Exploring Collaboration Networks in Open-Source Projects

Andrejs Jermakovics, Alberto Sillitti, Giancarlo Succi

Exploring collaboration networks in open-source projects

Andrejs Jermakovics, Alberto Sillitti, and Giancarlo Succi
Free University of Bolzano-Bozen, Piazza Domenicani 3, 39100
Bolzano-Bozen, Italy {ajermakovics,asillitti,gsucci}@unibz.it,
WWW home page: http://www.case.unibz.it/

Abstract. Analysis of developer collaboration networks presents an opportunity for understanding and thus improving the software development process. Discovery of these networks, however, presents a challenge since the collaboration relationships are initially not known. In this work we apply an approach for discovering collaboration networks of open source developers from Version Control Systems (VCS). It computes similarities among developers based on common file changes, constructs the network of collaborating developers and applies filtering techniques to improve the readability of the visualized network. We use the approach in case studies of three different projects from open source (phpMyAdmin, Eclipse Data Tools Platform and Gnu Compiler Collection) to learn their organizational structure and patterns. Our results indicate that with little effort the approach is capable of revealing aspects of these projects that were previously not known or would require a lot of effort to discover manually via other means, such as reading project documentation and forums.

Keywords. Collaboration networks, version control systems, open source

1 Introduction

Analysis of collaboration networks in a development environment can provide decision support for improving the software development process [24]. It has already been widely used in open source [1] and closed source software for exploring collaboration [32], predicting faults [10, 21, 25], studying code transfer [23] and many other activities [4, 7]. Moreover, since software artifact structure is strongly related to the organization's structure (*Conway's Law*) [5] it becomes important to understand the developer networks involved. The analysis of these networks is often leveraged using visualizations and their appearance plays a significant role in how people interpret the networks [14], [20]. It is, therefore, important that the network visualizations are easy to interpret and represent the actual network as closely as possible.

A common problem is that the actual social networks are not known and need to be discovered. Many existing approaches rely on communication archives to discover the networks [1, 6, 29]; however these are not always available and do not necessarily reflect collaboration on code [22]. Additionally mapping people across multiple communication systems (issue trackers, forums, email) involves considerable manual effort [26]. Another possibility is to use dedicated software for tracking development time and pair programming effort [9] but such studies require prior setup.

Most common source of developer networks is the Version Control Systems (VCS) [13, 19, 31]. The underlying idea is that frequent access and modification on the same code implies communication and sharing. The advantages of using VCS is that they are commonly available for all software development activities, can be mined automatically without human involvement and directly reflect collaboration on code.

Once a developer network is constructed it can be analyzed to improve understanding of the software development process and the organizational structure. The recovered organizational structure then can be used in informing new collaborators or observing the integration of new members. Additional applications include tracking code sharing, finding substitute developers with related code knowledge and assembling communities with prior collaboration experience.

In this work we study the collaboration networks of three open source projects in order to learn their organizational structure and collaboration patterns. To do so we use an approach that mines collaboration networks from version control systems and computes similarities between developers based on commits to common files. Once the similarities are computed the network is visualized using a force-directed graph layout.

2 Approach

Our proposed approach [18] uses VCS to mine commits to source code files that developers make. It then computes similarities between committers and visualizes them in a network using similarities as link strengths. In cases when the network is too dense it offers multiple filtering techniques to reduce the number of links.

A crucial part of the approach is the similarity measure because it is used as the basis for visualization and filtering. For our purposes, we adopted an established similarity measure, which is also used in Collaborative Filtering techniques [28], [30] of Recommender Systems. A number of user similarity measures are derived from the ratings that users assign to items. In our case, we use source files as the items and the number of changes as the rating, which is similar to an approach for recommending software components [20].

Cosine similarity between two developers is obtained by calculating the cosine of the angle between their corresponding vectors d_i and d_j:

$$similarity(d_i, d_j) = \cos(d_i, d_j) = \frac{d_i \cdot d_j}{|d_i||d_j|} = \frac{\sum_k d_{i,k} d_{j,k}}{\sqrt{\sum_k d_{i,k}^2 \sum_k d_{j,k}^2}} \quad (1)$$

The similarity accounts for files that were modified only a few times and also for files that many people modify. Thus it is greater if the two developers modified the same files a large number of times and 0 if they did not share any files.

