The Importance of Social Network Structure in the Open Source Software Developer Community

Matthew Van Antwerp
Department of Computer Science and Engineering
University of Notre Dame
Notre Dame, IN 46556
mvanantw@cse.nd.edu

Greg Madey
Department of Computer Science and Engineering
University of Notre Dame
Notre Dame, IN 46556
gmadey@cse.nd.edu

Abstract

This paper outlines the motivations and methods for analyzing the developer network of open source software (OSS) projects. Previous work done by Hinds [5] suggested social network structure was instrumental towards the success of an OSS project, as measured by activity and output. The follow-up paper by Hinds [4] discovered that his hypotheses, based on social network theory and previous research on the importance of subgroup connectedness, were vastly different than the results of his study of over 100 successful OSS projects. He concluded that the social network structure had no significant effect on project success. We outline how his approach disregarded potentially important factors and through a new study evaluate the role of the OSS developer network as it pertains to long-term project popularity. We also present an initial investigation into the adequacy of using the SourceForge activity percentile as a long-term success metric. In contrast with Hinds, we show that previously existing developer-developer ties are an indicator of past and future project popularity.

1. Introduction

This paper presents some shortcomings of previous work evaluating the role of network structure in OSS project success. Community ties in the developer network are thought to be instrumental for project success [7]. However, it remains unclear how these ties affect project success or how they help mold future developer communities. Such knowledge may help bring together project contributors sooner than they would have and may help identify unfruitful ties. Hinds and Vreugdenhil concluded that the developer network structure has no significant effect on the success of a project [4, 10], however they both looked at static or conflated networks, ignoring structural evolution. Repeated developer-developer links generally indicate a previously successful collaboration as well as a high likelihood of another successful collaboration (with success measured by long-term popularity).

2. Problem Statement

While the static network may not have structure that predicts success or failure [4], how the network got to that state may be important and may be a better predictor of long-term project popularity. Hinds' study had a project window starting with a project's first software release. At that point, many of the community ties already exist. Open source software projects don't exist in vacuums. Many factors affect how project communities are formed and previous collaboration may be an important one. In particular, we looked at the incidence of repeated developer ties and how the occurrence of such ties affected the activity percentile for those SourceForge projects.

3. Method

Using data from the SourceForge Research Data Archive [2, 9] and the new dataset of concurrent versions system (CVS) metadata described in [8], we analyzed how previous network ties affect future developer communities. First, the percentage of ties that existed on previous projects was identified. Then, we determined which projects have long-term popularity and identified whether or not previously-existing developer community ties increase the odds of producing such projects. To measure long-term popularity, we used the SourceForge activity percentile. Activity percentile will be high if the project is popular since it measures recent traffic, development, and communication. If the project is not popular in the long run, this will fall down towards zero as development and downloads diminish. Activity percentile was taken from a month after the last developer-developer tie was formed from the CVS dataset.

The OSS network is defined as follows. Developers and projects are considered nodes in the graph. If a developer works on a project, there is an edge between the developer and that project. Since developers can only work on projects, the resulting graph will be bipartite, with developers and projects being the two groups having no edges within those groups. The developer network can be created from the aforementioned bipartite graph. For each developer, create a node and create an edge between two developers if there is an edge from each of them to any one project in the bipartite graph. A tie was only created between them if they worked on the project at the same time, i.e. their time frame windows overlapped. Using this method, an unweighted graph can be created.

3.1 SRDA Database

This database, located at http://srda.cse.nd.edu is available for scholarly research to registered users. It

IEEE computer society

contains over 4 straight years of monthly data dumps which are snapshots of SourceForge's back-end database. Data available there includes all the descriptive information on users and projects (or groups) as well as the activity percentile used to measure long-term popularity.

3.2 CVS Archive

The CVS archive is a database containing all of the information in the publicly-available CVS logs hosted by SourceForge. Their servers were spidered and all log files were downloaded, parsed, and stored in a PostgreSQL database. An entity-relation diagram is provided in figure 1. The relevant data here are the developer-group links, which form a bipartite network. From this starting point, we can calculate the developer-developer links. An added benefit to this database is that during the parsing process, all timestamps were converted to unix timestamps, a simple integer. While extracting developer-group links from the database, we also extracted the oldest and youngest timestamps for that developer on that particular project. We made the reasonable assumption that the developer was working on the project between these two times, while we cannot be sure of their contributions to the project outside of that time window.

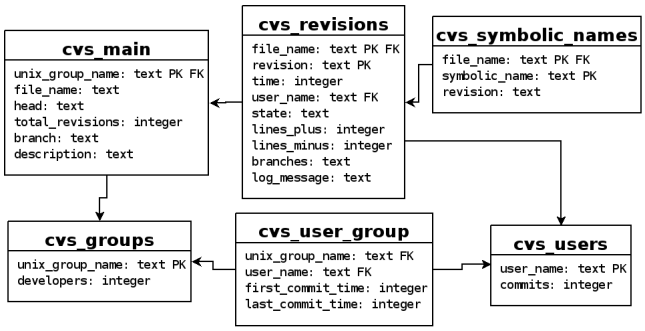


Figure 1: Entity-relation diagram of the CVS database.

3.3 Popularity Measurement Limitations

The issue of measuring OSS "success" has been analyzed in many papers. While success metrics are important, they are often subjective and qualitative, making them unwieldy for studying more than a handful of projects. Instead of using a success metric or group of success metrics, we analyzed the developer network in terms of the SourceForge activity percentile, explained further in section 7. This data was taken from the March 2008 SRDA snapshot, since the CVS data was complete through February 2008.

4. Hypotheses

Hypothesis 1: *Popular projects are more likely to contain ties that existed on previous projects than the average project.* Incidence of previous ties will be relatively straightforward to calculate for the average project (or a random sampling of sufficient size). For choosing popular projects, this will be more difficult. However, SourceForge provides an activity percentile based on many different metrics. Projects with high activity percentile are popular projects since it is based on downloads, site views, development activity, and administrator activity.

