What Dynamic Network Metrics Can Tell Us About Developer Roles

Mathias Pohl
Dept. for Software Engineering
University of Trier, Germany
pohlm@uni-trier.de

Stephan Diehl
Dept. for Software Engineering
University of Trier, Germany
diehl@uni-trier.de

ABSTRACT

Software development is heavily dependent on the participants of the process and their roles within the process. Each developer has his specific skills and interests and hence contributes to the project in a different way. While some programmers work on separate modules, others developers integrate these modules towards the final product. To identify such different groups of people one approach is to work with methods taken from social network analysis. To this end, a social network has to be defined in a suitable way, and appropriate analysis strategies have to be chosen. This paper shows how a network of software developers could be defined based on information in a software repository, and what it can possibly tell about roles of developers (and what not) in the process of the application server TOMCAT.

Categories and Subject Descriptors

D.2.8 [**Software Engineering**]: Metrics—*complexity measures, performance measures*

General Terms

Human Factors, Measurement

Keywords

Social Network Analysis, Identifying Roles

1. INTRODUCTION

Development of software is a highly collaborative task. Modern software systems are too complex to be built by a single person. Even the programming of a single module usually is done by several people. In particular, there are two common ways of collaboration among developers. On the one hand developers can work together on purpose. Two people knowing each other personally or sharing the same interests will try to find a solution for their task together. In terms of open source software this can be achieved by communication via email or instant messaging in addition to mailing lists or the bug reporting forums. Unfortunately, this information (if it can be retrieved at all) usually is quite incomplete. On the other hand, there is also unintentional collaboration. This can be a result of working on the same resources, i.e. on the same files in the project. For example, a programmer who wants to fix a certain bug can change several files that have been edited by others before. If the bug was a simple coding error, then there would be no need for further collaboration. However, if it was a result of design flaws then the redesign of the module will be put on the agenda which again results in more collaborative activities, and further changes to the same files by these developers. Our research goal is to detect these and similar collaboration patterns based on archived information, including software repositories, mailing lists, and bug-databases.

This paper presents an initial case study based on the information from the TOMCAT-project [1]. In this first case study the co-author relationship is interpreted as a network that is analyzed in a very basic way. To identify changes in this network it is separated in several time-slices to allow dynamic analysis. Although the basic information is not very reliable and only few analytic effort is done the results give an impression what dynamic collaboration networks could tell about the development process of software. In particular, the dynamic aspect distinguishes our work from previous work on analyzing static co-author networks [6] or static email networks [2].

2. DYNAMIC DEVELOPER NETWORK

As already mentioned, the co-author relationship can possibly provide interesting insights into developer roles. To allow the analysis of these relationships with methods known from social network analysis a developer network has to be created. The definition of the network is straight-forward as all information is simply merged. To allow dynamic analysis the network is sliced into several distinct networks that may or may not have different structures. More formally a *(static) developer network* can be defined as follows:

Let V be the set of all programmers committing to a project and let t be a period of time. Then the *developer network* N^t is an ordered pair $N^t = (V, E)$ where

$$E \;:=\; \{\{p,q\} \in \mathcal{P}_2(V) \mid p \text{ and } q \text{ contributed to the same file during period } t\}\,.$$

The term $\mathcal{P}_2(V)$ denotes the set of all subsets of V with size 2. Figure 1 shows a developer network for TOMCAT.

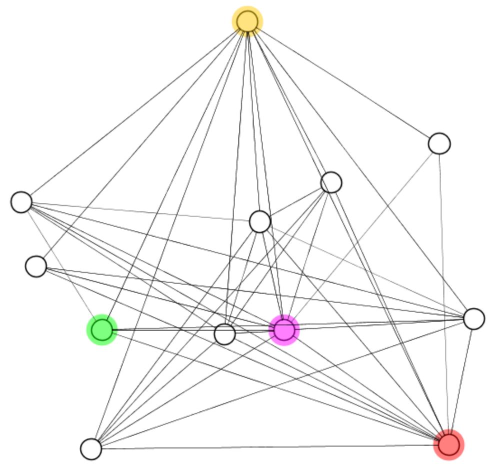

Figure 1: The developer network for Tomcat at January 4th, 2000. In this network only transactions on Java-files are considered.

A *dynamic developer network* $\mathcal{N}$ is a sequence of developer networks where each network is restricted to a different period of time. More formally:

$$\mathcal{N} := \left(N^{t_1}, \dots, N^{t_n}\right)$$

The periods $t_1, \dots, t_n$ can be obtained using different algorithms. The simplest one is to divide the overall development time in equally sized intervals.

After obtaining a dynamic developer network a whole bunch of properties can be computed for all elements of each network in the sequence [4]. However, so far it is not quite clear which property can really give useful insight into the collaboration of developers.

In the following study the dynamic network is analyzed with respect to the so-called *betweenness* [5] of selected developers within the TOMCAT-project. The betweenness centrality of a node indicates how many shortest paths between different actors in the network contain this node. The rationale of this measure is that actors routing many connections in the network possibly have a much higher influence within the community.

