Unveiling Collaboration Structures in Software Development Projects

Michael Schwind
Technical University Kaiserslautern, schwind@wiwi.uni-kl.de

C Wegmann
TU Kaiserslautern, wegmann@wiwi.uni-kl.de

Oliver Wendt
Technical University Kaiserslautern, Germany, wendt@wiwi.uni-kl.de

Follow this and additional works at: http://aisel.aisnet.org/ecis2008

Recommended Citation
http://aisel.aisnet.org/ecis2008/43
UNVEILING COLLABORATION STRUCTURES IN SOFTWARE DEVELOPMENT PROJECTS

Michael, Schwind, Technical University Kaiserslautern, 67653 Kaiserslautern, Germany, schwind@wiwi.uni-kl.de
Christian, Wegmann, Technical University Kaiserslautern, 63753 Kaiserslautern, Germany, wegmann@wiwi.uni-kl.de
Oliver, Wendt, Technical University Kaiserslautern, 67653 Kaiserslautern, Germany, wendt@wiwi.uni-kl.de

Abstract

We investigate the structures of collaboration in software development groups by analyzing the data derived from the versioning system SVN (subversion) for software projects of different types and sizes. Our research is part of the project “Economics of Networks in Information Systems” (EONIS) which aims to investigate the impact of network structures on the effectiveness of infrastructures, organizations and processes in the domain of information systems. The goal of this subproject in EONIS is to reveal the network topology of collaboration groups for development projects connected with proprietary software in industry as well as in open source software projects. For this purpose we have developed a software tool that is able to analyze the update data in SVN systems with respect to the changes in the software code and defined a collaboration coefficient that is able to express the strength of their relationship in terms of collaboration in the network. We present the first results from the analysis of an open source project with about 126 software developers by constructing a weighted network from a matrix representation of the collaboration coefficients that have been calculated for each pair of software developers.

Keywords: Collaboration Structure, Open Source Software, Versioning System, Network Topology
1 INTRODUCTION

In this paper we present the first results of our research project “Economics of Networks in Information Systems” (EONIS) which aims to investigate the impact of social network structures on the effectiveness of infrastructures, organizations and processes in the information system (IS) domain. The goal of our work is to reveal the structures of collaboration in software development groups by analyzing the data of the versioning system SVN\(^1\) (subversion) for software projects of different types and sizes. The project’s goal is to analyze the network topology of collaboration groups in software development projects for proprietary software production (PSP) processes as well as for open source software (OSS) communities. For this purpose we have developed the software tool SVNNAT that is able to analyze the update data in SVN systems with respect to the changes in the software code and define a coefficient that is able to express the degree of collaboration between the actors in the software development groups. With the SVNNAT tool it is possible to trace and visualize the evolution of the collaborative relationships between the software developers over time. SVNNAT is able to represent the developers’ relationships as a network graph by forming a weighted matrix structure from their collaboration coefficients. The nodes of the resulting network represent the software developers (actors) and the weighted edges indicate their degree of collaboration. Based on the graph we investigate the typical structure of collaboration networks in a software production group with about 120 developers. Finally, we show how we intend to reveal differences between the properties of the network topology for PSP and OSS collaboration groups.

2 COLLABORATION AND NETWORK STRUCTURES

2.1 The Network Topology of Socio-Technical Systems

For a long time random networks have been assumed to be the basic model for the topology of social networks (Erdős & Rényi, 1959), until in 1998 a new type of network, called small-world topology, was discovered by Watts and Strogatz (1998). These networks are characterized by the existence of many strongly clustered sub-graphs that are connected to each other only by a few long distance connections. This property corresponds to Granovetter’s hypothesis of “The Strength of Weak Ties” which claims that the real power of social networking is not derived from a multitude of strong ties to neighbours in individuals’ local social groups of but from infrequent weak ties to distant social groups (Granovetter, 1973). Activated by these results and enabled by the fact that the rapidly growing Internet allowed researchers to measure exactly how socio-technical networks evolve, another type of real world network was discovered by Albert-László and Barabási (1999): scale-free topology. Scale-free property denotes that the degree distribution\(^2\) of a network is a fat-tailed exponential distribution. This scale-free property has been identified as existing for several types of socio-technical networks, like the Internet, e-mail networks, or citation and scientific collaboration networks (Barabási et al., 2002; Vazquez et al., 2002; Ebel et al., 2002).

