Collaboration in Open-Source Projects: Myth or Reality?

Yuriy Tymchuk, Andrea Mocci, Michele Lanza
REVEAL @ Faculty of Informatics – University of Lugano, Switzerland

ABSTRACT

One of the fundamental principles of open-source projects is that they foster collaboration among developers, disregarding their geographical location or personal background. When it comes to software repositories collaboration is a rather ephemeral phenomenon which lacks a clear definition, and it must therefore be mined and modeled. This throws up the question whether what is mined actually maps to reality.

In this paper we investigate collaboration by modeling it using a number of diverse approaches that we then compare to a ground truth obtained by surveying a substantial set of developers of the Pharo open-source community. Our findings indicate that the notion of collaboration must be revisited, as it is undermined by a number of factors that are often tackled in imprecise ways or not taken into account at all.

Categories and Subject Descriptors

D.2.7 [**Software Engineering**]: Version Control

General Terms

Software Repositories

Keywords

Software ecosystems, collaboration

1. INTRODUCTION

Figure 1 is not abstract art. It is a graph depicting the Pharo[1] open-source community in terms of collaborating developers. It contains 940 nodes, the developers, and 13,832 edges where each edge represents the fact that two developers have collaborated, by committing source code to the same project repository. The average node degree is 29.43, which means each developer seems to have collaborated with nearly 30 other developers. One might be tempted to think that what we have here is a highly collaborative community, an embodiment of the open-source philosophy [12].

[1]See `http://pharo-project.org`.

MSR'14, May 31 – June 1, 2014, Hyderabad, India
Copyright 2014 ACM 978-1-4503-2863-0/14/05...$15.00
http://dx.doi.org/10.1145/2597073.2597093

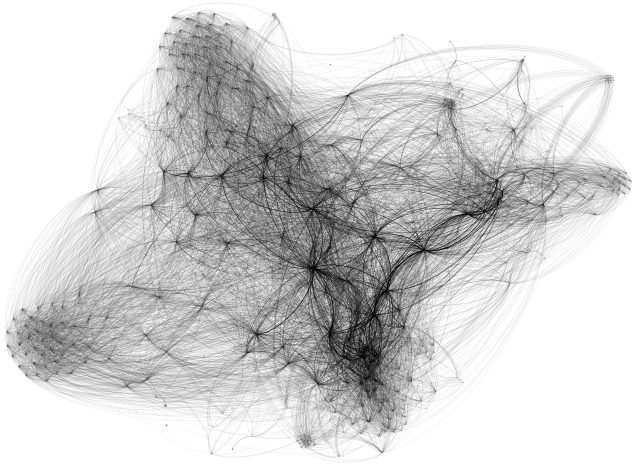

Figure 1: Collaboration between all users of SmalltalkHub

We have mined the underlying data from SmalltalkHub[2], a super-repository (*i.e.,* "a container of several projects developed in parallel" [9]) similar to GitHub. The question that we investigate is *"to what extent does the collaboration information than can be mined from software repositories represent the real and factual collaborations?"*. Researchers have proven that low quality data leads to unreliable assumptions and fragile conclusions [7][1]. In the given context a finding that would reveal that mined collaboration data is unreliable would undermine any research effort in this direction.

To provide an answer to our question, we have established a ground truth on collaboration by surveying several developers of the same open-source community, and have compared existing approaches, as well as novel approaches, which model the concept of collaboration to the ground truth by applying them to the data mined from a super-repository. Our findings reveal that not only is collaboration an ephemeral phenomenon, but that relying on a sole source of information might lead to wrong conclusions. Also, we cannot corroborate the previous findings of Meneely and Williams [10]. We make the following contributions:

1. An investigation of the concept of collaboration in an open-source community, performed by establishing a ground truth through a survey among developers.
2. A comparison of state-of-the-art approaches, as well as custom approaches, with the ground truth, and a discussion on why previous work in the area cannot be corroborated.