Formally this centrality measure is computed as follows: If $N^t = (V, E)$ is a network then the betweenness centrality $C_B(v)$ for a node $v \in V$ is defined as

$$C_B(v) = \sum_{s \in V, t \in V, s \neq t s \neq v \neq t} \frac{d_{s,t}(v)}{d_{s,t}}$$

where

$d_{s,t}$ is the set of all shortest paths between s and t

and

$d_{s,t}(v)$ is the set of all shortest paths between s and t containing v.

To obtain relative values $C_B(v)$ can be normalized to $C'_B(v)$ by dividing $C_B(v)$ by the number of node pairs that do not contain v, i.e.

$$C'_B(v) := \frac{C_B(v)}{(|V|-1)(|V|-2)}.$$

The betweenness centrality for the nodes in a network can be computed in time $O(|V|^2 + |V||E|)$ [3].

In context of software development, the betweenness measure can be used to reveal programmers that are "putting things together" and hence have a crucial role in the development process.

3. THE NETWORKS BEHIND TOMCAT

The analyzed networks are taken from version 3 of TOMCAT. This project has already been analyzed previously with respect to collaboration and developer roles [7]. The researched time period starts in October 1999 and ends in November 2004. For the definition of the dynamic developer network this period is equally divided into 128 slices (which results in an equal distance of around 14 days between each consecutive slice). To smooth the computed betweenness centrality values each slice contains the information of 90 days of development (and not just the correspondent 14 days). The smoothness then is based on the fact that this definition makes relational information from transactions present in several slices.

For the purpose of this paper, we restricted the analysis to four selected developers of the project. In Figure 2 the betweenness values for these four developers is shown. It reveals that the developer depicted by yellow color is one of the most "integrating" person throughout the development cycle. At the beginning of the inspected period many people were contributing which resulted in a very dense network. In such networks there exist no nodes with outstanding betweenness centrality which explains why yellow is not that high at the left side. When the development of the product came to an end (version 4 of TOMCAT was to be appearing) only a few changes are committed to the repository and hence the network's connectivity decayed. This explains why the centrality values fall down to zero during the period.

In addition to identifying programmers that play a central role it is interesting how their role really looks like. When restricting the network to co-author relationship of JAVA-files (Fig. 3) then there is only a small difference to the complete network. Still yellow has one of the highest betweenness values during the period and the curve looks also similar to that from Fig. 2.

However, inspecting only documentation files (identified by their filename extension `.html` and `.txt`) the image looks different (Fig. 4). At the beginning of the development two developers worked on these files together but this pattern disappears relatively fast. During a long period those working together in the source code seem to collaborate only at a few points in time. This artifact might indicate different efforts towards code quality and documentation quality and hence should be analyzed in more detail. It should be noted that a co-author network for documentation files is always pretty small in TOMCAT (see Fig. 7). Although betweenness centrality can heavily vary in small networks for different nodes a higher similarity could be expected.

The curves in Fig. 5 show the values for the network restricted to commits to source code in the C-programming

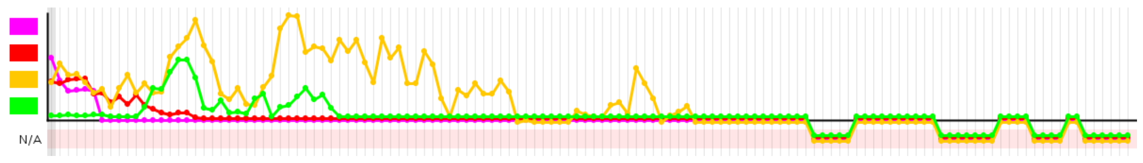

Figure 2: The betweenness centrality values based on all transactions for the selected programmers. As in all other figures each curve consists of 128 distinct points.

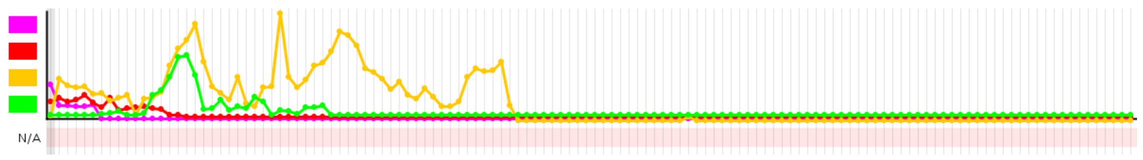

Figure 3: The betweenness centrality values based on all transactions containing changes of `.java`-files for the selected programmers. The plotted curves look similar to those in Fig. 2.

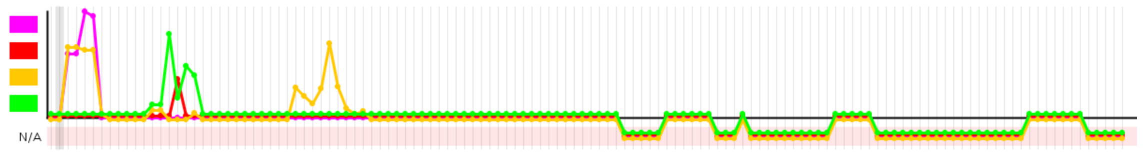

Figure 4: The betweenness centrality values based on all transactions affecting documentation files (`.html` and `.txt`) for the selected programmers.