Hypothesis 2: *Ties made on previous projects are more likely to lead to popular projects than projects with no previous developer ties.* This is roughly the converse of hypothesis 1.

5. Results

Data extracted from the developer-developer network and activity percentiles for the projects on which those ties were formed supports these hypotheses. Popular projects are more likely to contain previously existing developer-developer ties than the average project. Also, ties made on previous projects are more likely to lead to popular projects than the average project. These results are shown graphically in this section.

5.1 Developer Network Statistics

The developer network had 73,828 developers. Ignoring overlapping time windows (developers connected if they ever worked on the same project, regardless of when they worked on it), 53,085 of those were connected to at least one other developer, leaving 20,743 isolated developers (meaning they work on one or more project as sole developer). The largest connected component contains 24,827 developers, only 33.63% of all developers. There were 1,429,527 developer-developer links in the network. When we heed overlapping time windows (developers only connected if they worked on the same project at the same time, yielding more accurate tie information), the number of developer-developer links shrinks to 396,590. Of these, only 10,491 are duplicates or the original links that were later duplicated. 4,857 developer-developer links were repeated at least once. One pair of developers have worked on 10 projects together. Unsuprisingly, one of those users has a total of 99 repeated developer links, i.e. 99 times that developer collaborated on a project with a developer with whom he previously collaborated. Those ties are with 42 unique developers. The number of duplicate links (developer-developer links that previously existed) is 5,634, which is 1.42% of all developer-developer links. These developer links are of particular interest since the developers decided to work together again on another project.

Figure 2 contains a graph of the distribution of activity percentile for all projects. This is normalized by SourceForge's equation so that the values are between 0 and 100. This metric used to be perfectly evenly distributed (figure 2 would look like a rectangle) but since the middle of 2007 that is no longer the case, most likely due to the purging of inactive projects. Figure 3 contains the distribution for projects that have at least two developers. This is a more accurate baseline for comparison since developer-developer links are only possible on projects that have at least two developers. Figure 4 contains the distribution for projects that contain either repeat links or links that were repeated on a project at a later time. As in hypotheses 1 and 2, these projects tend to have higher activity percentile than the typical project that has at least two developers (as well as far more activity than the typical project).

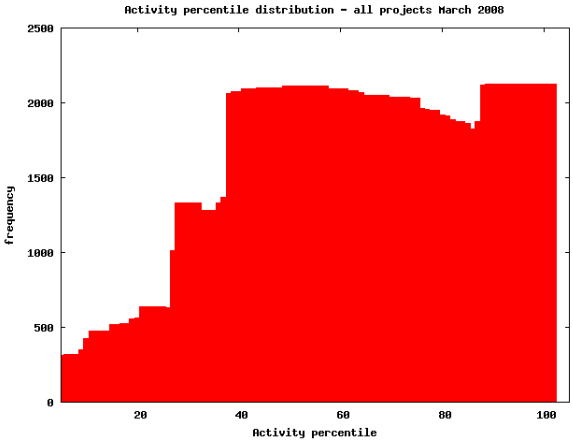


Figure 2: Distribution of activity percentile. March 2008 data used.

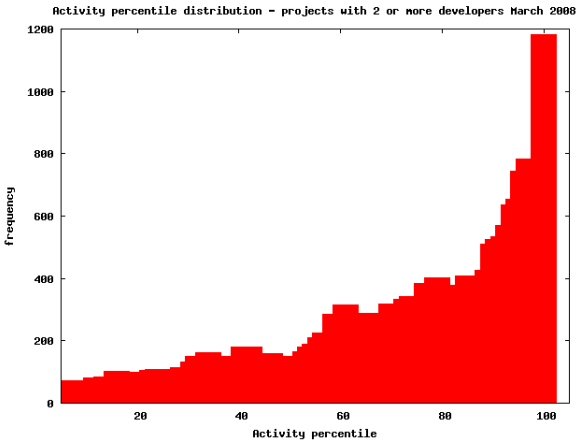


Figure 3: Distribution of activity percentile for projects with at least 2 developers. March 2008 data used.

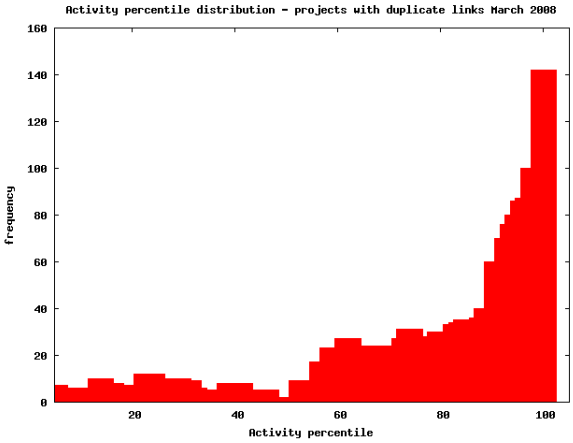


Figure 4: Distribution of activity percentile for projects with repeat links or links that would later be repeated. March 2008 data used.

Figure 5 contains the distribution of activity percentile of projects with developer-developer links that would later be repeated (the first collaboration only, regardless of how many times it was repeated). Figure 6 contains the distribution of activity percentile of projects with developer-developer links that are repeated (contains all repeat collaborations for each developer-developer pair).

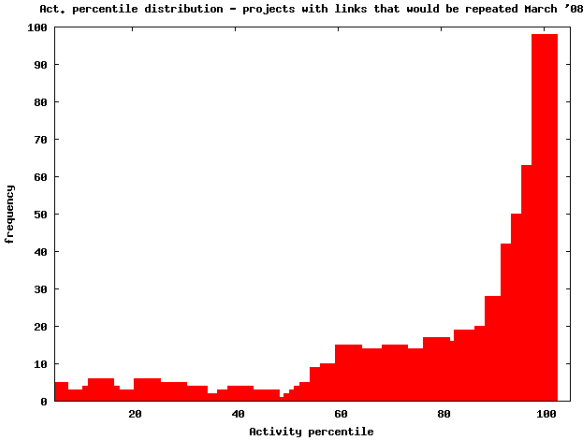


Figure 5: Distribution of activity percentile for projects with links that would later be repeated. March 2008 data used.