The approach has been previously validated [18] on two projects where the structure was known and was able to discover actual developer networks. It is implemented in a software visualization tool Lagrein [16, 17] which shows software metrics together with collaboration networks. The tool provides interactive exploration of collaboration networks and user adjustable link filtering. Since networks initially appear very dense due to a large number of links, the tool allows removing low weight links to view the network at different levels of detail.

2.1 Network Visualization

A common choice for social network visualizations [12, 29] is to use multidimensional scaling (MDS) [3, 8] or force-directed algorithms for graph layout [11].

Force-directed algorithms model graph vertices as having physical forces of attraction and repulsion. They iteratively compute vertex positions until the difference between desired and actual distances is minimized. Their advantage is that groups with high connectivity are placed together and similar vertices are placed closer than dissimilar ones.

We apply Fruchterman-Reingold [11] force-directed graph layout to the constructed network by setting the link lengths to developer dissimilarity in order to place similar developers together and dissimilar ones apart. The size of each node in the network is proportional to the number of commits the developer made and no link is created between developers with similarity 0. In cases when the visualization appears complex due to a large number of links, we apply filtering to remove low similarity links.

For visual appeal, links are visualized with transparency (using alpha blending). The transparency of a link is proportional to the similarity – high similarity links are solid while low similarity links are transparent. Thus strong links can be spotted immediately and the viewer can see which edges will disappear first during filtering.

Most developer networks are initially too dense due to large number of links. This makes the networks harder to read and hinders force-directed graph layout. For these reasons we filter out low weight edges using a user specified threshold and re-apply graph layout.

3 Case Study: phpMyAdmin project

Initially, we repeated an existing study [13] of the phpMyAdmin[1] project to compare the resulting developer networks. The project is a web application for administering MySql relational database management systems and is written in PHP, HTML and JavaScript. It lets database administrators create/manage databases and tables, edit data and execute SQL statements. The tool is widely used by system administrators and after fifteen years is a stable and mature product. It has also received multiple awards and several books have been written about its usage. Details about the project's development are outlined in Table 1.

Table 1. PhpMyAdmin project details

Property	Value
Repository type	Subversion (SVN)
Analyzed period	2001-2004
Codebase size	100 KLOC
Commits	10K
Languages	PHP, HTML, JavaScript
Committers	16

3.1 Project History

The project was started as a web frontend for MySQL in 1998 by Tobias Ratschiller who was an IT consultant at the time. Due to lack of time he abandoned the project but it already was one of the most popular MySQL administration tools having a large number of people willing to contribute (Table 2) and a large number of patches coming in. At that time in 2001 the project was taken over by Marc Delisle, Olivier Müller and Loïc Chapeaux who registered the project on SourceForge.net[2] project hosting site and the development has continued there ever since.

Table 2. PhpMyAdmin project contributors

Contributor name	Commiter id	Role in the project
Marc Delisle	lem9	Project Manager / Founder
Olivier Müller	swix	Developer / Founder
Loïc Chapeaux	loic1	Developer / Founder
Michal Čihař	nijel	Project Manager
Robin Johnson	robbat2	Developer
Garvin Hicking	garvinhicking	Developer

[1] http://www.phpmyadmin.net

[2] http://sourceforge.net/projects/phpmyadmin/

Contributor name	Commiter id	Role in the project
Alexander M. Turek	rabus	Developer

3.2 Collaboration network

Similarly to our approach, the developer network was extracted from the version control system, however the links were created when two developers committed to the same directory and all links had the same weight. With a network obtained using such approach the authors notice that it is impossible to determine the importance of each developer and conclude that all developers play the same role. They also mention that the network might be misleading due to link computation at directory level. We confirm this observation and discover a different structure in the project's network using our approach for the same period (until 2004).

First we compute the links the same way – at directory level and without assigning weights to them. The resulting network is similar in layout to the network in the previous study and, indeed, no particular structure is evident (**Fig. 1**) due to the density of edges.

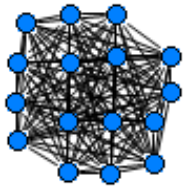

Fig. 1. PhpMyAdmin collaboration network computed at directory level. All contributors appear to have an equal role in the project.