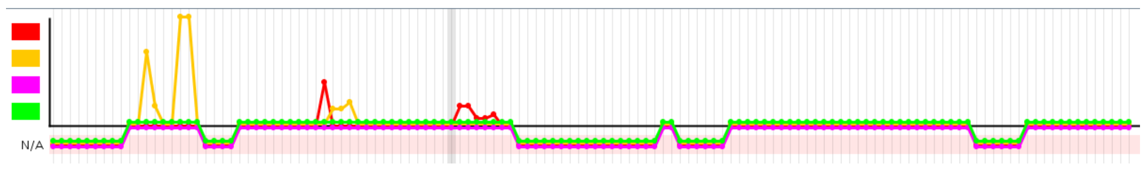

Figure 5: The betweenness centrality values based on all transactions related to source files written in the C programming language for the selected programmers.

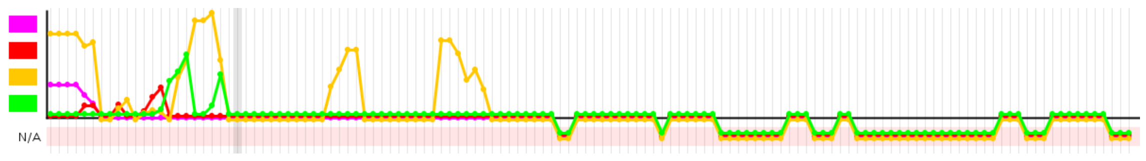

Figure 6: The betweenness centrality values based on all transactions concerning build scripts for the selected programmers.

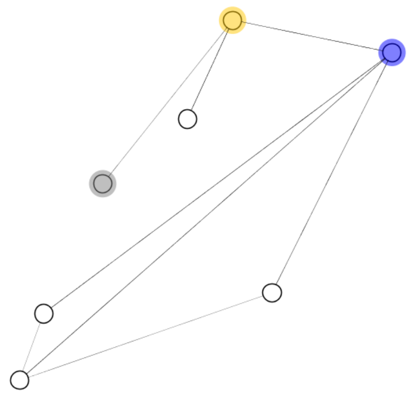

Figure 7: The developer network for Tomcat at January 4th, 2000 considering transactions on documentation files only.

language (identified by the filename extension `.c` and `.h`). Although this part of the software is supposed to be written once and for all (this is the connecting module for the Apache webserver) there is some development going on. Finally build scripts (Ant- and makefiles) are edited in a similar way than the source code files (see Fig. 6). This similarity indicates that those programmers editing source code are also aware of the build scripts which is usually done by the integrated development environment. Since this result is not surprising it suggests that the betweenness values of the described developer network is not a random value but a possible metric to work with in the analysis of software development processes.

4. DISCUSSION

Developers of a software system in some sense form a social network. As illustrated in this paper such a network can give answers to important questions towards quality management of software. As an example those people working together on a specific module should also work together on the module's documentation. In the illustrating example of Tomcat this property does not completely hold true at least not according to the computed centrality value.

However, since the data basis is not very large this approach is not sufficiently reliable. To obtain a more powerful tool for this kind of analysis we want to combine information from several data sources (e.g. email, bug reports, etc.) as part of our future work. Furthermore there exist many more network characterization measures that could be indicating interesting facts. Many of these may turn out useful for gain insights into software development processes based on the above mentioned data sources.

Acknowledgement

Mathias Pohl is partially supported by *Deutsche Forschungsgemeinschaft (DFG)*, grant no. DI 728/6-2.

5. REFERENCES

[1] The Tomcat-project. `http://tomcat.apache.org`, last visited on 24th of January, 2008.

[2] C. Bird, A. Gourley, P. Devanbu, M. Gertz, and A. Swaminathan. Mining email social networks. In *Proceedings of the 2006 international workshop on Mining software repositories MSR06*, pages 137–143, New York, NY, USA, 2006. ACM.

[3] U. Brandes. A Faster Algorithm for Betweenness Centrality. *J. of Math. Society*, 25(2):163–177, 2001.

[4] U. Brandes and T. Erlebach, editors. *Network Analysis: Methodological Foundations*, volume 3418 of *Lecture Notes in Computer Science*. Springer, 2005.

[5] L. C. Freeman. Centrality in Social Networks. Conceptual Clarification. *Social Networks*, 1(3):215–239, 1979.

[6] L. Lopez, J. M. Gonzalez-Barahona, and G. Robles. Applying Social Network Analysis to the Information in CVS repositories. In *First International Workshop on Mining Software Repositories, MSR 2004 (ICSE Workshop), Proceedings*, 2004.

[7] P. Weißgerber, M. Pohl, and M. Burch. Visual Data Mining in Software Archives to Detect How Developers Work Together. In *Fourth International Workshop on Mining Software Repositories, MSR 2007 (ICSE Workshop), Proceedings*, page 9, 2007.