[2]See `http://smalltalkhub.com`

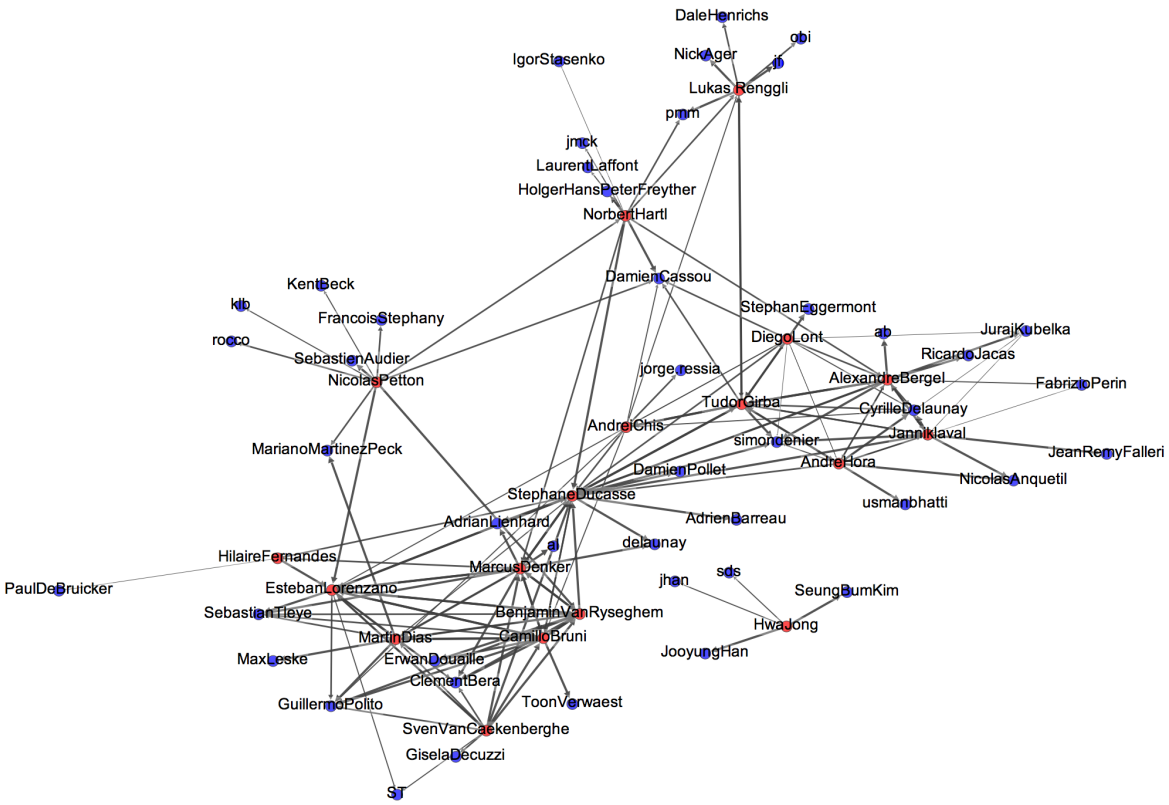

Figure 2: Ground truth collaboration graph with 74 developers and 163 collaboration edges.

2. RELATED WORK

Several existing approaches investigated the relationship between developers and software repositories, assuming these are faithful sources for models of collaboration and trying to mine and/or visualize information from them.

Heller *et al.* [4] applied visualization techniques to user profiles and repository metadata from GitHub. In particular, they produced directed graphs in which nodes represent users geographic locations and edges represent a) follower relationships, b) successive commits, or c) contributions to the same project. Such visualizations are used to express hypotheses about GitHub community.

De Souza *et al.* [2] presented a visualization approach to analyze open-source software projects and relate the complexity of development process with the complexity of artifacts under development. Among all the visualizations, they also represent the relationship between project members who have contributed to the same modules as a graph.

Jermakovics *et al.* [5, 6] proposed an approach to discover collaboration networks of open source developers from Version Control Systems (VCS). The approach 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. The approach has been validated in case studies of three different open-source projects (*e.g.,* phpMyAdmin) to learn their organizational structure and patterns. Results indicate that with little effort the approach is capable of revealing aspects of those projects that were previously not known or would require effort to discover manually via other means, such as reading project documentation or forums.