Afterwards, we apply the proposed approach and apply link filtering. We can notice that there are two main groups. By looking at the changed files, we noticed that these groups work on different sets of files, however most of these files are located in the root directory of the project. For this reason, link computation at directory level produced dense and compact network and we conclude that the computation is more reliable at file level.

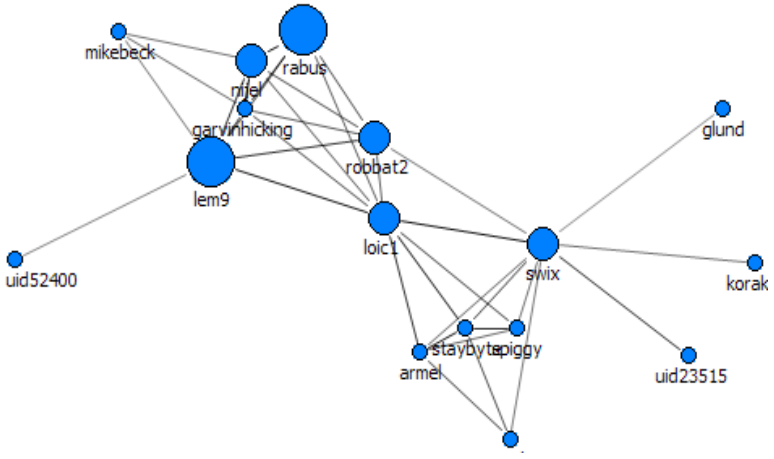

Fig. 2. phpMyAdmin collaboration network computed at file level and filtered links. Some contributors appear having a more central role

While exploring the modified file list we noticed multiple developers, which are the only committers to some files. They work on their own subset of files and no one else works on these files. They make many commits (large nodes in the graph) and also are well connected to other developers indicating significant collaboration activity. Thus we conclude that all developers do not play the same role and there are some with more central and important roles. We later confirmed this by examining sourceforge.net[3] and project's home page where several such developers (lem9, nijel, swix, loic1) are mentioned as project managers and maintainers.

4 Case Study: Eclipse DTP Project

Having experimentally selected [18] Cosine similarity and filtering as effective methods for discovering team structure we proceeded to apply the technique to the Eclipse Data Tools Platform (DTP) project[4]. The choice of the project is arbitrary and the goal of the study is to demonstrate the use of the approach in revealing aspects of the organizational structure that are not known beforehand. Details of the project's development are outlined in Table 3.

Table 3. Eclipse DTP project details

Property	Value
Repository type	Concurrent Versions System (CVS)
Analyzed period	2005-2010
Codebase size	1.4 MLOC
Commits	90K
Languages	Java, XML
Committers	25
Subprojects	5

Eclipse Data Tools Platform is a collection of tools and frameworks for database handling and provides an abstract API over database drivers, connections and data so that they can be used in a generic way. It also provides UI inside Eclipse to define database connections and to execute SQL statements. Originally it was started by Sybase in 2005 and later attracted a large community, which is managed by a committee consisting of Sybase, IBM and Actuate. The project is large (1.4 MLOC) and is composed of several subprojects: Connectivity, Enablement, Incubator, Model Base and SQL Development Tools.

[3] http://sourceforge.net/projects/phpmyadmin/
[4] http://www.eclipse.org/datatools/

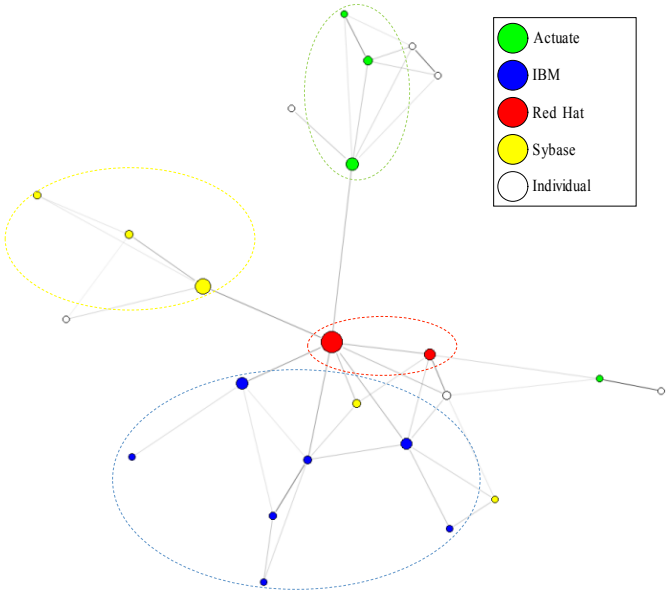

Fig. 3. Eclipse DTP project committer network. There is higher collaboration of contributors within each company

