Mining Email Social Networks*

Christian Bird, Alex Gourley,
Prem Devanbu, Michael Gertz
Dept. of Computer Science, Kemper Hall,
University of California, Davis,
Davis, California Republic.
cabird,devanbu@ucdavis.edu

Anand Swaminathan
Graduate School of Management,
University of California, Davis,
Davis, California Republic.
aswaminathan@ucdavis.edu

ABSTRACT

Communication & Co-ordination activities are central to large software projects, but are difficult to observe and study in traditional (closed-source, commercial) settings because of the prevalence of informal, direct communication modes. OSS projects, on the other hand, use the internet as the communication medium, and typically conduct discussions in an open, public manner. As a result, the email archives of OSS projects provide a useful trace of the communication and co-ordination activities of the participants. However, there are various challenges that must be addressed before this data can be effectively mined. Once this is done, we can construct social networks of email correspondents, and begin to address some interesting questions. These include questions relating to participation in the email; the social status of different types of OSS participants; the relationship of email activity and commit activity (in the CVS repositories) and the relationship of social status with commit activity. In this paper, we begin with a discussion of our infrastructure and then discuss our approach to mining the email archives; and finally we present some preliminary results from our data analysis.

Categories and Subject Descriptors

D.2.8 [**Software Engineering**]: Metrics—*Empirical, Open Source*

General Terms

Human Factors, Measurement

Keywords

Open Source, Social Networks

*We gratefully acknowledge support from NSF Humanities and Social Sciences Division, Grant Number SES 0525263.

MSR'06, May 22–23, 2006, Shanghai, China.

1. INTRODUCTION

Large-scale software development projects invariably require a lot of communication and coordination (C&C) amongst the project workers. We distinguish these activities from engineering activities, where actual artifacts such as source code or documents are modified. The difficulty and intensity of the required coordination effort is quite high; this is often cited as the reason why adding more developers doesn't necessarily speed-up development [4]. C&C activities influence (and are influenced by) the design, structure and evolution of software systems. In traditional, commercial software organization, C&C activities may occur informally, and would be difficult to study. Even if coordination and communication are computer-mediated, the traces of these activities are usually not made public by commercial organizations. Open-source software (OSS) projects on the other hand, inherently conduct all their activities in public, and in fact, this public, open enactment is key to their success [16, 11]. In particular, every open-source project includes one or more public mailing lists wherein project stakeholders can communicate and coordinate their activities. The entire trace of these mailing lists are archived and available for study.

These archives, along with the versioned source code repositories and other on-line artifacts constitute a unique and valuable resource for the study of C&C activities in software projects. There is at UC Davis an interdisciplinary effort to mine this resource, and use the resulting data to study the relationship with C&C activities in OSS projects, and the actual development activities. In this paper, we describe our experiences with this effort, and some early results. We begin first with a description of the phenomena that we are mining; then we describe our data extraction tools; finally, we present an early look at the data.

2. CHATTERERS & CHANGERS

A mailing list in an OSS project is a public forum. Anyone can post messages to the list. Posted messages are visible to all the mailing list subscribers. Posters to mailing lists include developers, bug-reporters, contributors (who submit patches, but don't have commit privileges) and ordinary users. Mailing lists can be quite active; for example, on the Apache developer mailing list, there were about 4996 messages in the year 2004 and 2340 in 2005. For gcc, these numbers were 19173 and 15082. Over the lifetime of the project, we estimate that over 2000 distinct individuals have sent messages to the Apache developer list. A subscriber may

respond to a message on the public forum, which then becomes visible to everyone. Roughly 73% of messages elicit response messages. A response b to a message a is an indication that the sender of $b, (s_b)$ found that the sender of $a, (s_a)$ had something interesting to say; thus the response from s_b indicates that the original message a represented information flowing from s_a to s_b It is also an indication of status, *i.e.,* s_b indicates that s/he found s_a's email worth reading, and worthy of response.

The level of activity of developers on the mailing list varies dramatically. The most active developer on the mailing list sent 4486 messages during the life of the project. The least "chatty" developer sent just 10 messages. There were, of course, non-developers who sent just one message. Messages reflect communication interactions between developers. Some developers have a great many interactions: one developer's emails had responses from 254 distinct individuals. Likewise, another developer replied to messages from 281 distinct individuals. However, the vast majority of individuals particpating on the email list sent very few messages, and received very few replies to their messages. This type of "Pareto" distribution is common in social phenomena [14].