The related work most similar to ours is the one by Meneely and Williams [10], who studied if developer networks can be corroborated with what a developer perceived as a collaboration. A developer network is a graph, built from version control change logs, where vertices represent a developer on the team; edges exist where two developers made a version control commit to the same source code file within one month of each other. To measure developer perceptions, they conducted an online survey, personalized to each developer of a team, based on that developer's Social Network Analysis (SNA) metrics. Developers answered questions about other members of the team, such as identifying their collaborators and the project experts. While the outcomes of the analysis found some discrepancies between the developer network and the survey, the paper concludes that developer networks (as built from version control systems) are supported by developer perceptions with statistical significance. They did not analyze super-repositories, but selected project repositories.

In the following sections, we will try to replicate these findings by conducting a similar analysis of developer collaboration in the SmalltalkHub super-repository. We first establish a ground truth through a developer survey, and then we compare it against different models of collaboration built from the SmalltalkHub data.

3. ESTABLISHING A GROUND TRUTH

To assess the quality of any approach that tries to model the concept of collaboration among developers, one needs to establish a ground truth which allows approaches to be compared with. We argue, along the lines of Meneely and Williams [10], that establishing a reliable ground truth in this context should be done by asking the developers themselves.

Our focus is the Pharo open-source development community which uses SmalltalkHub as project management super-repository, featuring at the time of writing 1,033 developers and 1,212 projects consisting of 111,351 packages.

We surveyed 25 developers of which 18 responded to our survey. The survey consisted in having each developer indicate for 10 other developers their level of agreement to the statement "I know and have collaborated (by contributing source code) with the following people", providing a score on a 5-points Likert [11] scale: 1 (strongly disagree), 2 (partially disagree), 3 (neutral), 4 (partially agree), and 5 (strongly agree). We processed the data by creating a graph where nodes represent developers and weighted, directed edges represent the strength of the collaboration between two developers. The graph contained 74 nodes and 180 edges with the following distribution of weights: 17 with weight 0 (strongly disagree), 10 with weight 1, 25 with weight 2, 43 with weight 3, 85 with weight 4. The total weight of the edges is 529. Figure 2 depicts the graph after eliding standalone nodes and edges with weight 0. Red vertices represent surveyed developers, while the blue ones represent developers who only appeared as subjects of the survey.

4. MODELING COLLABORATION

The graph described in the previous section (Figure 2) was obtained by mining a complete SmalltalkHub snapshot taken at 2:02 AM of the 8th of December 2013. The 11.5 GB MongoDB dump in question features 1,048 projects, 102,407 packages, and 940 developers.

4.1 Dataset Creation

The first thing that strikes the eye is that there is a problem of multiple identities, *i.e.,* a developer can have several user accounts. We tackled this aliasing problem by first running an algorithm based on the Levensthein distance [8] to detect similar user names, which we then collapsed. Moreover, we manually analyzed the remaining user names and discovered several other cases of multiple identities, which we also collapsed. The resulting dataset thus contains 832 actual users. There is no way to guarantee that we removed all double identities, which poses a first threat to this kind of work.

4.2 Collaboration Mining Approaches

We present diverse approaches to mine collaborations from our dataset by selecting only developers who are present in the ground truth collaboration graph depicted in Figure 2. Each approach thus produces a graph which contains at most the same number of developers but with different connections and different weights, based on the principles behind each approach. For each approach the weights are normalized in the range 0-4 using quartile distribution, where 0 means absent edge (no collaboration) and 4 means "very strong collaboration".

We are modelling collaboration between developers as a graph, and to associate a quality measure to each approach we will use the most common measure of difference between graphs: graph edit distance (GED). In our case it boils down to summing up, for each edge, the absolute value of the weight difference to the corresponding edge contained in the ground truth graph.

For each approach we describe the underlying key idea and report the result in terms of nodes and edges present in the produced graph and the GED to the ground truth graph.