The Eclipse project provides information on its committers using its Commits Explorer[5]. This application allowed us to learn that contributions to the project have been made by many individual committers and multiple organizations including Actuate, IBM, Red Hat and Sybase. Using its CVS repository we constructed the developer network of 25 people for the period 2005-2010. **Fig. 3** shows the resulting network obtained with link filtering and having each company colored in different color.

The visualization of the network allows us to gain quick insight into the organizational structure of the project. We can see a large contribution from IBM in terms of number of people however Red Hat stands in a more central role. An interesting aspect is that contributors do not contribute equally to all parts of the project. They collaborate closely with other contributors from the same company and to a much lesser extent with contributors from other companies.

[5] http://dash.eclipse.org/

5 Case Study: Gnu Compiler Collection (GCC) Project

The GNU Compiler Collection (GCC)[6] is a large system of compilers that supports numerous programming languages (C, C++, Java, Objective-C, Fortran, Ada, Go) and compiles to native code of many processor architectures. It is one of the oldest open source projects and has a large number of contributors developing its numerous front-end and back-end projects. It is produced by the GNU Project[7] with Richard Stallman (a.k.a RMS) and now is widely used as the standard compiler on popular Unix-like operating systems, including Linux and BSD.

The organizational process of the project is described as "cathedral" style by Eric S. Raymond [27] due to the fact that the project was under strict control by the Free Software Foundation (FSF). Many developers that were not satisfied with this model started their own forks of projects and formed the EGCS (Experimental GNU Compiler System). EGCS saw more activity than the GCC development and therefore later was made the official version of GCC. As a result the project opened up more and adopted a more "bazaar" style development model to allow more contributions. Studies of the project [33] show that the development process is largely maintenance and less new software creation. The details of the project's development are outlined in Table 4.

Table 4. GCC project details

Property	Value
Repository type	Subversion (SVN)
Analyzed period	1988-2010
Codebase size	8 MLOC
Commits	100 K
Languages	C/C++, Java, Fortran and others
Committers	350

One part of GCC is the GNU Fortran (GFortran) project whose purpose is to develop the Fortran compiler front end and run-time libraries for GCC. GFortran development is part of the GNU Project. Initially in 2000 it was developed by Andrew Vaught as project g95 that was a free Fortran 95 standard compiler using GCC backend. Andrew wrote most of the parser and for a while work on g95 continued to be collaborative until the late 2002 when he decided to be the sole developer of g95. Project GFortran then was forked from the g95 codebase and collaborative development has continued there since together with integration with the GCC codebase. Since the forking both codebases have significantly diverged. Most of the interface with GCC was written by Paul Brook.