The community on the Apache developer mailing list is concerned primarily with software, and so the question naturally arises as how email activity relates with development activity. This activity can be conveniently recovered from the versioned source code repository (CVS in this case). As has been reported in earlier research [8] on Linux, development activity, as recorded in CVS, also shows a few developers doing the bulk of the work.

Our research goal is to study the relationship of the C&C activities of developers, as revealed in the email archives, to their software development activity. Specifically, we are interested in how the activities and connections between developers on the mailing list relate to their development activity in the source code. We are interested in the following types of questions:

- *What are the properties of the social network of developers?*
- *Are developers who send a lot of messages on the mailing list also very active in source code changes?*
- *Do developers play a different role than non-developers in the social network?*
- *Do the most active developers have the highest status among developers ?*

Unfortunately, answering these types of questions requires facing some challenges in data extraction, primarily having to do with resolving aliasing issues on the email archives and cvs archives.

3. OF DOGS AND DEVELOPERS

"On the Internet, no one knows if you're a Dog" —so goes the famous New Yorker Cartoon. It is difficult (and sometimes impossible) to determine the identity of individuals who correspond on mailing lists using aliases. The same individual can use different email aliases. For example the developer *Ian Holsman* uses 7 different email aliases, including `ian.holsman@cnet.com`, `ianh@holsman.net`, and `ianh@apache.org`[1]. Sometimes aliases have very little relationships to developers (or dogs): the developer *Ken Coar* uses the name *Rodent of unusual size* associated with email address `ken.coar@golux.com`. Ignoring these aliases and treating these as distinct email personalities would confound later steps of data analysis. Likewise, when cvs comments are made, developers use a cvs account name. Fortunately, since access and accounts to cvs are controlled centrally, there is less of an aliasing problem with cvs account names. However, in order to relate email activity and programming/development activity, we must correlate email names with cvs account names. Given the possibility of choosing arbitrary aliases, one can make two important observations: first, an individual determined to maintain an anonymous alias can always do so[2] ; second, any automated algorithm for resolving aliases will be inexact, and must be supplemented by subsequent manual analyses.

We now describe our hybrid automated/manual approach to resolving aliases

3.1 Unmasking Aliases

Most emails include a header that identifies the sender, of this form:

```
From: "Bill Stoddard" <reddrum@attglobal.net>
```

This header reveals immediately the problem—*Bill Stoddard*, who here uses the handle `reddrum` is actually also `bill@wstoddard.com`. But how can we know that?

Overview: Our first step in resolving aliases is to automatically crawl messages and extract all such headers to produce a list of $< name, email >$ identifiers (IDs). Once this is done, we execute a clustering algorithm that measures the similarity between every pair of IDs. This could occur if either the names are similar, or if the emails are similar, or if the names and the emails are similar (the precise details of the algorithm are explained below). IDs that are sufficiently similar are placed into the same cluster. Once clusters are formed, they are manuall post-processed.

Apache Summary: In the case of the Apache developer mailing list, we began with 2544 separate IDs. The clustering algorithm produced 1581 clusters. The largest of these had 70 members, the next biggest 55, and so on; finally, there were 163 doubles, and 1271 singletons. Naturally, these clusters contained errors, and had to be manually post-processed. Mindful of the need for manual post-processing, we deliberately set the cluster similarity threshold quite low: it is much easier during a manual step to split clusters than to unify two disparate clusters from a very large set. Manual processing of the 1581 clusters produced 2012 distinct individuals, some of whom have many aliases. One noteworthy example is *Rasmus Lerdorf*, with 11 aliases:

```
rasmus@apache.org,
rasmus@bellglobal.com,
rasmus@lerdorf.ca,
rasmus@lerdorf.com,
rasmus@lerdorf.on.ca,
rasmus@linuxcare.com,
```

[1]Email addresses are used with permission from the mailinglist participants.