2.2 Software Collaboration Graphs and Network Structure

Inspired by the results of this socio-technical network analysis, the impact of social relations on productivity became an issue in the research on effectiveness in software development. The impact of

---

\(^1\) http://subversion.tigris.org

\(^2\) The degree of distribution of a network determines the degree of the nodes in a network (simply counts the number of edges to each node) and plots their frequency in a diagram. For further properties of scale-free networks see Hein et al. (2006).
collaboration on software quality and effectiveness has been investigated for OSS projects in particular.

Several studies have tried to identify the structure of collaboration in software development networks. However, the direct impact of the implicit network structures on these benchmark parameters has not yet been evaluated.

We categorize three types of approaches while looking at some recent literature in this domain:

- **The first approach** is to extract the network structure deriving from class dependencies directly by analyzing classes, methods, and modules in object-oriented software code. The resulting graphs, called ‘class collaboration graphs’, are based purely on the code level of software (mostly deriving from JAVA or C++ software) and primarily reflect information about the functional dependencies in software projects. The structure of the developers’ collaboration network can only be revealed indirectly by this method. Network topologies resulting from the analysis of class collaboration graphs regularly exhibit parameters that are typical of scale-free networks (Valverde et al., 2002; Valverde & Sole, 2007; Myers, 2003; Hyland-Wood et al., 2006). Lopez-Fernandez et al. (2006) present a CVS-based approach to the analysis of software collaboration structures that strongly relies on social network analysis techniques. They present two main views on the collaboration structures that can be extracted from CVS data: module networks, where each node represents a particular software module and an edge between two nodes indicates that at least one software author has contributed to both modules and committer networks, where the nodes represent the software authors and an edge between two nodes indicates that two authors have contributed to at least one common module. By analyzing the statistical properties of module and committer networks for the OSS projects Apache, Gnome and KDE, Lopez-Fernandez et al. identify small-world properties for the resulting network structures.

- **The second approach** primarily addresses the social layer of the collaboration structure in software development networks. By analyzing the e-mail traffic between the developers as well as newsgroup threads that result from the reporting of program errors and problems, this first group analyzes the social layer directly. This data is mostly extracted from communication functions that support versioning and development systems like CVS (concurrent versions system)³ (Nakakoji et al., 2005; Crowston et al., 2005). The first analytical results from this data also support the assumption that these collaboration networks show a power-law degree distribution like most other social networks (Valverde et al., 2006).

- **The third approach** is based on the analysis of versioning data derived from CVS and SVN databases. These systems contain the changes and the updates of the corresponding program code directly and are therefore well suited to reveal the collaboration structures and the behavioural patterns of the actors in the software development networks (Koch & Schneider, 2002). The result would be a topology indicting collaboration on the socio-technical level. The software tool ‘SoftChange’ is an example of such a CVS-based approach. However, their research is focussed on the development of collaboration between the actors in OSS networks over time and ignores the collaboration topology (Mockus & German, 2003). The eclipse-based software tool ‘Ariadne’ analyzes dynamic invocation relationships between procedures, called ‘social call graphs’, derived from code information in CVS repositories of collaborative software development projects. Social call graphs are closely related to the class collaboration graphs considered in the first type of approach; however, the pure code information is enhanced with patterns of social collaboration information which has been directly extracted from the eclipse software development environment (De Souza et al., 2007). This approach does not consider the application of measures derived from social network analysis.

³ http://www.nongnu.org/cvs/
2.3 Analysis Method of SVNNAT

We have chosen the socio-technical approach described in the previous section in order to analyze collaboration graphs in OSS and PSP software projects.

Our approach calculates an index that expresses the degree of collaboration between the developers in a software project. For this purpose we search for local clusters of code changes in the SVN system which have been ‘committed’ by pairs of authors. By scanning the entire programming code, we can calculate a correlation index between all the clusters that have been identified in the project. The result is a ‘collaboration’ matrix that shows the degree of collaboration between all the developers of this software. Because program code has a hierarchical structure, we have to repeat the clustering and correlation process on the different programming levels (code blocks, files, subprojects etc.). In this paper the clustering process is described for different levels of aggregation in the programming code of a project. Fig. 1 illustrates the levels of analysis for the program code in SVNNAT.

![Figure 1. Levels of analysis for the program code in SVNNAT](image)

To calculate a value for the collaboration of two authors we distinguish between four levels of detail. The detailed information we have about the file contents are blocks of code, their order and length in a given revision and the authors of these blocks.

