenon has influenced the way how software developers interact with one another. Many projects are completed by collaborative efforts of many developers from various parts of the world. To shed further light on collaborations among developers, in this entry, we analyze a network of developers working with one another on thousands of projects. We are interested in finding both high- and low-level patterns to answer the following research questions:

1. High level: network-level statistics. Are all developers connected to every other developers in the network? How many clusters of connected developers are there in the network? Do the large collaboration clusters appear more often than the small ones?
2. Low level: common topological collaboration patterns. What are some common structures frequently occurring in a large set of clusters of connected developers?

Both the high- and low-level patterns reveal interesting properties of collaborations among developers. We believe understanding properties of developer collaborations is an important first step to designing an effective solution to further improve existing collaborations and create new ones.

Historical Background

In this section, we first introduce some related work in analyzing software developer networks. We then introduce some work in pattern mining. There has been a number of work in software engineering that analyzes social or expertise networks among developers. Xu et al. (2005) and Madey et al. (2002) are among the first that investigate collaborations in open-source community. Some others visualize the socio-technical relationships from developer collaboration networks (Sarma et al. 2009; de Souza et al. 2007). Our work extends theirs by mining for more patterns from a developer collaboration network extracted from a more recent snapshot of SourceForge.Net.

One of the first algorithms that checks for subgraph isomorphism or subgraph matching is proposed by Ullmann (1976). Several extensions to this algorithm are proposed by Cordella et al. (1999) and Foggia et al. (2001). A graph-matching library called VFLib provides the implementations of these algorithms (Foggia 2001). Frequent subgraph mining is an extension to the pattern-matching problem, and it is known to be an NP-complete problem (Cordella et al. 2004). To address this challenge, Yan and Han (2003) proposed the concept of frequent closed patterns to reduce the search space. In this work, we combine the techniques of subgraph matching and frequent subgraph mining to extract topological patterns from a large collaboration network.

Methodology

We start by extracting a collaboration network from a software super repository. We treat developer collaboration clusters which corresponds to the various connected components in the network as a graph database. We refer to this database as the collaboration graph database CGD. We then perform some network-level analysis to find properties related to the connectivity of this graph and the number of clusters of connected developers. Next, we extract low-level topological patterns that are exhibited by many of the clusters. We describe our strategy to mine these topological patterns in the following paragraphs.

To mine for topological patterns, we first split the large collaboration graph into many connected components, each being a cluster of connected developers. This forms our graph database. We then need to perform frequent subgraph mining. Many existing subgraph mining solutions however do not scale for large graphs. For example, the implementation of CloseGraph (Yan and Han 2003) could only handle graphs with at most 254 nodes and 254 edges. How could we scale the frequent subgraph mining algorithm to mine patterns from our dataset?

We note that in our graph database, which consists of clusters of connected developers, most graphs are of small sizes, and only a few are of large sizes. Based on this observation, we perform a divide-and-conquer approach combining graph mining and graph matching to achieve scalability. The following are our proposed approach:

Step 1: Our collaboration graph database, let us refer to it as CGD, is divided into two sub-databases: large graphs CGD_L and small graphs CGD_S. We consider a graph with more than 254 nodes or edges as being large.

Step 2: Mine for frequent patterns from CGDS with minimum support msup′, where msup′ = (msup − | CGD_L |) ≈ msup, and output frequent closed patterns along with their support, i.e., sup(P, CGD_S). We use CloseGraph (Yan and Han 2003) to mine these patterns.

Step 3: Next, for each pattern P mined at step 2, perform a graph-matching process, by using the tool developed by Foggia (2001), to find graphs in CGD_L where pattern P is a subgraph of. The number of such graphs is denoted as sup(P, C GD_L). We use VFLib (Foggia 2001) to perform graph matching.

Step 4: For each graph pattern P, its total support (denoted as sup(P, CGD) or sup(P)), which is equal to sup(P, CGD_S) + sup(P, CGD_L), is computed. We then sort the mined patterns in a descending order based on their total support values.