[2]The identity of the infamous *"David who wishes to remain anonymous"*, who spammed several email lists, offering to post personal adverts in Ukraine, was not easily found.

rasmus@madhaus.utcs.utoronto.ca,
rasmus@mail1.bellglobal.com,
rasmus@php.net,
rasmus@raleigh.ibm.com,
rasmus@vex.net.

Five of these were discovered using the *Name Similarity* rule (described below); the other six were put into this cluster because of the *Email similarity* rule. *Paul Richards* was another promiscuous email masquerader, with 10 aliases. Subsequent random sampling of the clusters by one of the authors (who didn't do the manual filtering) revealed no discernible errors (see caveat on this later).

Clustering Algorithm: This algorithm takes a flat list of IDs and clusters them (recall that an ID is a $< name, email >$ tuple). The first step is create pairwise similarity measures for every pair of IDs. Two IDs with similarity measure exceeding an empirically set threshold are placed into the same cluster. The similarity measure is computed by proceeding as follows:

1. *Normalize name:* We remove all punctuation, suffixes (*"jr"*); turn all whitespace into a single space; remove generic terms like "admin", "support", from the name; we also split the name (using whitespace and commas as cues) into first name and last name.

2. *Name Similarity:* We use a scoring algorithm based on the Levenshtein edit distance [5, 13, 17] between the full names, and the first and last names separately. We consider names similar if the full names are similar, or if both first and last names are similar. Thus, *Andy Smith* is similar to *Andrew Smith*, but *Deepa Patel* is dissimilar to *Deepa Ratnaswamy*. This is a very productive rule for identifying clusters of similar emails.

3. *Names-email Similarity:* Two IDs are also scored highly similar if the emails and names match. If the email contains both first and last names (and the lengths of the names are at least 2 characters) we consider them matched. Also, if the email contains the initial of one part of the name and entirety of the other part, then it is considered a match. Thus *Erin Bird* matches `erinb` and `ebird`.

4. *Email Similarity:* If the Levenshtein edit distance between two email address bases (not including the domain, after the "@") is small, two emails are considered similar (as long as the two bases at least 3 characters long)

5. *Cumulative ID similarity:* The similarity between two IDs is the maximum of the 3 mentioned above. This generous rule creates larger clusters; however, splitting too-large clusters is easier than unifying smaller clusters (from a very large number of clusters).

The Cumulative ID similarity is computed for all pairs of IDs; IDs with similarity exceeding a threshold are placed into clusters. The clusters are then manually post-processed as described above. The final results produced were hand-inspected by another member of the team, and appears to be free of evident errors. Of course, given the possibility of choosing arbitrary aliases, such manual inspection is fallible. In future work, we will undertake a more formal, sampling-based techniques technique for determining bounds on the error rates in our results. We propose to email a randomly chosen subset of individuals on the list, and ask them if the set of aliases we have found for them is accurate. Assuming that mis-classification errors are uniformly distributed in our clusters, we should be able to calculate confidence intervals on the actual error rate in our clusters.

CVS alias resolution: We use a similar approach to resolving cvs account names to email aliases. Similarity metrics are calculated on all pairs of mailing list aliases and CVS names. The final matched list is hand-inspected, also as described above; the same caveats apply, and in future work, we will use the same random sampling approach to statistically bound the errors in our results.

3.2 Data Extraction

We gathered data by parsing the email activity on the Apache HTTP Server Developer mailing list over a period starting in 1999 to the current date. Earlier email data was not included because we do not have version-control information before then; we only used the email data for the period during which the source code change data was also available. For every email, we extracted from the email header the message identifier, the sender, the sent time, and the identifier of the message (if any) to which this message was a reply. When a reply-to header was found, the sender s of the reply was someone who found the initial message of interest; and so the sender s was marked as a recipient of the original message. In this way, we were able to extract communication links between pairs of individuals.

We were able to parse 101,637 messages out of 102,611 messages in the mailing list. A small proportion of messages could not be parsed, because of malformed headers. Approximately 1.3% of the messages were in this category. Malformed headers can fail to provide a message identifier, and can also fail to provide a reply-to identifier. This could be due to misbehaving email clients. We are working on ways to rectify this problem. Quoted text content (as done in [1]) is one approach, whereby one message is identified as a reply if it quotes text from another. Meanwhile, we believe that our results are reasonably robust, and would not be affected much when these (currently unparseable) messages are included in the analysis.