![Figure 2. Distance measures used for SVNNAT’s search for clusters on the first level (program code).](image)

- The first and lowest level of collaboration between two authors is the correlation between two code blocks produced by these authors. We define a distance measure for the distance between two code blocks. If code block $B$ is behind code block $A$, the distance $\Delta AB$ between both is the...
difference between the first line number of code block $B$ and the last line number of code block $A$ less 1 to get a distance of 0 for adjoining code blocks. With this distance measure we can define a modified cosine function to get a high correlation between neighboring code blocks produced by two authors and also a small correlation for more distant code blocks (see Fig. 2).

$$\text{lineCorrelation}(\Delta_{AB}) = \begin{cases} 0, & \text{if } \Delta_{AB} \geq \Delta_{\text{max}} \\
\cos\left(\frac{\Delta_{AB} - \pi}{2}\right), & \text{if } \Delta_{AB} < \Delta_{\text{max}} \end{cases}$$

The modifications lead to a collaboration of 1 for code blocks with $\Delta_{AB} = 0$ and a collaboration of 0 for code blocks with the distance $\Delta_{AB} \geq \Delta_{\text{max}}$ which is a given maximum distance.

- The second level of collaboration considers code clusters in a file. Code clusters are generated by a cluster algorithm which assigns a code block to its nearest neighboring code block if this neighbor is not further away than the maximum distance $\Delta_{\text{max}}$. The cluster algorithm considers all code blocks produced by two authors and can also create clusters which contain only a single author. A cluster is composed of a set of code blocks from author A and a set of code blocks from author B. If one of these sets is empty the correlation is 0. The correlation in a single cluster can be calculated in two ways: 
  - **average linkage (AL)**
  - **nearest neighbor (NN)**.

**Average Linkage Correlation:**

$$\text{clusterCorrelation}(\text{Cluster}_x) = \frac{\sum_{x \in A \cup B} \text{lineCorrelation}(\Delta_{AB})}{|A| \cdot |B|}$$

**Nearest Neighbour Correlation:**

$$\text{clusterCorrelation}(\text{Cluster}_x) = \frac{\sum_{x \in A \cup B} \text{lineCorrelation}(\Delta_x)}{|A| + |B|}$$

$\Delta_x$ is the distance between the code block $x$ and its nearest neighbour for the other author.

Both correlations are normalized and return a cluster correlation between 0 and 1 since the code block correlation is a value between 0 and 1.

- The third level of collaboration considers the correlation within a whole file. We define the correlation within a file as the weighted sum of cluster correlations. The number of lines in a cluster serves as a weighting factor.

$$\text{fileCorrelation}(\text{File}_f) = \frac{\sum_{\text{Cluster}_x \subset \text{File}_f} \text{clusterCorrelation}(\text{Cluster}_x) \cdot |\text{Cluster}_x|}{\sum_{\text{Cluster}_x \subset \text{File}_f} |\text{Cluster}_x|}$$

$|\text{Cluster}_x|$ is the number of lines by both authors in the cluster $x$.

- The fourth and highest level of collaboration is the collaboration of both authors under consideration in the whole software project. It is calculated as the weighted sum of file correlations. The number of lines created by both authors serves as weighting factor.

$$\text{authorCorrelation}(\text{Author}_a, \text{Author}_b) = \frac{\sum_{\text{File}_f \subset \text{Files}_{AB}} \text{fileCorrelation}(\text{File}_f) \cdot |\text{File}_f|}{\sum_{\text{File}_f \subset \text{Files}_{AB}} |\text{File}_f|}$$

$\text{Files}_{AB}$ contains all files on which author A and author B have participated in the revision under consideration. $|\text{File}_f|$ is the number of lines by both authors in the file $f$. 
2.4 Functionality of SVNNAT

SVNNAT uses a separate database to store the relevant versioning data that is extracted from the SVN host systems. Once connected to the SVN system, the data concerning the code changes in the software project under analysis is transferred to this database. Every file modified at a ‘commit’ is assigned to the participating authors. Fig. 3 (left side) shows the resulting list of authors who have committed changes in the OSS development process of the SVN itself, as displayed by SVNNAT. After the transfer of the relevant data has been completed, SVNNAT is able to calculate the values of the collaboration matrix for the agents involved in the development process according to both methods presented in section 3.1. Fig. 3 (right side) shows the results deriving from the calculation of the collaboration matrix for the latest transferred revision of the SVN project.

Fig. 3. SVNNAT presenting a list of software developers who made changes to the SVN project (left) and the calculation results for the corresponding collaboration matrix (right).