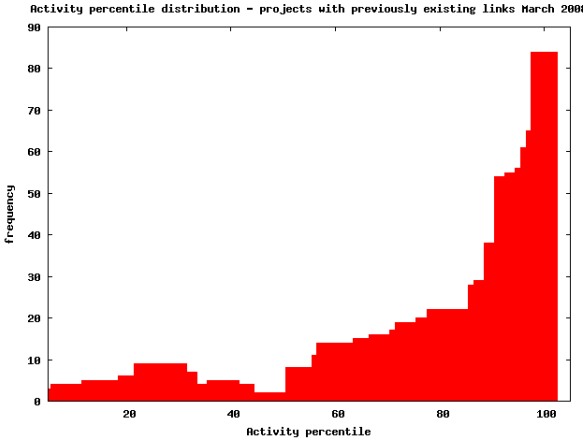


Figure 6: Distribution of activity percentile for projects with links that are repeated. March 2008 data used.

6. Activity Percentile

Since activity percentile is the success metric used for this study, an investigation into long-term trends is provided here to support our assumption that it is an adequate success metric. We would like to investigate these trends to see if, over a long enough span, projects separate into distinct groups. Indeed, this is exactly what happens, with a widening gap occurring between the two groups which warrants further investigation. In this section, we present initial study of activity percentile trends over a 50 month span. Using data from SRDA, we obtained percentile for all

projects that were alive in April 2005, the first month the `stats_group_rank` table had `percentile` as an attribute. At that time, there were 129,468 projects with `group_id`s appearing in that table. Since activity percentile is normalized, it is evenly distributed from 0 to 100. A graph of this appears in figure 7.

Below is the calculation for activity percentile. It is based equally on traffic, development, and communication (it is the sum of those three calculations) and is normalized by the highest project totals.

Traffic: (

(log (prior 7 days download total + 1) / log (highest all-project download total + 1))

+ (log (prior 7 days logo hits total + 1) / log (highest all-project logo hits + 1))

+ (log (prior 7 days site hits total + 1) / log (highest all-project site hits + 1))

) / 3

Development:

(

(log (prior 7 days cvs commit total + 1) / log (highest all-project total + 1))

+ ((100-age of latest file release (in days, max 100)) / 100)

+ ((100-days since last project administrator login (max 100)) / 100)

) / 3

Communication:

(

(log (prior 7 days Tracker submission count + 1) / log (highest all-project total + 1))

+ (log (prior 7 days ML post count + 1) / log (highest all-project total + 1))

+ (log (prior 7 days Forum post count + 1) / log (highest all-project total + 1))

) / 3

The Traffic component is only considered for projects with File Releases. The Communication component is only considered for projects that have categorized themselves within the Software Map (Trove categorization). Tools that use an all-time ranking instead of a weekly ranking, such as search results, will use an aggregate tool item count rather the data from the past 7 days, as displayed above. total = traffic + development + communication This was formerly published on SourceForge's website at http://alexandria.wiki.sourceforge.net/Statistics but is not available. The above is copied from [10].

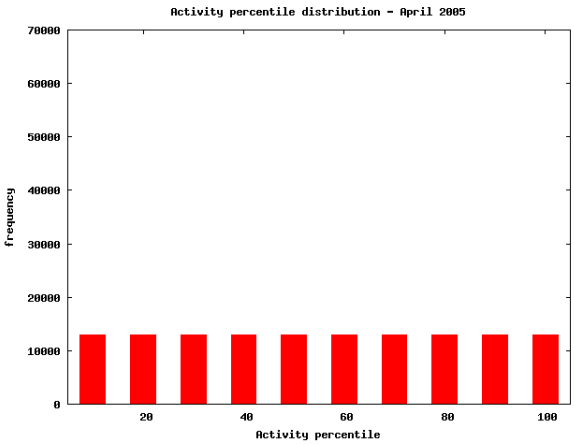


Figure 7: Distribution of activity percentile for projects alive in April 2005, binned. April 2005 activity percentile data used. This is the base line for comparison for figure 8, shown on the same y-axis scale.

Observing only those projects alive in April 2005, it is clear that activity percentile for projects tended to decline over that span. In fact, of the 129,468 projects, nearly 60,000 of them had an activity percentile of zero in May 2009, just over 4 years later. The distribution of May 2009 activity percentiles for projects that were alive in April 2005 is shown in figure 8.

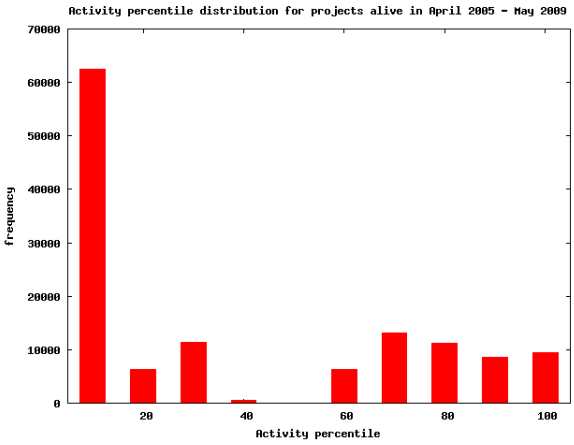


Figure 8: Distribution of activity percentile for projects alive in April 2005, binned. May 2009 activity percentile data used.

