Knowledge Homogeneity and Specialization in the Apache HTTP Server Project

Alexander C. MacLean, Landon J. Pratt, Charles D. Knutson, and Eric K. Ringger

Computer Science Department, Brigham Young University, Provo, Utah
amaclean@byu.edu, landonjpratt@byu.edu, knutson@cs.byu.edu, ringger@cs.byu.edu

Abstract. We present an analysis of developer communication in the Apache HTTP Server project. Using topic modeling techniques we expose latent conceptual sub-communities arising from developer specialization within the greater developer population. However, we found that among the major contributors to the project, very little specialization exists. We present theories to explain this phenomenon, and suggest further research.

1 Introduction

Private information is "information possessed by a relatively small segment of the population" [8]. For example, in development organizations certain individuals specialize and become *de facto* leaders within conceptual domains. Latent roles develop and define structure and vulnerability within an organization. In practice, these individuals become centers of private information subcommunities and thereby control the shape and flow of information within that sub-community. The identities of these individuals are latent in that there is often no obvious correlation between the overt organizational structure and the centers of knowledge within that structure.

We should be clear that private information is neither inherently good nor bad. Instead, its influence is context specific. Private information refers to information known to an individual or small set of individuals, not necessarily information that is hoarded or deliberately withheld. Rather, private information exists as a natural byproduct of organizational learning. A deeper discussion is presented by Krein, et al [8].

1.1 Specialization

Developer specialization is a form of private information in which developers within an organization become expert in a particular concept domain. Example domains might include UI developers, "the database guy," or any small group of developers who are essential to interacting with a particular piece of the product.

Specialization is often found in large organizations where the scope of the project is such that no single person has the time or capacity to master all

aspects of a product. Organizations benefit from specialization by minimizing the overlap of skill acquisition since it can be time consuming and expensive. Individuals benefit from specialization through job security and decreased initial training requirements.

Although beneficial when viewed from a schedule and budget perspective, specialization introduces risk into an organization. For example, in an organization with two core developers, is it appropriate to put both of them on the same plane for a business trip? Or, what contingency should the organization put into place in the event that these two developers decide to retire? While development organizations could certainly recover from the untimely departure of specialized developers, and other developers exist who are capable enough to fill the roles, could the overall organization recover from the delays imposed by the resulting loss in productivity and stay competitive in an aggressive marketplace?

Developing a contingency plan for unexpected change is relatively straightforward when the specialized roles are overtly expressed in the organizational structure or are generally understood among the developers. However, danger arises when the roles are latent and therefore difficult to identify.

1.2 The Apache HTTP Server Project

The Apache HTTP Server (httpd) is an open-source HTTP server implementation and the leading HTTP server with 59% market share as of January 2011 [3]. The project is maintained by a group of volunteers from around the world who collaborate through the use of online tools such as email and chat [2]. Most importantly for this analysis, project tenets stipulate that all development and managerial communication must happen in the publicly available mailing lists.

We are interested in this project because of its open-development philosophy (anyone who proves their worth within the meritocracy can contribute) as well as its success. We would like to use open source projects such as those maintained by the Apache Foundation as an analogue for all software development organizations. Whereas it is difficult and rife with legal issues to obtain source code and developer communications from closed source enterprises, it is comparatively trivial to obtain the same information from the Apache Foundation. If we can show that open source projects behave like their closed source counterparts we can circumvent this roadblock.

1.3 Reference Organizations

The authors are personally familiar with three large software development organizations that each exhibit evidence of private information. Each organization is unique with regards to the degree of geographic distribution and size, yet all contain easily identifiable private information. These organizations are presented as references against which we may compare the Apache HTTP Server project. For consistency we refer to the organizations by number, even though we also identify Organization 2 by name.

For each of these organizations we note the degree of geographic distribution in an effort to illustrate that private information exists whether or not developers are collocated. Since the httpd project is massively distributed, it is important to show that private information is not necessarily caused by lack of face-to-face communication.

Organization 1 Organization 1 builds and maintains a large suite of software products comprised of both legacy and new components. The legacy engine is highly complex and is central to the success of the suite. Although a large organization develops the products and builds new features, only two developers may change the legacy engine. Both of the developers have been with the company for over twenty years. Most products in this organization target either consumer grade computers or cloud-based computing systems.

This organization is somewhat distributed (and is growing more distributed over time). However, a large portion of the developers are still collocated.

Organization 2 Organization 2, IBM, is large (426,751 employees, with gross revenues of $99 billion in 2010 [1]) and builds many disparate products. Despite well-established corporate practices, Krein, et al, still found that "loss of specialized information arises from both reorganization and loss of employees" and that "employee loss... creates information gaps" [8]. Their study specifically targeted extended stakeholders[1], and therefore is not a direct analogue to a development organization. However, it illustrates some of the organizational fragility that arises from private information.

IBM is largely distributed, and has employees in more than 40 countries [1]. Many IBM employees telecommute, further increasing the distributed nature of the organization.