Fig. 4 shows several visualizations for the results of the data extraction and calculation process performed by SVNAT. On the upper left side one can see the collaboration matrix for one file that uses a colour scale from light yellow to dark red for visualization purposes. Light yellow indicates weak ties between the lines of program code written by a pair of software developers while dark red indicates a strong correlation between the lines of program code written by a corresponding pair of software developers. Fig. 4 on the upper right side shows the history of code changes over the time for the whole project (lines of code aggregated) committed by the software developers involved in the project. The aggregation starts with the first revision of the program code in the time period selected for collaboration analysis. The plane graph on the lower left side of Fig. 4 depicts the number of lines in one file that have been changed per revision by the corresponding software developers (lines of code aggregated). More detailed information is given using the plane graph on the lower right side of Fig. 4 (lines of code not aggregated). Our visualization method follows the history flow concepts that have been applied to study cooperation and conflict between editors in Wikipedia projects (Viegas et al., 2004).
Figure 4. **SVNNAT** displaying the results of the software update analysis in the OSS SVN project: collaboration matrix of one single file (upper left), history of all lines developed by developers (upper right), aggregated number of lines developed by a developer (lower left) and a detailed view of the number of lines developed by a developer (lower right).

Additionally **SVNNAT** has the following important features:

- It makes it possible to export the network structure that results from the calculation of the correlation matrix employing the methods presented in section 3.1 using the data format of the network analysis toolset Pajek⁴. Pajek has been developed especially for the detailed analysis of social network structures in terms of statistical properties such as degree distribution, betweenness or centrality (de Nooy et al., 2005).

• It allows the direct interactive 2-dimensional visualization of the collaboration networks that have been retrieved by SVNNAT using the Java Universal Network / Graph Framework JUNG\(^5\).

• One can calculate and visualize the degree distribution of the networks retrieved by SVNNAT based on weighted and non-weighted edges. The data for the degree distribution can also be exported to other programs in order to analyze the resulting network type.

• The authors’ identity can be anonymized in order to guarantee non disclosure of sensitive data if the SVNNAT software is used in commercial environments.

2.5 Results

SVNNAT has been applied to the analysis of two software projects: The OSS project SVN (subversion)\(^6\) itself and a PSP project of medium size enterprise that works as a supplier for software development in the German industry. Fig. 5 shows the graph of two small collaboration networks resulting from the analysis of the PSP project. The correlation matrix data was used to generate the collaboration graph employing Pajek. The nodes in the graph indicate the developers and the edges are interpreted as collaboration ties, while the greyscale of the edges indicates the collaboration intensity. Light grey edges indicate weak collaboration in contrast to black edges that demonstrate a strong collaboration. For better visualization the nodes in the 3-dimensional network model have been arranged using the spring and gravity embedding algorithm that was proposed by Fruchterman & Reingold (1991).

![Figure 5. 3-dimensional network graphs resulting from the calculation of the collaboration matrix for 15 software developers in a PSP project employing the average linkage method (left) and the nearest neighbour method (right). The nodes indicate the developers and the edges the collaboration ties while the greyscale indicates the intensity of collaboration. The light grey edges indicate weak collaboration in contrast to black edges that demonstrate the closest collaboration.](image)

Fig. 6 depicts the collaboration network that has been calculated using the average linkage method with \(\Delta_{\text{max}} = 100\) based on the SVN data of our PSP project with 21 software developers as generated by SVNNAT visualization capability. The same method of analysis has been applied to the OSS project to retrieve the collaboration network shown in Fig. 7. The colour of the edges represents the degree of collaboration in the network. Dark red edges indicate close collaboration between the authors. The SVNNAT visualization uses a 2-dimensional version of the Fruchterman and Reingold (FR) algorithm to cluster the nodes of the network.\(^7\) On the upper right side of the OSS collaboration network one can recognize a cluster of closely collaborating software developers within this project which is indicated by the dark tone of the edges.

---

\(^5\) [http://jung.sourceforge.net/]

\(^6\) [http://subversion.tigris.org]

\(^7\) Currently the FR algorithm uses only the number of edges in a node to calculate the clustering. A version that employs the weight of the edges is in preparation.
Figure 6. 2-dimensional SVNNAT visualization of the collaboration network of 21 software developers in a PSP project employing the average linkage method with $\Delta_{\text{max}} = 100$.