Using random sampling, 1000 projects were selected for which 50 months of activity percentile data was obtained. This data was clustered using various clustering methods but no particularly interesting trends emerged that were not already apparent. Figure 9 shows a fairly even distribution in the first graph. Compare that to the second graph which shows activity percentile for those same projects 4 years later. Most projects have tailed off to the bottom portion of the graph. Slightly more than a third of the projects remain in the upper half. There is a clear division between these projects, with a large range separating the two areas without any

projects at all among the sampled projects.

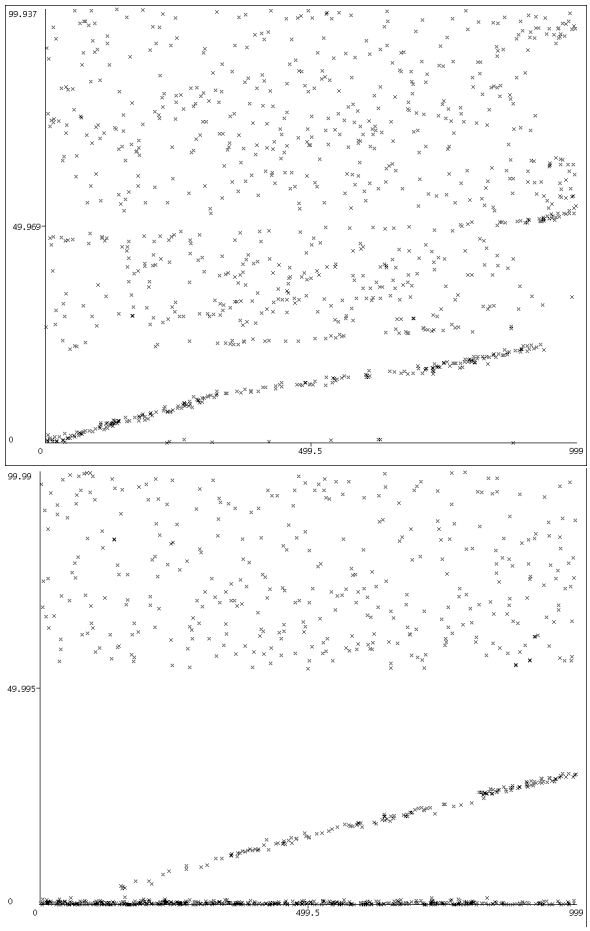


Figure 9: The top graph is the distribution of activity percentile in April 2005 for 1000 randomly sampled projects. The bottom graph is distribution of activity percentile in May 2009 for 1000 randomly sampled projects alive in April 2005. X-axis is an arbitrary instance numbering.

In all random sample data sets tested, there was a noticeable drop in activity percentile for substantial number of projects from May 2007 to June 2007. Most of these projects were in a cluster that started in the bottom portion. This anomaly must be accounted for in analysis. All other month transitions exhibited marginal movement in the activity percentile.

Figures 10, 11, 12, 13, 14, and 15 display the monthly distributions of activity percentile over a two year span for all projects that appeared in the June 2007 (sf0607) schema. Schema names are included in the graph titles following the convention `sfMMYY` where `MM` is the month and `YY` is the year. After about a year, a divide begins to show at around the 50 percent mark. Gradually this gap deepends until no projects appear in a specific range, and then that range becomes wider. In the May 2009 data set, no project that was alive in June 2007 appears between 43.0546840967 and 54.5382245153, a gap of over 10 percentile points. A division this clear and pronounced was not expected. Projects fall into two main groups where the bottom half tends to have activity percentile trail off over time. The second group is the upper half which tends to hold fairly steady. It is not clear why not even a single project from the second group falls down into the gap between these two groups. This gap warrants further investigation.

The results presented in this section show that in the long term, activity percentile is an adequate measure of success, since projects tend to fall into one of two distinct groups, one indicating success, the other being the group of less successful projects that are headed towards inactivity. It also shows that after two years, the gap between projects is apparent and significant.

7. Limitations

It is important to emphasize that the developers here are strictly limited to those who have made CVS commits and furthermore, only includes data from SourceForge. These are the only developers visible through the CVS logs. Projects have a community that is very important for project development but are ignored here due to the data source. People who actually use the software produced by the developers can request features, submit bugs, submit bug fixes, and provide other feedback about the project on message boards and mailing lists. While these users are crucial for project success, they are not a part of this study because the primary data source is the collection of CVS logs. A more complete study would include other SCMs, such as Git and SVN. A factor that may influence the activity percentile distribution is the experience level of developers. In the repeated ties data, this may be an influence, however even for the distribution of original ties that were later repeated, it was far more likely to result in a successful project than the typical project. The monthly granularity of activity percentile data (due to how often we receive data dumps for SRDA) is another small limitation that should be noted.

Another limitation with using activity percentile is that we took the measurement (for all projects) from a month after the last developer-developer tie was formed. This was for the sake of consistency as well as the difficulty in deciding when to measure the long-term popularity of a project. Collaboration lengths and project age vary greatly. Determining when to measure popularity or success in relation to developer tie formation and project age is a problem that warrants investigation. For example, it would be useful to know how long a project must be active for it to have a high probability of long-term popularity. The previous section contains initial investigation on the topic of activity percentile trends.

8. Previous Work

While much work has been done to examine the social structure of OSS projects, very little of it has been focused on how the structure affects success and how the evolution of the structure affects success, which is the focus of this paper. Hinds hypothesized that the cost of ties between subgroups would outweigh the benefits as the number of ties rose (implying a negative-U shape). He also suggested that certain activity and output metrics would only trend negatively as the number of ties rose between two particular groups [5]. In his follow up paper [4], he discovered that many of his hypotheses

were incorrect and concluded that "social network structure of an open source software project community has no important effect on community success." He went on to outline several possible reasons for the seemingly counterintuitive results of his analysis. In [10], Vreugdenhil largely mirrored the approach and analysis of Hinds, but on a smaller slice of data (one month), while Hinds analyzed projects over a two year timespan. His conclusions were the same as Hinds'. Ngamkajornwiwat, et al examined OSS developer network evolution [6]. They discovered OSS developer network evolution patterns in KOffice. The work done in this paper is towards the same high-level goal of discovering the role social network evolution plays in OSS project success. Hahn, et al studied social ties and how developers joined projects [3]. Hahn determined how people join projects based on their previous social ties. Crowston and Howison examined how the social structure affected communication patterns for bug fixes in various OSS projects [1].