4. DESCRIPTION: SMALL WORLD

The distributions of the data that are shown in Figure 1 describe the behavior of the participants of the email list. Each is a histogram showing the number of people exhibiting a particular kind of behavior. The character of the distributions is consistent with previously observed social phenomena, and show the typical long-tailed characteristic in the log-log domain plot.

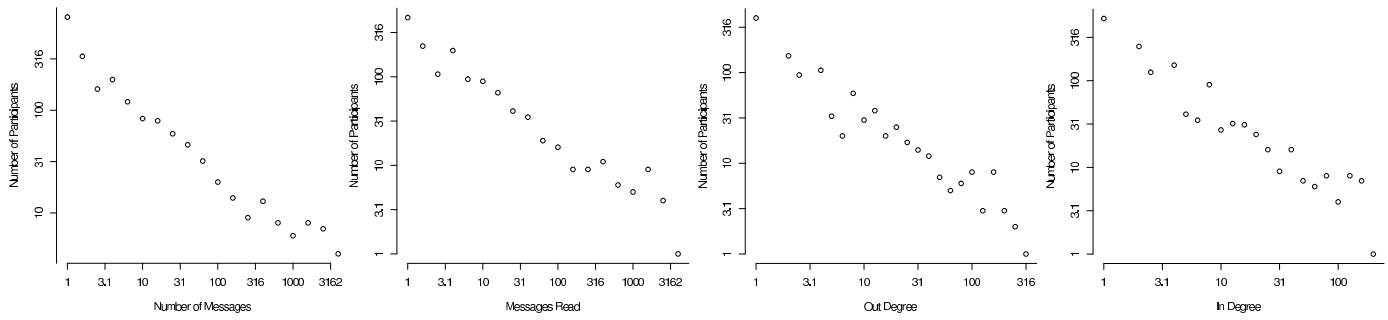

Figure 1: Note that all diagrams are log-log scale. Reading left to right: first, the distribution of people vs. number of messages they sent; next, vs. the number of reply messages they received. Note that a few people account for the bulk of the sending & reply activity. The next two indicate the structure of the social network. First, the out-degree in the social network; finally, vs. the in-degree in the social network. Out degree is an indication of status, as it indicates the number of different people who replied to the ego's messages. In-degree indicates the number of different people whose messages ego responded to. All distributions show power-law character. The degree distributions show small-world character of the email social network.

The first shows a histogram of message-sending behaviour. The vast majority of people send only one message, and there are some who send a great many. The second is the histogram of message replying behavior. The next two are based on the social network, where an individual s_a has a link to an individual s_b if s_b replied to a message from s_a. Higher out-degree for s_a is an indication of higher status, since more individuals have found messages from s_a of interest. Individuals whose messages attracted no replies were excluded from this graph. Out-degree also shows a scale-free, or power-law distribution, characteristic of small-world social networks [2, 9, 10, 15]. In-degree measures the number of different people to whom an individual has replied-to, and is an indication of the level of engagement of an individual in the mailing list and the breadth of his/her interests. This distribution also shows a small-world character.

Next, we examine (Figure 2) the relationship between the number of messages sent by an individual, and the number of distinct respondents who replied to that individual. For this graph, we only considered individuals who had actually received at least one response to their message. It can be seen that there is a strong relationship; in fact, we note a very high Spearman's rank correlation, around 0.97. It should be noted that these are not the same phenomena; the number messages one sends need not necessarily correlate with the number of different people that consider that message worth responding to. This may be due to community norms, *i.e.*, people only post relevant messages, and the community by and large responds to messages. It may also be due to a survival effect, whereby only people who receive replies from several people keep sending messages. We are currently using time-series regression analysis to examine the latter theory, that only people who receive replies to their messages continue to be active on the mailing list.