Experiment and Results

We use database dumps of SourceForge.Net collected by Madey et al. described in Antwerp and Madey (2008) as our dataset. For this study, we take the snapshot extracted on September 2009. From the database we investigate 192,706 projects. We only consider projects with ≥100 downloads and refer these projects as active projects. We find that there are 28,087 active projects. From these 28,087 projects, a collaboration network which consists of 55,694 developers is extracted.

High Level: Network-Level Statistics

We find that the collaboration graph of SourceForge.Net is made of many disjoint connected components or collaboration clusters (i.e., CCs). There are 6,744 collaboration clusters. We also find that only a very small portion of the developers work alone. Among the 55,694 developers, 838 (1.5%) do not work with any other developers. Furthermore, a very large collaboration cluster (CC) consisting of 30,111 developers (54.07%), which corresponds to the core community of developers,

exists. Other CCs are much smaller in size, for example, the second largest CC only has 117 developers.

Figure 1 plots the number of CCs of different sizes (number of nodes or number of edges) – the graphs (b) and (d) zoom into CCs of size ≤20. From Fig. 1a, c, we note that CCs of small sizes appear much more frequently than those of large sizes. If the plot of two variables follow a straight line in a log-log graph, they are likely to follow power law. Thus, a power lawlike behavior is observed by the graphs drawn in Fig. 1a, b.

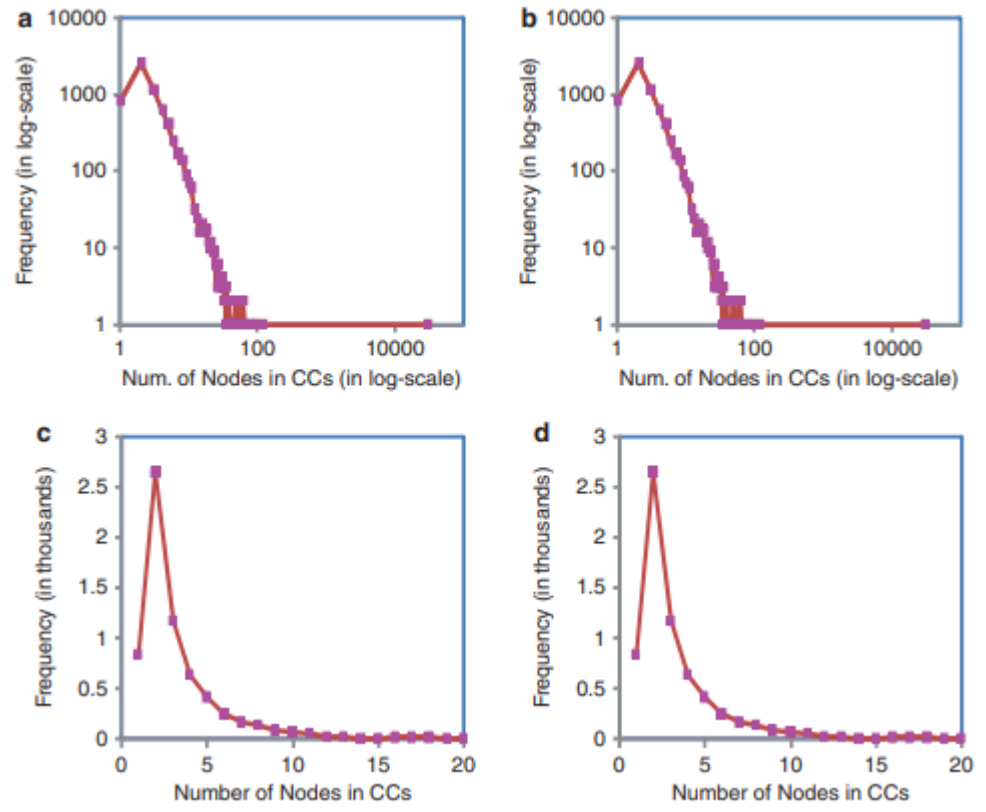

Fig. 1 Collaboration Patterns in Software Developer Network

Size (num. of nodes) vs. frequency (a) all CCs, (c) CCs with num. of nodes ≤20 and size (num. of edges) vs. frequency (b) all CCs, (d) CCs with num. of edges ≤20