9. Future Work

Developer-developer connections are binary for the purposes of this study. However, they can be weighted with many possible values. The amount of overlap in their respective time-windows on the project at hand, the number of commits made by each user, and even more fine-grained analysis such as which files they worked on. Two developers who frequently commit changes to the main project source file indicates a stronger connection than two developers who have never committed changes to the same file on a project but instead work on disjoint parts of the same project. All of this information can be extracted from the CVS database archive. We put forth that long-term project popularity, as measured by the SourceForge activity percentile, is a rough success metric. An initial study of activity percentile trends is included here but further investigation is warranted. Do projects that are successful by more rigorous measurements always have a fairly high activity percentile? Does it fluctuate? Do failed projects always eventually have activity percentile decline? This can fairly easily be studied with SRDA data.

Furthermore, a handful of repeated developer pairs were part of larger groups of repeats. In other words, an entire group worked together again on a different project. On initial inspection, these were projects with a large number of developers and they were closely related projects. For example, a large group of developers worked on both `collective` and `plone-i18n`, the former being a collection of products for use with the Plone CMS and the latter being an effort to internationalize the Plone CMS. Another group of developers worked together on a number of projects, all related to the programming language `tcl`, such as `tclpro`, `tcllib`, and `tktoolkit`, among others.

10. Conclusions

Long term popularity was the metric used to analyze the importance of developer-developer links in the SourceForge CVS developer network. This is because it is an easily obtained metric available for all SourceForge projects as well as a rough indicator of project success. We have displayed the activity percentile trends over multi-year spans and discovered that a large divide develops between the two clusters of projects, clearly visible after 2 years.

Based on activity percentile, previous ties are generally an indicator of past success and usually lead to future success. Intuitively, this makes sense. When a repeated developer-developer link appears, it tells us something about both projects linking the two developers. Generally, the previous project was successful on some level. The two developers likely worked together well. Since they decided to work together again, this means they probably expect the experience to be similar to previous experiences and again form a successful collaboration.

11. Acknowledgments

The authors would like to thank Dr. Nitesh Chawla for early conversations related to this topic.

12. References

[1] K. Crowston and J. Howison. The social structure of free and open source software development. *First Monday*, 10(2), 2005.

[2] Y. Gao, M. Van Antwerp, S. Christley, and G. Madey. A research collaboratory for open source software research. In *FLOSS '07: Proceedings of the First International Workshop on Emerging Trends in FLOSS Research and Development*, page 4, Washington, DC, USA, 2007. IEEE Computer Society.

[3] J. Hahn, J. Y. Moon, and C. Zhang. Impact of social ties on open source project team formation. In E. Damiani, B. Fitzgerald, W. Scacchi, M. Scotto, and G. Succi, editors, *OSS*, volume 203 of *IFIP*, pages 307–317. Springer, 2006.

[4] D. Hinds. *Social Network Structure as a Critical Success Condition for Open Source Software Project Communities*. PhD thesis, Florida International University, 2008.

[5] D. Hinds and R. M. Lee. Social network structure as a critical success condition for virtual communities. In *HICSS '08: Proceedings of the Proceedings of the 41st Annual Hawaii International Conference on System Sciences*, page 323, Washington, DC, USA, 2008. IEEE Computer Society.

[6] K. Ngamkajornwiwat, D. Zhang, A. G. Koru, L. Zhou, and R. Nolker. An exploratory study on the evolution of oss developer communities. In *HICSS '08: Proceedings of the Proceedings of the 41st Annual Hawaii International Conference on System Sciences*, page 305, Washington, DC, USA, 2008. IEEE Computer Society.

[7] E. S. Raymond. *The Cathedral and the Bazaar*. O'Reilly & Associates, Inc., Sebastopol, CA, USA, 1999.

[8] M. Van Antwerp. Studying open source versioning metadata. Master's thesis, University of Notre Dame, Notre Dame, IN, January 2009.

[9] M. Van Antwerp and G. Madey. Advances in the sourceforge research data archive. In *Workshop on Public Data about Software Development*

(WoPDaSD) at The 4th International Conference on Open Source Systems, Milan, Italy, 2008.

[10] B. Vreugdenhil. The influence of social network structure on the chance of success of open source software project communities. Master's thesis, Erasmus University, Rotterdam, The Netherlands, March 2009.