4.2.1 Number of Common Projects

Key Idea: This approach counts the number of project repositories to which two developers have both committed source code to.

Result: 68 nodes, 174 edges, GED: 201.

Reflections: The main drawback of this approach is that it ignores *when* developers committed to the same project. In principle the developers' work timeliness might have never crossed. Also, it ignores the extent to which a developer contributes to a project, *e.g.,* 1 commit has the same weight as 1,000 commits.

4.2.2 Number of Common Versions

Key Idea: This approach, similar to the one presented by de Souze *et al.* [2], sums up for each project the minimum number of versions contributed by two developers to the same project repository.

Result: 68 nodes, 174 edges, GED: 165.

Reflections: This approach refines the previous one by quantifying the extent of collaboration *per project*. This approach still has the drawback of ignoring *when* developers committed to the same project.

4.2.3 Number of Alternating Versions

Key Idea: This approach considers the timeline consisting of versions, for a certain project, committed by two given developers. This approach quantifies the collaboration between them by computing the number of versions that have a preceding version by the other author. In essence it tries to model the fact that the "paths" of developers have crossed. As in the previous approach, we sum up collaboration values for each project to obtain the final result.

Result: 68 nodes, 174 edges, GED: 171.

Reflections: This approach introduces a simple implicit timing constraint that ensures that versions considered for collaboration are interleaved. This approach still ignores *explicit* timing constraints.

4.2.4 Number of Alternating Versions in a Time Frame

Key Idea: This approach computes collaboration as the previous one, but considering only versions committed in a time frame of 30 days.

Result: 62 nodes, 160 edges, GED: 189.

Reflections: This approach introduces a time frame constraint that excludes interleaved versions that are too distant in time. We adopted the span of the time frame of the approach by Meneely and Williams [10]; however, this time frame could be in principle adapted to the customs of a specific development community.

4.3 Reflections

The results of comparing the different collaboration mining approaches with respect to the ground truth in terms of the GED show significant difference. In the best case, the GED amounts to 31% of the total weight of the ground truth graph edges. We conclude that data mined from software repositories is likely to *not represent* the actual collaboration that is happening among developers. This also means that any research that uses version control systems as a source for collaboration studies must take into account this discrepancy.

5. DISCUSSION

In the previous section we discussed how all the different models of collaborations constructed from repository data are different from the ground truth, in the sense that the graph edit distance is significantly high. This finding conflicts with the results of Meneely and Williams [10], that essentially stated that developer networks matched the ground truth obtained by surveying developers. To understand the reasons behind this discrepancy, a fundamental aspect to check is that the ground truth is constructed in the same way.

By inspecting the survey made by the authors, we found that the way they measured collaboration to build the ground truth is inverted

with respect to us. In fact, the survey contained the following answer options for the question "In the context of (name of project), what is your connection to the following people?":

A: I have never heard of this person before;

B: I recognize this name, but I don't know much about them;

C: I know who this person is, but I have not worked with them directly;

D: I have worked with this person on this project.

The only answer that states that developers collaborate is option D, flattening collaboration to a boolean value. Instead, in our survey, we had only one option that characterizes absence of collaboration, while the four remaining ones represented increasing levels of collaboration.

To compare our findings with the results of Meneely and Williams, we adjusted our model by collapsing all the degrees of collaboration into one, and thus representing collaboration as either present or absent. As a result the total weight of the ground truth graph became the same as the number of its edges, *i.e.,* 174. If we consider the simplest approach described in Section 4.2.1 (number of common projects) then the GED against the ground truth model is 11, therefore close to it. This value is considerably low and it matches the results of Meneely and Williams.

This shows that collaboration model mined from the software repository is matching the actual situation only if we consider collaboration as a binary option that either exists or not. But when we try to refine the collaboration model, and express different degrees of collaboration, the data that we mine significantly differs from the ground truth.

No Hope for Social Network Analysis? One might wonder whether this type of research is intrinsically flawed. We believe it is not, but that only looking at the versioning system is probably too limited. To mine collaboration one should also factor inside other sources of information, such as mailing lists and bug tracking systems [3], where another major part of social interactions take place.