[6] http://gcc.gnu.org/
[7] http://www.gnu.org/

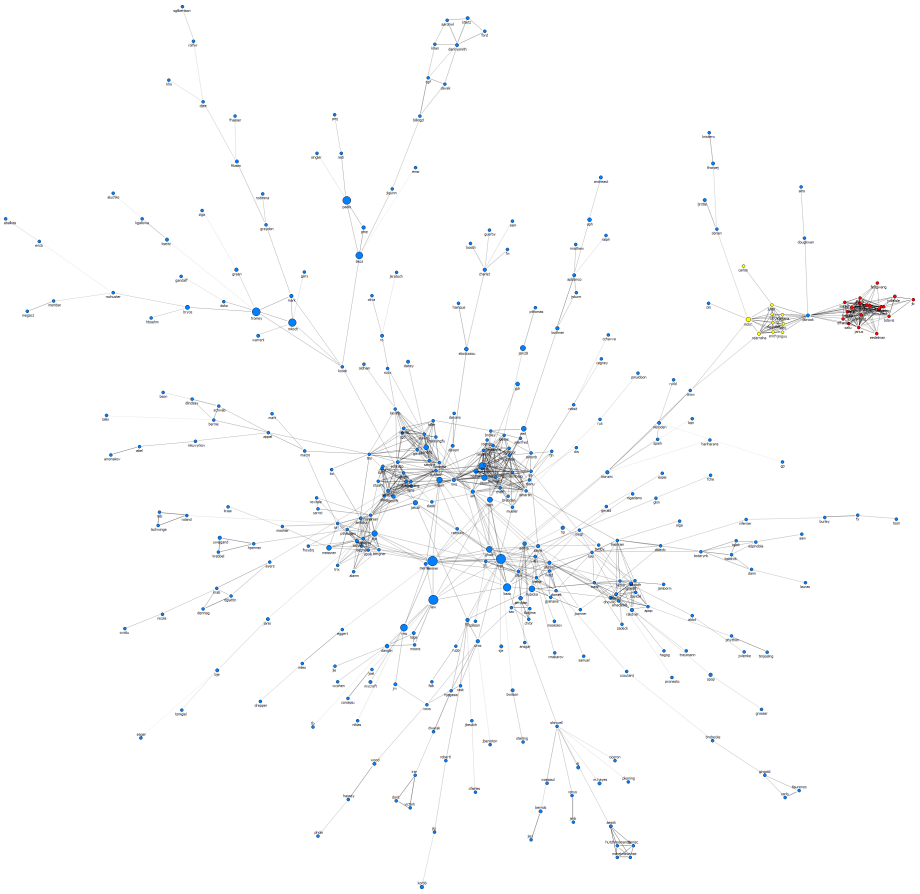

Fig. 4. GCC Collaboration network with Fortran community in top-right (red).

We extracted and analyzed their Subversion commit log in the period from 1988-2010 containing commits from 349 committers. From the collaborator network we can immediately see a large and strongly connected core and a lot of scattered contributors in the periphery around the core. We also can see smaller strongly connected groups suggesting that there are sub-communities within the bigger GCC community as observed in other open-source projects [1].

A particularly interesting aspect is that this network also contains a strongly connected group separate from the core (marked red). By looking at the changes of this group, we can see that they are developing the Fortran front-end because most of their commits were to */gcc/fortran* and */gcc/libgfortran* directories. These directories are listed on the project's homepage as the ones where contribution takes place. Thus we can discover that this community is rather closed because it mostly collaborates

among its own members and to a much lesser extent with the rest of the GCC contributors.

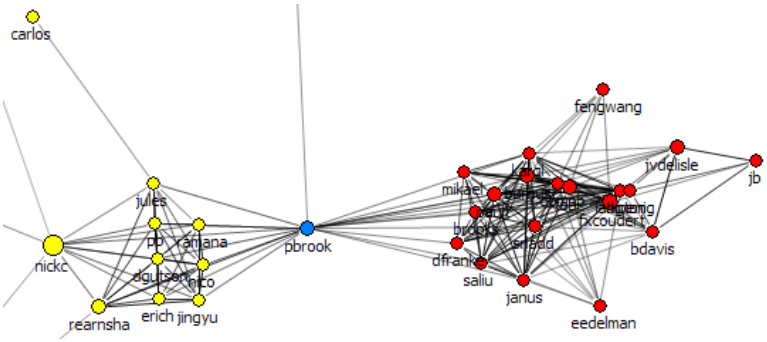

Fig. 5. ARM and Fortran communities in GCC

When we zoom in we can see another closed group to the left of Fortran community. That group (highlighted in yellow) mostly works on the ARM architecture since their commits were in the */gcc/config/arm* subdirectory. One committer (pbrook) stands out in the middle between the ARM and the Fortran communities indicating a lot of involvement in both. We verified this information using the project's contributions page[8], which acknowledges Paul Brook exactly for his work on GNU Fortran and the ARM architecture. He is also listed to have written most of the GFortran code that interfaces with the rest of GCC. Thus by viewing the network we are actually able to identify communities and roles without the need to go through published information or communication archives.

To summarize, by applying the method on open source projects we conclude that it is able to discover various aspects of the projects that were not evident before. We verified them using additional information from the projects however discovering using visualization involves much less effort. These results also added more confidence in the credibility of the approach and we conclude that it can be useful even if the full scope of usefulness is not established.