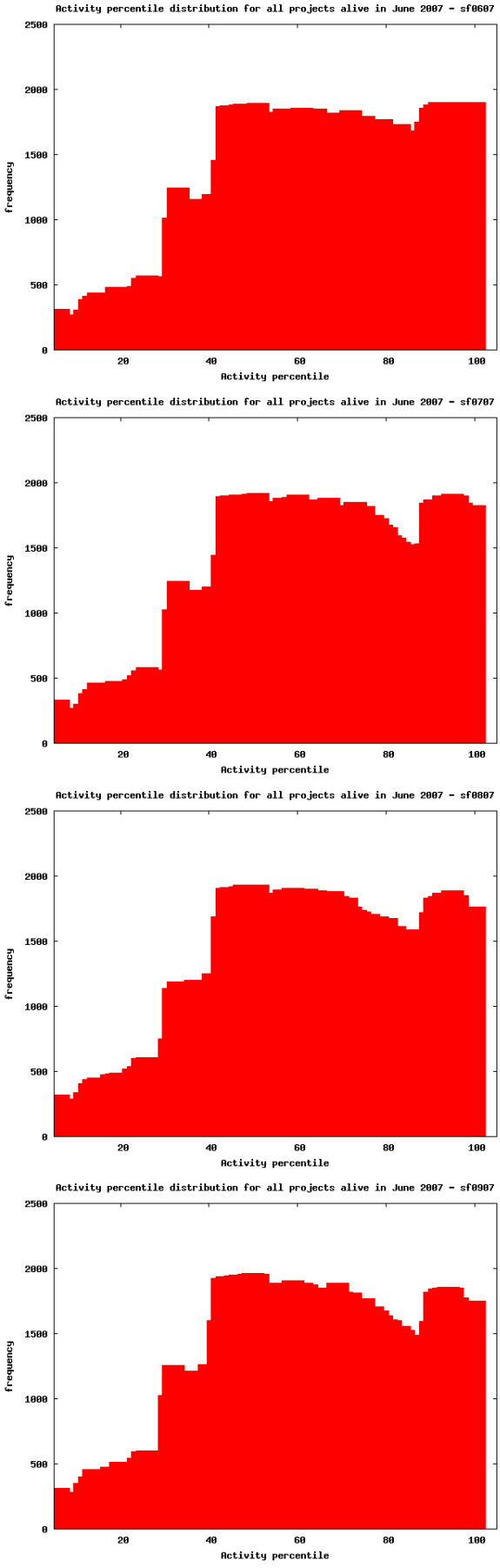
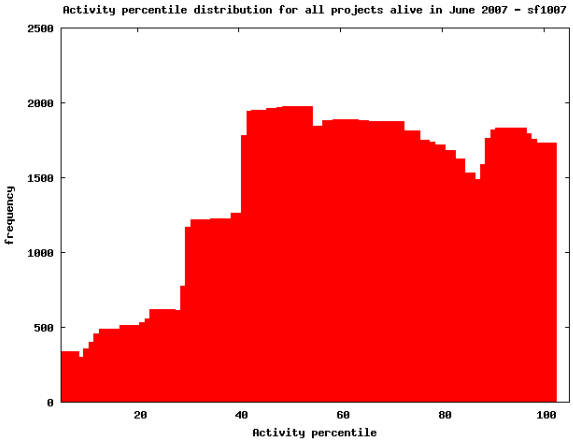
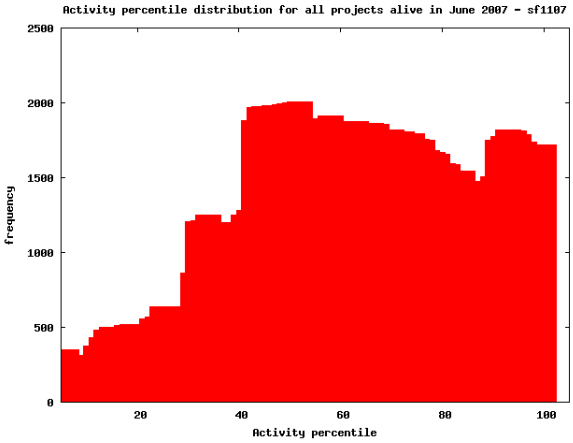
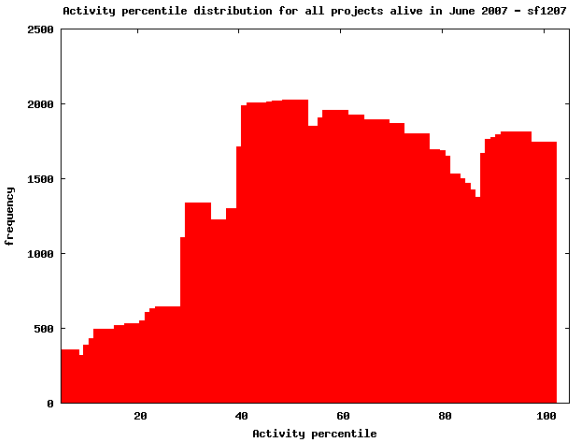


Figure 10: Distribution of activity percentile from June to September 2007.







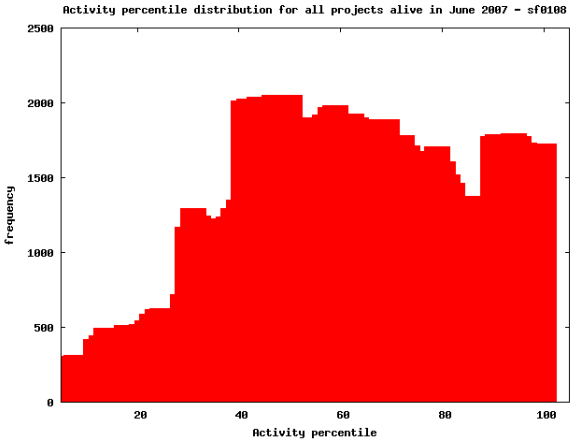
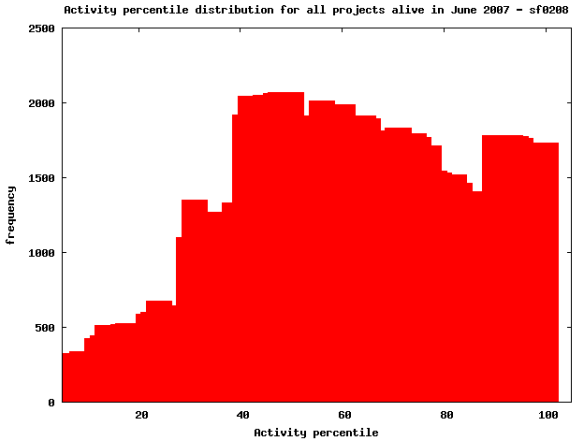
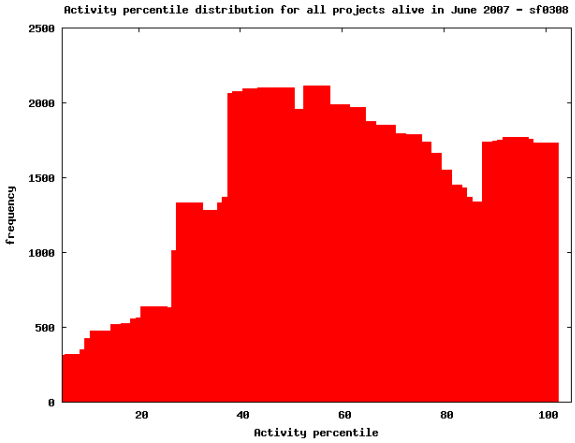
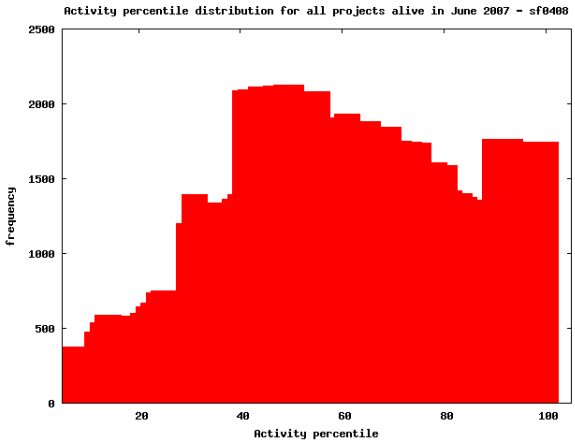