Finally we present (in Figure 3) a pruned email social network; the full network is too large to render in a useful fashion on non-interactive media. Each directional link in Figure 3 indicates a message count of at least 150. For example, the arrow from *Alexei Kosut* to *Ben Laurie* indicates that the latter replied to at least 150 messages from the former;

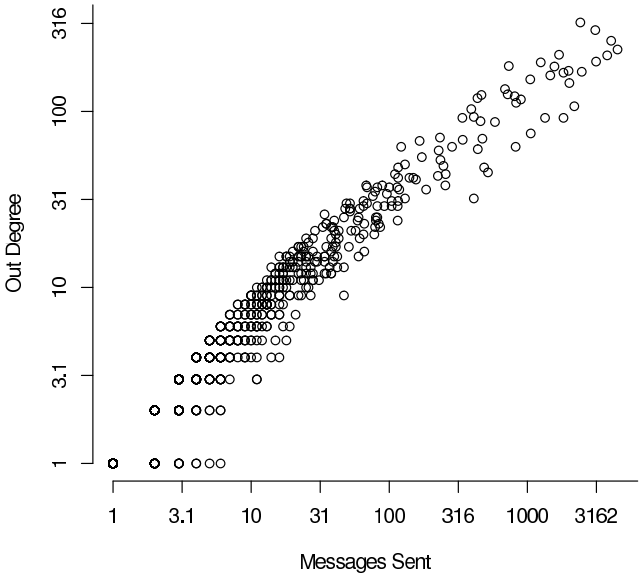

Figure 2: How out-degree (number of distinct respondents) grows with number of messages sent by ego, *n=1063*

this indicates that Laurie found a lot of Kosut's messages of interest. The reverse arrow indicates that this relationship was mutual. Unidirectional arrows, for example from *Slemko* to the *Rodent*, merely indicate that the former replied to less than 150 messages from the latter. Self links indicate that individuals sometimes replied to their own messages, sometimes after comments from others, sometimes to clarify their original message.

The high connectedness of certain individuals (*Gaudet, Laurie, Bloom, Jagielski, Rowe*) can be seen even in this pruned network; these individuals are in fact the most productive developers. Preliminary statistical data further supporting this is presented in the next section.

We conclude this section with some observations:

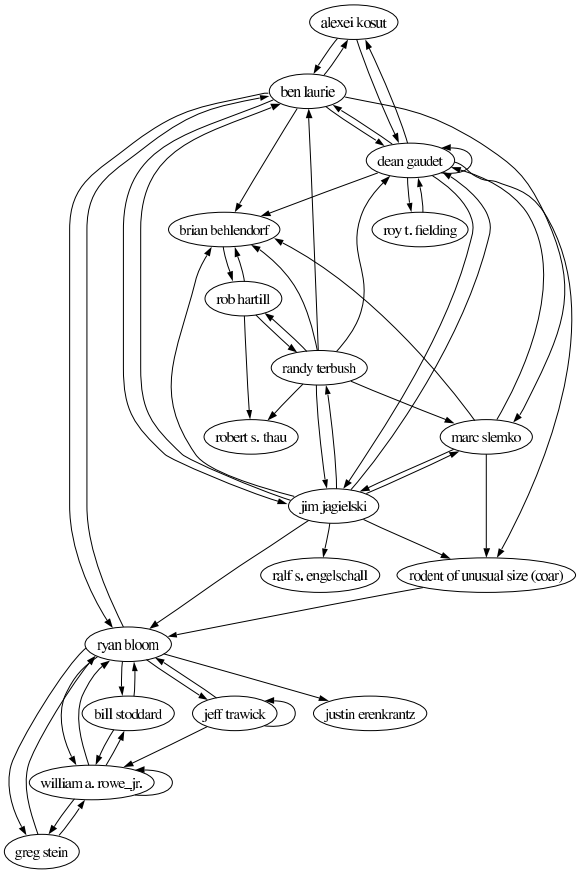

Figure 3: Pruned Social Network of Apache Emailers (Each link indicates at least 150 messages sent, or replied-to).

- *The number of messages sent by individuals, and the number of messages sent in reply to individuals, both follow a Pareto distribution;*
- *The social network of individuals on the email network, where an individual a has a link to an individual b if b replied to a message from a, shows a long-tailed degree distribution on both in- and out-degrees, characteristic of small-world networks.*
- *There is a strong relationship between the number of messages sent by an individual, and the number of* distinct *individuals who respond to that individual (also the out-degree in the social network). We are studying this phenomenon using time-series analysis.*

Next we turn to examine the relationship between email activity and development activity.