[8] http://gcc.gnu.org/onlinedocs/gcc/Contributors.html

6 Conclusions

In this work we studied collaboration networks of three open-source projects using visualizations. The networks were automatically discovered using an approach that analyses software repositories and finds similarities among developers based on commit counts to common files. Collaborating developers are placed closer together so that clusters and close collaboration becomes noticeable. Initially the networks appear very dense therefore we apply filtering to remove low weight links.

We found in phpMyAdmin project that there are developers with central roles and other contributors with peripheral roles. In Eclipse DTP project we noticed that there are contributions from multiple large companies however more collaboration is happening among developers within each company than between companies. Finally in GCC project we have observed that it consists of multiple sub-communities and that Fortran community is more separated from the other communities.

Overall open source projects vary greatly in their organization and collaboration patterns however these are often not documented. Automatic approaches for discovering collaboration networks can thus shed light on the structure of these projects and reveal information that was previously not known.

References

1. Di Bella, E., Sillitti, A., Succi, G. 2013. A multivariate classification of open source developers, Information Sciences, Elsevier, Vol. 221, No. 1, pp. 72 - 83, February 2013.
2. Bird, C., Pattison, D. D'Souza, R, Filkov, V., Devanbu, P. 2008. Chapels in the Bazaar? Latent Social Structure in OSS, in 16th ACM SigSoft International Symposium on the Foundations of Software Engineering, (Atlanta, GA)
3. Borg, I. and Groenen, P. 1997. Modern Multidimensional Scaling: Theory and Applications, Springer-Verlag.
4. Coman I., Sillitti A., Automated Identification of Tasks in Development Sessions, 16th IEEE International Conference on Program Comprehension (ICPC 2008), Amsterdam, The Netherlands, 10 - 13 June 2008.
5. Conway, M. E., 1968. How Do Committees invent?, Datamation, 14(4):28-31, April 1968
6. Crowston, K. and Howison, J. 2005. The Social Structure of Free and Open Source Software. First Monday, 10(2), 2005.
7. Di Bella E., Fronza I., Phaphoom N., Sillitti A., Succi G., Vlasenko J., Pair Programming and Software Defects – a large, industrial case study, Transaction on Software Engineering, IEEE, to appear (DOI 10.1109/TSE.2012.68).
8. Freeman, L. C. 2000. Visualizing Social Networks, Journal of Social Structure
9. Fronza, I., Sillitti, A., Succi, G., 2009. An interpretation of the results of the analysis of pair programming during novices integration in a team. 3rd International Symposium on Empirical Software Engineering and Measurement, ESEM 2009.

10. Fronza, I., Sillitti, A., Succi, G., Terho, M., Vlasenko, J. Failure prediction based on log files using Random Indexing and Support Vector Machines, Journal of Systems and Software 86 (1), 2013 , pp. 2-11

11. Fruchterman, T. M. G. and Reingold, E. 1991. Graph Drawing by Force-Directed Placement, Software-Practice and Experience 21, pp. 1129–1164

12. Heer, J. and Boyd, D. 2005. Vizster: Visualizing Online Social Networks. In Proc. of the 2005 IEEE Symposium on Information Visualization, page 5. IEEE Computer Society, (Washington, DC, USA), INFOVIS'05

13. Huang, S.-K., Liu, K.-M. 2005. Mining version histories to verify the learning process of legitimate peripheral participants. Proceedings 2nd International Workshop on Mining Software Repositories (MSR'05). ACM Press: New York NY; 84–78.

14. Huang, W., Hong, S., and Eades, P. 2006. How people read sociograms: a questionnaire study. In Proc. of the 2006 Asia-Pacific Symposium on Information Visualisation - Volume 60. Australian Computer Society, 199-206.

15. Jeh G. and Widom J. 2002. Simrank: a measure of structural-context similarity. In KDD '02: Proceedings of the eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pages 538–543.

16. Jermakovics A., Scotto M., Sillitti A., Succi G., 2007. Lagrein: Visualizing User Requirements and Development Effort. 15th IEEE International Conference on Program Comprehension (ICPC 2007), Banff, Alberta, Canada, 26 - 29 June 2007.