Figure 11: Distribution of activity percentile from October 2007 to January 2008







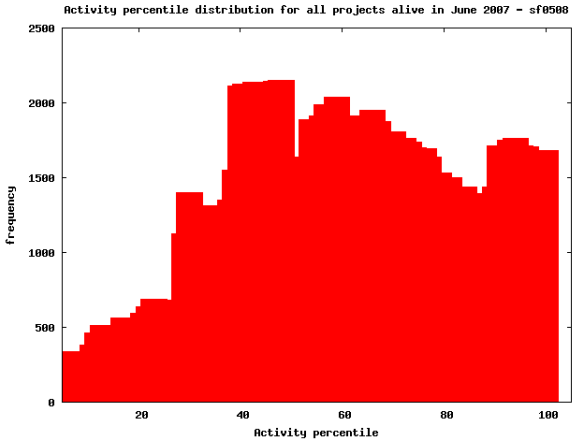
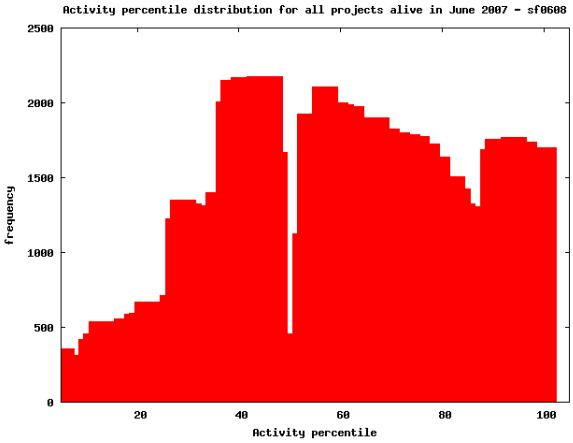
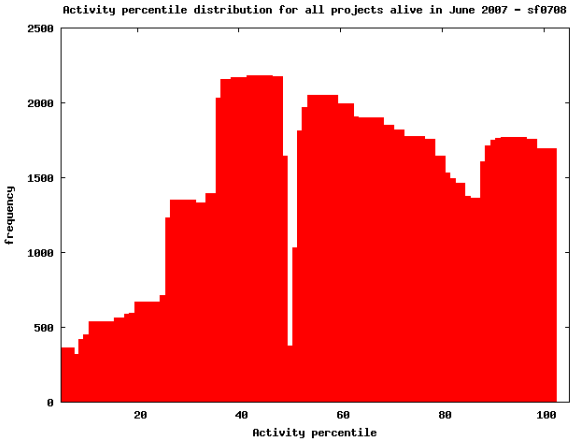
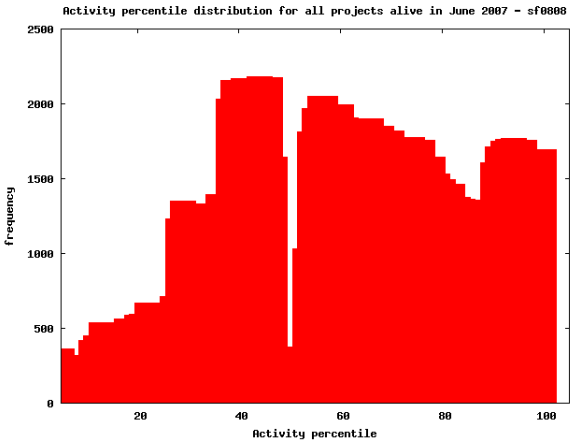