5. C&C ACTIVITY AND DEVELOPMENT ACTIVITY

In this section we discuss this question: *How does email activity relate to software development activity.* In order to study this question, we use data gathered from the cvs archives on how many changes (distinct commits) were made by each individual. In fact only 73 individuals have actually made commits to the versioned repository during the period beginning with 1999, (before which this repository was not used) until the present. There are two types of files, source and documents. We counted each separately, in order to study the relationship of source code and document activity with email activity.

5.1 Activity Correlation

There are large number of correspondents on the mailing list who do not have commit privileges, never make any changes to the project files. These individuals tend to be less active on the email list. In order to study the relationship between the effort spend on C&C activities, and development activities, we excluded individuals who have not made any changes to source code or documents from this study. By focusing on just those individuals who have made changes, we hope to get a clearer picture of the relationship of email activity with development activity.

Based on the data for just the 73 committers, we observe a Spearman's rank correlation of about 0.80 between the number of messages sent by an individual, and number of *source* changes they make. This clearly indicates that the more software development work an individual does, the more C&C activity the individual must undertake. There is a somewhat lower correlation, around 0.57, with number of *document* changes. We hypothesize that this is because source code activities require much more co-ordination effort than documentation effort, but further study, using time-series data is needed to determine this.

The total number of messages is only one aspect of a community's structure; the volume of messages sent by an individual (even if they receive replies) doesn't necessarily indicate the individual's position in the social network. Sociologists have invented several measures of an individual's position in a network, when viewed globally. We also study the relationship of some of these measures to the activity level of an individual.

5.2 Social Network Measures

We focus on 3 measures, *in-degree out-degree* and *betweenness*, which are indicators of the importance of an individual in a network. Out-degree and in-degree were discussed earlier; for this part, we normalize out-degree and in-degree by the total size of the network. For a node v in a graph g, betweenness BW is defined as follows:

$$BW(v) = \sum_{i,j,i\neq j,i\neq v,j\neq v} \frac{g_{ivj}}{g_{ij}}$$

where g_{ivj} is the number of shortest paths (geodesics)[3] in g, between i and j, that go through v; and g_{ij} is the total number of shortest paths from i to j.

High betweenness indicates that the person is a kind of broker, or gatekeeper in the social network; s/he plays a role in a great many interactions. Such people can have high status, and can also be bottlenecks. Actors who are high in betweenness centrality have the potential to control or disrupt communication or trust relationships between various end points. So we ask the question, *Are developers more likely to play the role of gatekeepers or brokers in the*

[3]Note that there may be more than one shortest path between two nodes if multiple paths are of the same length.

	changes	srcChanges	docChanges	outdegree	indegree	betweenness	mean	min	max
changes	1	0.789	0.932	0.520	0.474	0.553	912	0	16289
srcChanges	0.789	1	0.514	0.712	0.679	0.757	420	0	5741
docChanges	0.932	0.514	1	0.308	0.263	0.327	492	0	13420
outdegree	0.520	0.712	0.308	1	0.971	0.955	0.0080	0	0.0396
indegree	0.474	0.679	0.263	0.971	1	0.917	0.0067	0	0.0260
betweenness	0.553	0.757	0.327	0.955	0.917	1	0.0011	0	0.0965

Figure 4: Cross-correlation table, (using Spearman's rank correlation) showing relationship between the total number of changes, the changes to source, changes to documents, relative in-degree, relative out-degree, betweenness. Average, min and max are also shown. *n=73*

complete email social network? To answer this question, we computed the betweenness scores of developers ($n = 73$) and non-developers ($n=1123$) in the full email social network. The mean betweenness of developers is 0.0114, and the mean betweenness of non-developer is 0.000140. A simple T-test indicates a t-value of 5.07, which is highly significant. The other measures, out-degree and in-degree were also calculated. They also indicate that developers have a significantly higher status, as indicated in the table below.

	Developer	Non--Developer	T-value	Significance
Betweenness	0.0114	0.000140	5.07	$p < 0.001$
Out-degree	0.00666	0.000451	8.14	$p < 0.001$
In-degree	0.00794	0.000367	7.54	$p < 0.001$

So we can conclude that developers are higher in status than non-developers. Next, we consider just the population of developers, and study the indicators of status within this population.