17. Jermakovics A., Moser R., Sillitti A., Succi G., 2008. Visualizing Software Evolution with Lagrein. 22nd Object-Oriented Programming, Systems, Languages & Applications (OOPSLA 2008), Nashville, TN, USA, 19 - 23 October 2008.

18. Jermakovics, A., Sillitti, A., Succi, G. 2011. Mining and visualizing developer networks from version control systems, 4th International Workshop on Cooperative and Human Aspects of Software Engineering (CHASE 2011)

19. Lopez-Fernandez, L., Robles, G., and Gonzalez-Barahona, J. M. 2004. Applying social network analysis to the information in CVS repositories. In Proc. of 1st Intl. Workshop on Mining Software Repositories (MSR 2004), pages 101–105, 2004.

20. McCarey, F., Cinnéide, M. Ó., and Kushmerick, N. 2005. Rascal: A Recommender Agent for Agile Reuse. Artif. Intell. Rev. 24, 3-4 (Nov. 2005), 253-276.

21. Meneely, A., Williams, L., Snipes, W., and Osborne, J. Predicting failures with developer networks and social network analysis. In SIGSOFT '08/FSE-16: Proceedings of the 16th ACM SIGSOFT International Symposium on Foundations of software engineering, pages 13–23, New York, NY, USA, 2008. ACM.

22. Meneely, A., Williams, L., 2011. Socio-technical developer networks: should we trust our measurements? In Proceedings of the 33rd International Conference on Software Engineering (ICSE).

23. Mockus, A. 2009. Succession: Measuring transfer of code and developer productivity. In Proc. of the 2009 IEEE 31st International Conference on Software Engineering (May 16 - 24, 2009). IEEE Computer Society, Washington, DC, 67-77.

24. Petrinja, E., Nambakam, R., Sillitti, A., 2009. Introducing the OpenSource Maturity Model. 2nd Emerging Trends in FLOSS Research and Development Workshop at ICSE 2009, Vancouver, BC, Canada.

25. Pinzger, M., Nagappan, N., and Murphy, B. 2008. Can developer-module networks predict failures?. In Proceedings of the 16th ACM SIGSOFT International Symposium on Foundations of Software Engineering (Atlanta, Georgia, November 09-14, 2008). SIGSOFT '08/FSE-16. ACM, New York, NY, 2-12

26. Pinzger, M., Gall, H. C. 2010. Dynamic analysis of communication and collaboration in OSS projects. Collaborative Software Engineering. Springer, 265-284

27. Eric S. Raymond. The Cathedral and the Bazaar. O'Reilly & Associates, Inc., Sebastopol, CA, USA, 1999

28. Resnick, P., Iacovou, N., Suchak, M., Bergstrom, P., and Riedl, J. 1994. GroupLens: An Open Architecture for Collaborative Filtering of Netnews. In Proceedings of CSCW '94, Chapel Hill, NC.

29. Sarma, A., Maccherone, L., Wagstrom, P., and Herbsleb, J. 2009. Tesseract: Interactive visual exploration of socio-technical relationships in software development. In Proceedings of the 2009 IEEE 31st International Conference on Software Engineering (May 16 - 24, 2009). International Conference on Software Engineering. IEEE Computer Society, Washington, DC, 23-33

30. Shardanand, U. and Maes, P. 1995. Social information filtering: algorithms for automating "word of mouth". In Proceedings of the SIGCHI Conference on Human Factors in Computing Systems. ACM Press/Addison-Wesley Publishing Co., New York, NY, 210-217

31. de Souza, C. R., Quirk, S., Trainer, E., and Redmiles, D. F. 2007. Supporting collaborative software development through the visualization of socio-technical dependencies. In Proceedings of the 2007 international ACM Conference on Supporting Group Work. GROUP '07. ACM, New York, NY, 147-156

32. Wolf, T., Schröter, A., Damian, D., Panjer, L. D., and Nguyen, T. H. 2009. Mining Task-Based Social Networks to Explore Collaboration in Software Teams. IEEE Softw. 26, 1 (Jan. 2009), 58-66.

33. Yamauchi, Y., Yokozawa, M., Shinohara, T., & Ishida, T. (2000, December). Collaboration with Lean Media: how open-source software succeeds. In *Proceedings of the 2000 ACM conference on Computer supported cooperative work* (pp. 329-338). ACM.