To investigate whether six degrees of separation exists in the software developer network, we analyze the largest CC with 30,111 nodes. We calculate the diameter of the CC and the average length of the shortest paths between two nodes in the CC using JUNG (Java Universal Network/Graph Framework). Following (Leskovec 2010), only 1,000 randomly sampled nodes from the graph are considered; we calculate the shortest paths for all possible pair of nodes in this 1,000 randomly sampled nodes. We find that the diameter is 19 and the average shortest path lengths is 6.55 (≈6.6). This result confirms that six degrees of separation exists in the core community of software developers that we analyze.

Low Level: Common Topological Collaboration Patterns

We divide the graphs into three sets of graphs, large, small, and very small, based on their number of nodes and edges. There is one large graph which has more than 254 nodes and 254 edges, and there are 36 large graphs having more than 254 nodes or 254 edges. We categorized graphs who have only one node as very small graphs. We exclude these graphs as they do not correspond to collaborations. The other graphs are referred to as small graphs.

For small graphs, we run CloseGraph (Yan and Han 2003), which is a closed frequent subgraph mining algorithm, to produce an initial set of patterns. Then we run VFLib (Foggia 2001), which is a graph-matching algorithm, on the large graphs to update the support count of the patterns. At the end, the final set of frequent patterns is reported.

The result of the experiment extracting the top 30 most common topological collaboration patterns is shown in Fig. 2. We put the collaboration patterns in descending order based on their supports.

Legend:
N : Number of nodes **E** : Number of edges **Sup** : Support **Dia** : Diameter **Den** : Density **Con** : Connectivity

Rank	ID	Pattern	N	E	Sup	Dia	Den	Con	Rank	ID	Pattern	N	E	Sup	Dia	Den	Con	Rank	ID	Pattern	N	E	Sup	Dia	Den	Con
1	G_1		2	1	5906	1	1	0.5	11	G_{11}		5	5	1400	3	0.5	1	21	G_{21}		6	5	994	4	0.33	0.83
2	G_2		3	2	3250	2	0.67	0.67	12	G_{12}		5	5	1398	2	0.5	1	22	G_{22}		6	5	994	4	0.33	0.83
3	G_3		3	3	3135	1	1	1	13	G_{13}		5	5	1379	3	0.5	1	23	G_{23}		6	5	994	3	0.33	0.83
4	G_4		4	3	2061	3	0.5	0.75	14	G_{14}		5	5	1371	2	0.5	1	24	G_{24}		6	5	993	3	0.33	0.83
5	G_5		4	3	2061	2	0.5	0.75	15	G_{15}		5	6	1370	3	0.6	1.2	25	G_{25}		6	6	991	3	0.4	1
6	G_6		4	4	2040	2	0.67	1	16	G_{16}		5	6	1370	2	0.6	1.2	26	G_{26}		6	6	991	4	0.4	1
7	G_7		4	5	1922	2	0.83	1.25	17	G_{17}		5	7	1369	2	0.7	1.4	27	G_{27}		6	6	990	4	0.4	1
8	G_8		4	6	1915	1	1	1.5	18	G_{18}		5	6	1349	2	0.6	1.2	28	G_{28}		6	6	990	3	0.4	1
9	G_9		5	4	1416	3	0.4	0.8	19	G_{19}		5	8	1276	2	0.8	1.6	29	G_{29}		6	6	990	2	0.4	1
10	G_{10}		5	4	1414	4	0.4	0.8	20	G_{20}		5	10	1272	1	1	2	30	G_{30}		6	6	990	3	0.4	1

Fig 2: Collaboration Patterns in Software Developer Network

Mining topological collaboration patterns from SourceForge.Net

From Fig. 2, 3, 3, 14, and 10 patterns appear in ≥3,000, 2,000–3,000, 1,000–2,000, and 900–1,000 CCs, respectively. Patterns G_1, G_2, G_4, and G_{10} have "linear chain" structures. Patterns G_5, G_9, and G_{21} are "fork" like. A few patterns, for example, G_{20}, have a higher level of connectivity among nodes. Patterns G_2, G_5, G_6, G_7, G_8, G_{12}, G_{16}, G_{17}, G_{18}, G_{19}, G_{20}, and G_{29} contain a "hub" which is a node that is connected to every other node in the pattern. Among these patterns, patterns G_3, G_8, and G_{20} have multiple hubs. Pattern G_5 has a "star graph" structure where there exists a node that links to all other nodes, while the other nodes are disconnected with one another. Patterns G_1, G_3, G_8, and G_{20} are complete graphs which show that everyone collaborates with everyone else.