6. THREATS TO VALIDITY

Construct validity threats are concerned with whether what one measures is what one intends to measure. In this case they mainly come from the way the survey was performed, *i.e.,* we cannot be certain that developers filled out the survey in a truthful way. We tried to mitigate this threat by manually verifying parts of the ground truth graph, as two of the authors of this paper are part of the Pharo community and are therefore aware of a number of collaborations. We could not find any stark contradictions in the ground truth graph.

Statistical conclusion validity threats concern the fact that there is enough data to support claims. Clearly, such an experiment would need to be repeated with a larger set of developers. Moreover, we do not make any claims regarding statistical significance, but limit ourselves to pointing out that there seems to be some limitations to the way one can mine collaboration data. In essence, SNA must be taken with a grain of salt.

External validity threats are concerned with the generalizability of results. We only ran our experiments with one community, albeit a fairly large one, relying on one specific super-repository, SmalltalkHub. To confirm our findings we should repeat our experiment with other communities as well.

7. CONCLUSION

We investigated to which extent the collaboration data mined from software repositories is representing the actual collaboration between developers of an open-source community. Through a survey we established a ground truth about the collaboration of developers of the Pharo open-source community. We ran a number of approaches that tried to mine the collaboration from a repository, to find that there is a significant difference to the ground truth. This suggests that SNA based on the data mined from version control systems are likely to not model real collaboration.

8. ACKNOWLEDGMENTS

We thank the Swiss National Science foundation for the support through SNF Project "ESSENTIALS", No. 153129.

9. REFERENCES

[1] C. Bird, A. Bachmann, E. Aune, J. Duffy, A. Bernstein, V. Filkov, and P. T. Devanbu. Fair and balanced? bias in bug-fix datasets. In *Proceedings of ESEC/FSE 2009*, pages 121–130, 2009.

[2] C. de Souza, J. Froehlich, and P. Dourish. Seeking the source: Software source code as a social and technical artifact. In *Proceedings of GROUP 2005*, pages 197–206, New York, NY, USA, 2005. ACM.

[3] A. Guzzi, A. Bacchelli, M. Lanza, M. Pinzger, and A. van Deursen. Communication in open source software development mailing lists. In *Proceedings of MSR 2013*, pages 277–286, 2013.

[4] B. Heller, E. Marschner, E. Rosenfeld, and J. Heer. Visualizing collaboration and influence in the open-source software community. In *Proceedings of MSR 2011*, pages 223–226, 2011.

[5] A. Jermakovics, A. Sillitti, and G. Succi. Mining and visualizing developer networks from version control systems. In *Proceedings of CHASE 2011*, pages 24–31, New York, NY, USA, 2011. ACM.

[6] A. Jermakovics, A. Sillitti, and G. Succi. Exploring collaboration networks in open-source projects. In *Open Source Software: Quality Verification*, pages 97–108. Springer, 2013.

[7] A. Lamkanfi, J. Pérez, and S. Demeyer. The eclipse and mozilla defect tracking dataset: a genuine dataset for mining bug information. In *Proceedings of MSR 2013*, pages 203–206, 2013.

[8] V. I. Levenshtein. Binary codes capable of correcting deletions, insertions, and reversals. *Cybernetics and Control Theory*, (10):707–710, 1966.

[9] M. Lungu, M. Lanza, T. Gîrba, and R. Heeck. Reverse engineering super-repositories. In *Proceedings of WRCE 2007*, pages 120–129. IEEE CS Press, 2007.

[10] A. Meneely and L. Williams. Socio-technical developer networks: Should we trust our measurements? In *Proceedings of ICSE 2011*, pages 281–290, New York, NY, USA, 2011. ACM.

[11] A. N. Oppenheim. *Questionnaire Design, Interviewing and Attitude Measurement*. Pinter, London, 1992.

[12] E. Raymond. *The Cathedral and the Bazaar - Musings on Linux and Open Source by an Accidental Revolutionary*. O'Reilly, 1999.