Figure 12: Distribution of activity percentile from February to May 2008







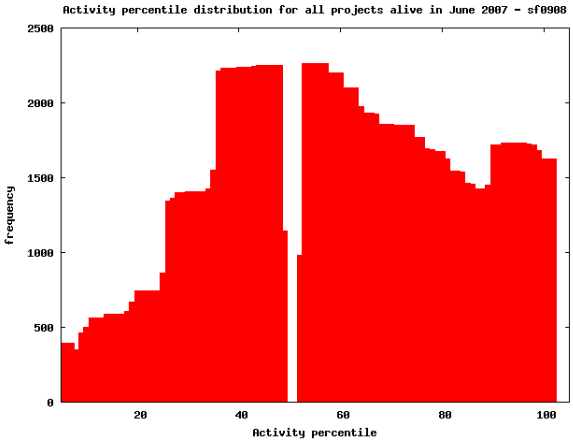
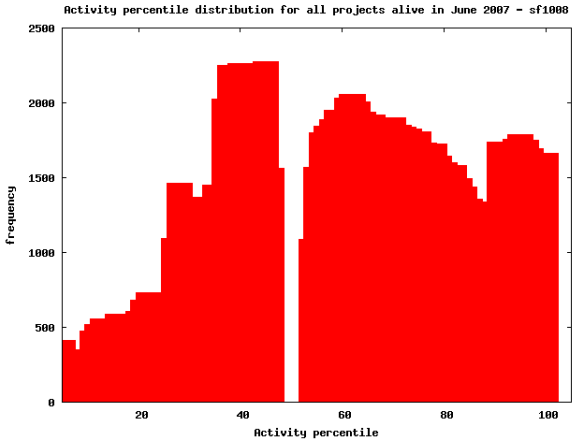
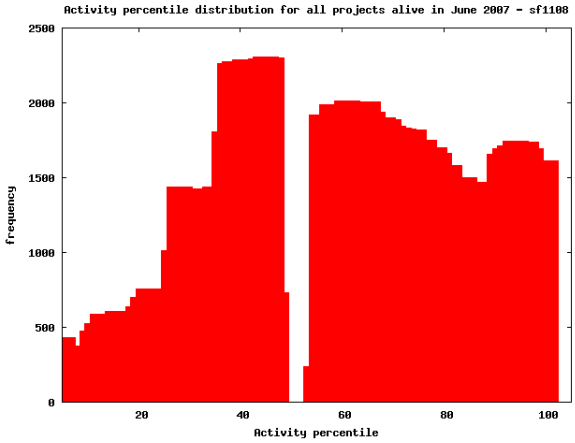
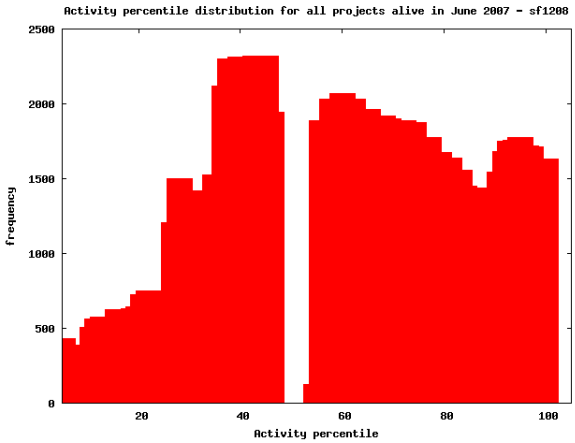


Figure 13: Distribution of activity percentile from June to September 2009







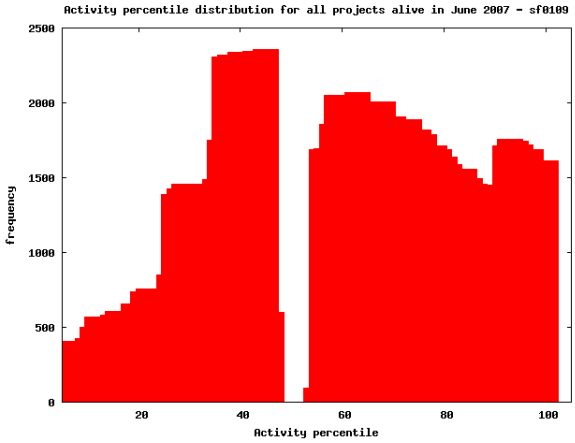


Figure 14: Distribution of activity percentile from October 2008 to January 2009

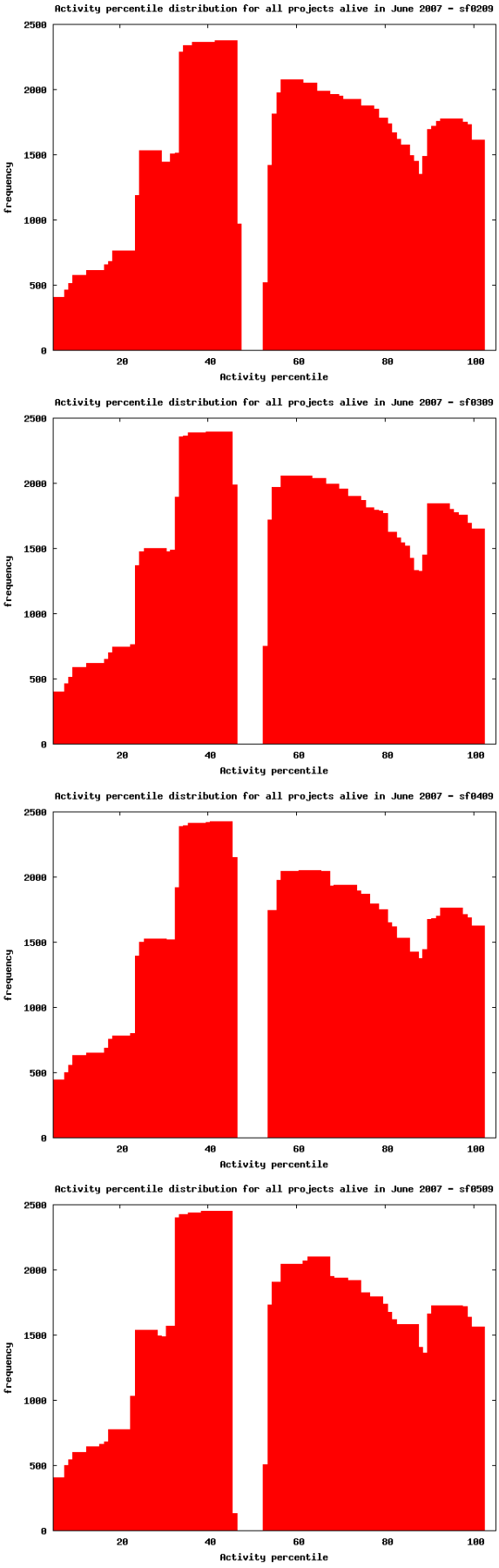


Figure 15: Distribution of activity percentile from February to May 2009