Figure 7. 2-dimensional SVNNAT visualization of the collaboration network of 126 software developers in an OSS project employing the average linkage method with $\Delta_{\text{max}} = 100$. 
After having retrieved the collaboration network from the SVN data, we now want to investigate what type of network has evolved using SVNNAT analysis. Looking at analysis of similar networks with weighted edges, such as citation and author collaboration networks, suggests that it is probable that our network has a scale-free topology (Barthelemy et al., 2005). However, the work of Lopez-Fernandez et al. (2006) comes to the conclusion that small-world structures are the prevalent network type in module and committer networks as described in section 2.3. For this reason we have employed the estimation procedure proposed by Clauset et al. (2007) and tried to fit our data into a scale free degree distribution. Clauset et al. provide a MATLAB procedure that calculates the best power-law fit using the maximum likelihood method together with a Kolmogorov-Smirnov goodness-of-fit test. Fig. 8 shows the logarithmic graph of the weighted degree distribution for the OSS collaboration network. The x-axis depicts log of weighted degrees, and the y-axis denotes the associated cumulative distribution function. For the left side of the graph the average linkage method has been used, whereas the log-degree distribution on the right side is derived from a network that was retrieved from the SVN data using the nearest neighbour method. Both samples were generated using a distance measure with a cut-off threshold of $\Delta_{max} = 100$ according to that depicted in the graphs of Fig. 2.

![Logarithmic graph of the weighted degree distribution for the collaboration network of 126 software developers in an OSS project employing the average linkage method (left) and the nearest neighbour method with $\Delta_{max} = 100$.](image)

However, based on the procedure described Clauset et al. (2007), we had to reject our hypothesis that the collaboration network we detected has scale-free properties for all the types of networks we retrieved from the SVN data of the OSS project subversion. For the PSP project a significant rejection of the hypothesis that the network retrieved from the SVN data is of a small-word type could not be rejected significantly due to its small number of nodes. We think that the degree distribution of our collaboration networks is more close to that of an exponentially growing network (Dorogovtsev & Mendes, 2002). The networks retrieved have also not yet been tested for small-world properties as suggested by Lopez-Fernandez et al. (2006). The next goal of our research is to identify the exact network type that best fits the data of the collaboration network we have derived from the analysis of software projects with SVNNAT. For this purpose we need a reliable base of empirical data. We intend to retrieve data from OSS projects, such as Mediawiki, Python and KDE with more software developers involved, to reach a better significance level of our tests on the resulting collaboration networks.

---

8 http://www.mathworks.de/
networks. Additionally, we will compare the topological properties of these collaboration networks with respect to differences between OSS and PSP projects.

3 CONCLUSION

In this paper we have introduced a method of calculating coefficients measuring collaboration between software developers based on the data of the versioning system subversion (SVN) for open source software (OSS) projects as well as for proprietary software production (PSP) in industry. Our research investigates the impact of the collaboration structures in software production networks on the effectiveness of software development projects. For this purpose we have developed the software tool SVNAT that is able to analyze the revision and update data in SVN systems by calculating to the collaboration coefficient we have defined. The coefficient aggregates the information about collaboration in four stages, beginning with pattern of lines of code, continuing on the code module and file level, and ending on the authors’ collaboration level. The coefficient is used to construct a weighted network where the nodes represent the developers and the edges represent their collaboration relationship. Our first results derive from the analysis of an open source project with about 126 software developers. Interestingly the network structure retrieved from the collaboration relationships of the software developers does not follow the scale-free degree distribution that is often observed in the context of socio-technical networks. In order to be able to measure the performance and the relevance of particular actors in the software development network, we intend to employ network specific parameters, such as betweenness centralization, closeness centralization and degree centralization. However, these parameters have to be coupled with other indicators of success and productivity in the software production process in order to allow a valid rating of a software developer’s role and value for the project within the collaboration network. One problem of our measurement system is that mere activity indicated by changing code lines alone does not allow a reliable assessment of the value of a software developer’s work for the entire project. This is because such code changes can also result from erroneous changes of code lines that interfere with the project’s progress. We therefore have to introduce quality indicators for the code contributed to the project by the collaboration partners.

From the practical point of view our approach should enable project managers to have a better control on distributed software development projects. After identifying crucial players in the collaboration network a project manager can put the key players into positions, where their positive impact can foster the overall performance of the developer group. The same applies to the staffing of a new software development project. In this context, our approach could also be used to identify synergies between the collaborators, if specific attributes are attached to the nodes (actors) in the software developer network.

Our next goal in this ongoing research project is to identify the exact type of network that fits our data for a broader base of empirical data. We also intend to characterize the differences between collaboration networks that are derived from OSS and PSP projects. The integration of quality factors for the code changes and the integrated visualization and assessment of the value of a software developer’s work will be the next step in the further development of SVNAT.

References