5.3 Relative Status of Developers

Considering just the population of developers who have made changes to the source and documents ($n = 73$) we turn the reader's attention to Figure 4, which shows a table with the relevant descriptive statistics and correlation values. The top 3 rows (left 3 columns) are measures of activity: total changes, source code changes, and document changes. The bottom 3 rows (columns 4,5, and 6) are indicators of social status.

Considering just the 3 change variables, it can be seen that source changes are not as highly correlated with document changes, indicating that not all developers are engaged in both to the same degree. Thus, developer `nd` made 13420 document changes, and 2869 source changes, while developer `dougm` made 1322 source changes and 74 document changes. There are several others who were skewed in this way.

Turning now to the relative indicators of status, we can see that source changes shows the strongest rank correlation with the social network status indicators of normalized out- and in-degrees, and betweenness. In fact the correlation for betweenness is quite high, at 0.757. It should be noted that these are non-parametric correlation measures, and are thus more robustly indicative of a relationship. This indicates that even within the higher-status group of developers, the most active developers play the strongest role of communicators, brokers, and gatekeepers. It's also noteworthy that the correlation with document changes is much weaker, indicating that higher activity in source code is a stronger determinant of social status than activity in documents.

A later study of the developer mailing list and source code repository data for the Postgres[4] project showed that the social status measures had similar levels of correlation with source code changes [3]. The Postgres data, however, showed much higher correlations between document changes and social network measures than the Apache data. We plan to examine this statistic in future work.

We end this section with several preliminary conclusions:

- *The level of activity on the mailing list is strongly correlated with source code change activity, and to a lesser extent with document change activity.*
- *Social network measures such as in-degree, out-degree (normalized by the number of developers) and betweenness indicate that developers who actually commit changes, play much more significant roles in the email community than non-developers.*
- *Even within the select group of developers, there is a strong correlation between the abovementioned measures of social network importance and level of source code change activity.*

6. RELATED WORK

There has been considerable study of social behavior in on-line communities; we only survey work here in the OSS development context.

Social networks among developers have been studied from other perspectives. Xu *et al* [19] consider two developers socially related if they participate in the same project. Our view is to consider developers related if there is evidence of email communication; this is arguably a more direct evidence of a social link. Wagstrom, Herbsleb and Carley [18] gathered empirical social network data from several sources, including blogs, email lists and networking web sites, and built models of their social behavior on the network; these were then used to construct a simulation model of how users joined and left projects. Our goal is empirical rather than to run a simulation; we explicitly wish to study the relationship of email behavior and commit behavior in a single project.

Crowston & Howison [7] use co-occurrence of developrs on bug reports as indicators of a social link. They empirically demonstrate that the social networks of smaller projects are more central than those of larger projects, presumably larger

[4]http://www.postgresql.org

projects decentralize, to simplify C&C activities. This paper is more centered on the study of individual developers and how their email activity and social status changes with their commit activity.

Commit behaviour in versioned repositories has been used as indicator of social linkage. Lopez-Fernandez *et al* [12] consider two developers to be linked if they commited to the same module, and two modules to be linked if they were committed to by the same developer. The resulting social networks are similar in structure to ours. The work of De Souza *et al* [6] is similar, except that they study files instead of modules. This work also visualizes the changes in social position of developers within the social network over time, and that of modules in the module dependency network. Developers become more "central" in the social network over time. Turning to modules, they found that code ownership in some parts of the system was more stable than others. Finally, we note that these papers study collaboration networks, whereas our focus is more on communication networks; the relationship between the two is a subject of our current research.

7. CONCLUSION

We describe here our work on mining the email social network on the Apache HTTP server project. We had to face head-on the challenge of resolving multiple email aliases that were used by the same individuals; failing to do this would have seriously affected our ability to study the social network of developers. We have hand-inspected our alias resolution for errors; however, we acknowledge that our alias-resolution step is in need of further validation. Our goal is to do this by mailing a sample of participants get an idea of the accuracy of our alias resolution. Furthermore, a small number (less than 1.3%) of email headers could not be parsed; we are also working on resolving this. However, we believe, that a) there are likely to be only a few errors in the aliasing and b) that the preliminary results reported here are quite robust and unlikely to change significantly even as our data extraction improves.