Key Applications

From our experiment results, we find that there are many CCs, and most topological patterns are of small sizes. We also notice that some lower-rank topological patterns (i.e., patterns that appear less often) could be formed from higher-rank patterns (by adding nodes and edges). This suggests that the triadic closure principle, which states that if two individuals have a friend in common, then they are likely to become friends too, is observed. These observations are potentially useful for developing a collaboration recommendation system to foster further collaborations among software developers. Our analysis focuses on software developer network; however, the proposed approach could also be

applied to other networks that involve collaborations among people that could be represented as a graph.

Future Directions

As more and more open-source developers collaborate online, there are more and more data that are available for analysis and research. There are several promising future research directions. It is interesting to build a recommendation system that could effectively recommend developers and projects to other developers and projects. It would also be interesting to perform a longitudinal study to investigate how software collaboration patterns and developer networks evolve over time. Another interesting study would be an analysis on the impact of collaboration patterns to the success of a project. For example, one could recover the "good" and "bad" collaboration patterns that correlate to successful and unsuccessful projects, respectively.

Cross-References

Data Mining

Graph Matching

Online Communities

Acknowledgments

We would like to thank Greg Madey for sharing the SourceForge.Net dataset, Xifeng Yan for providing the binary of CloseGraph, and National Research Foundation (NRF) (NRF2008IDM-IDM004-036) for funding the work. This work was done while the first author was with the School of Information Systems, Singapore Management University.

References

Antwerp M, Madey G (2008) Advances in the sourceforge research data archive (SRDA). In: International conference on open source systems (OSS), Milano

Cordella L, Foggia P, Sansone C, Vento M (1999) Performance evaluation of the vf graph matching algorithm. In: IEEE international conference on image analysis and processing (ICIAP), Venice

Cordella LP, Foggia P, Sansone C, Vento M (2004) A (sub) graph isomorphism algorithm for matching large graphs. In: IEEE transactions on pattern analysis and machine intelligence (TPAMI)

de Souza CRB, Quirk S, Trainer E, Redmiles DF (2007) Supporting collaborative software development through the visualization of socio-technical dependencies. International ACM SIGGROUP Conference on Supporting Group Work, Sanibel Island

Foggia P (2001.) The vflib graph matching library, version 2.0

Foggia P, Sansone C, Vento M (2001) An improved algorithm for matching large graphs. In: IAPR-TC15 workshop on graph-based representations

Leskovec J (2010) Snap: network datasets. http://snap.stanford.edu/data/index.html. Accessed Mar 2010

Madey G, Freeh V, Tynan R (2002) The open source software development phenomenon: an analysis based on social network theory. In: Americas conference on information systems (AMCIS), Dallas

Sarma A, Maccherone L, Wagstrom P, Herbsleb J (2009) Tesseract: interactive visual exploration of socio-technical relationships in software development. In: International conference on software engineering (ICSE), Vancouver

Surian D, Lo D, Lim EP (2010) Mining collaboration patterns from a large developer network. In: Working conference on reverse engineering (WCRE), Beverly

Ullmann J (1976) An algorithm for subgraph isomorphism. J ACM 23(1):31–42. MathSciNet

Xu J, Gao Y, Christley S, Madey G (2005) A topological analysis of the open source software development community. In: Hawaii international conference on system sciences (HICSS), Big Island

Yan X, Han J (2003) Closegraph: mining closed frequent graph patterns. In: International conference on knowledge discovery and data mining (ACM SIGKDD), Washington