Our analysis indicates that the email social network is a typical electronic community; a few members account for the bulk of the messages sent, and the bulk of the replies. The in-degree and the out-degree distribution of the social network exhibit typical long-tailed, small-world characteristics. We also note that there is a strong relationship between the number of messages sent, and the number of different people who respond to them; this merits further study.

Our preliminary data also indicates a strong relationship between the level of email activity and the level of activity in the source code, and a less strong relationship with document change activity. Our data also gives strong indications that developers play a much more significant social role among *all* the participants in the mailing list. Furthermore, the data also supports preliminary finding that the level of activity in the source code is a strong indicator of the social status of a developer (among other developers); the document activity is not as strong an indicator.

Our near-term goal is to study these effects in a time-series basis, to investigate if there are causal relationships between development activity and social status. We are also very interested in studying the relationship between the architecture of the system, and social network of the developers (which is also known as Conway's Law).

8. REFERENCES

[1] R. Agrawal, S. Rajagopalan, R. Srikant, and Y. Xu. Mining newsgroups using networks arising from social behavior. In *WWW '03: Proceedings of the 12th international conference on World Wide Web*, 2003.

[2] A.-L. Barabási and R. Albert. Emergence of scaling in random networks. *Science*, 286:509–512, 1999.

[3] C. Bird, A. Gourley, P. Devanbu, A. Swaminathan, and M. Gertz. Mining email social networks in postgres. In *MSR '06: Proceedings of the International Workshop on Mining Software Repositories*, 2006.

[4] F. Brooks. *The Mythical Man-Month: Essays on Software Engineering, 20th Anniversary Edition*. Addison-Wesley, 1995.

[5] S. Chapman. Sam's string metrics page. `www.dcs.shef.ac.uk/ sam/stringmetrics.html`.

[6] J. F. P. D. Cleidson de Souza. Seeking the source: Software source code as a social and technical artifact, 2005. http://opensource.mit.edu/papers/desouza.pdf.

[7] K. Crowston and J. Howison. The social structure of free and open source software development. `opensource.mit.edu/papers/crowstonhowison.pdf`, November 2004.

[8] B. J. Dempsey, D. Weiss, P. Jones, and J. Greenberg. Who is an open source software developer? *Communications of the ACM*, 45(2):67–72, February 2002.

[9] L. C. Freeman. Centrality in social networks I. Conceptual clarification. *Social Networks*, 1:215–239, 1979.

[10] M. Granovetter. The strength of weak ties. *American Journal of Sociology*, 78:1360–1380, 1973.

[11] K. Kuwabara. Linux: A bazaar at the edge of chaos. *First Monday*, 5(3), March 2000.

[12] L. Lopez, J. M. Gonzalez-Barahona, and G. Robles. Applying social network analysis to the information in cvs repositories. In *Proceedings of the International Workshop on Mining Software Repositories*, 2004.

[13] G. Navarro. A guided tour to approximate string matching. *ACM Comput. Surveys*, 33(1):31–88, 2001.

[14] M. E. J. Newman. The structure and function of complex networks. *SIAM Review*, 45:167–256, 2003.

[15] J. Nieminen. On centrality in a graph. *Scandinavian Journal of Psychology*, 15:322–336, 1974.

[16] E. S. Raymond. *The Cathedral and the Bazaar: Musings on Linux and Open Source by an Accidental Revolutionary*. O'Reilly and Associates, Sebastopol, California, 1999.

[17] E. Ukkonen. Algorithms for approximate string matching. *Information & Control*, 64(1-3), 1985.

[18] P. A. Wagstrom, J. D. Herbsleb, and K. Carley. A social network approach to free/open source software simulation. In *Proceedings First International Conference on Open Source Systems*, pages 16–23, 2005.

[19] J. Xu, Y. Gao, S. Christley, and G. Madey. A topological analysis of the open source software development community. In *HICSS '05: Proceedings of the Proceedings of the 38th Annual Hawaii International Conference on System Sciences (HICSS'05) - Track 7*, 2005.