Organization 3 Organization 3 is large but still smaller than both organizations 1 and 2. Developers often specialize and become centers of conceptual communities surrounding hardware interfaces, user interfaces, network communication, fault tolerance, and other product specific topics. Unlike Organization 1, this organization primarily develops software that is shipped in embedded devices.

This organization is lightly distributed across half a dozen different locations. However, members of individual product groups are all collocated.

1.4 Private Information

Krein, et al, presented *the problem of private information*, a framework for understanding communication dynamics within a distributed organization. They studied extended stakeholders in an effort to understand how specialized information flows between stakeholders [8]. While inspired by Hayek's work in

[1] Non-developers.

economics [6], Krein's work represents the first application of the notion of private information to software organizations. Since Krein, et al, is an introductory study, few methods exist to aid us in practice. Still, the concept of private information is a powerful guiding metaphor and provides a framework within which to study knowledge homogeneity and specialization.

1.5 Goal

In order to discover pockets of private information within the Apache HTTP Server community, and thereby identify developer specialization, we analyzed committer email records and commit history. The goal was to identify two phenomenon: 1) committers who write exclusively about a particular topic, or 2) topics that are only discussed by a small set of committers. The first case indicates committers who specialize in a particular topic. The second case indicates topics that are dominated by particular committers.

2 Data

We gathered data for this study from the Apache Foundation during February of 2011. Our data set consists of the commit history and email archives for the Apache HTTP Server Project, spanning sixteen years (2/27/1995 - 1/31/2011).

2.1 Mailing Lists

The mailing list archives for the Apache Software Foundation are freely available. They consist of files stored in the mbox format that contain all of the communication on a given channel for a particular month. For this paper we only analyze the "dev" channel of the mailing lists for the httpd project and refer to it as "the mailing list." This channel should represent communication regarding project development. It consists of 124,938 messages and 166 developers[2] (see Section 3.1). The mailing lists were imported into PostgreSQL.

Developer contribution to the mailing list is not uniform. Instead, a small subset of developers generate most of the traffic (see Figures 1 and 2). Figure 1 shows the number of messages sent per developer for the entire time period. Note the power law distribution of developer email traffic. Figure 2 shows the same activity in 2009, illustrating that the activity patterns are consistent regardless of time window.

2.2 Subversion Repository

The commit history for the project consists of 46,336 revisions by 134 developers. However, as with mailing list activity, a small group of developers committed a majority of the changes to the project (see Figure 3). This behavior has been demonstrated in previous studies [7].

[2]Not all registered committers actually made changes to the project.

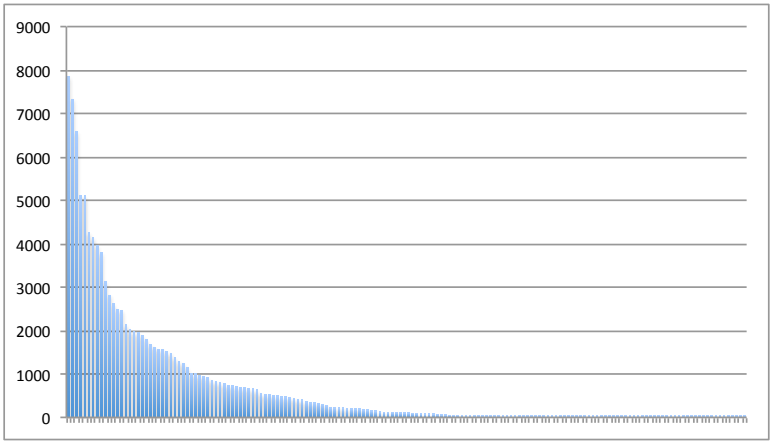

Fig. 1. Messages per developer. Each bar represents a single developer, sorted by number of messages.

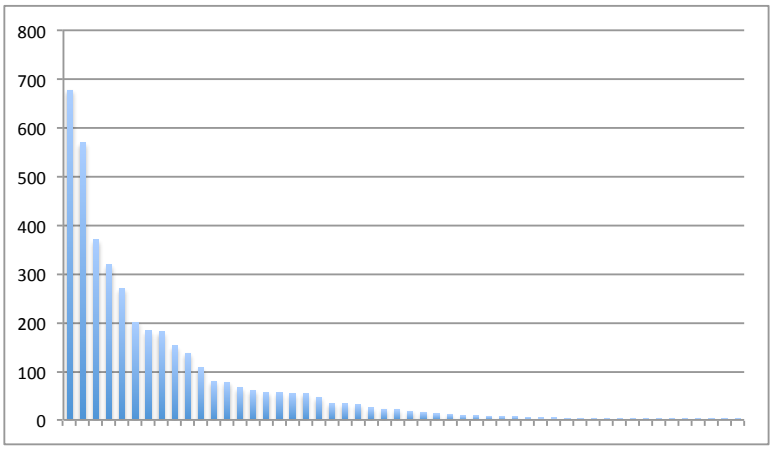

Fig. 2. Messages per developer in 2009. Each bar represents a single developer, sorted by number of messages.

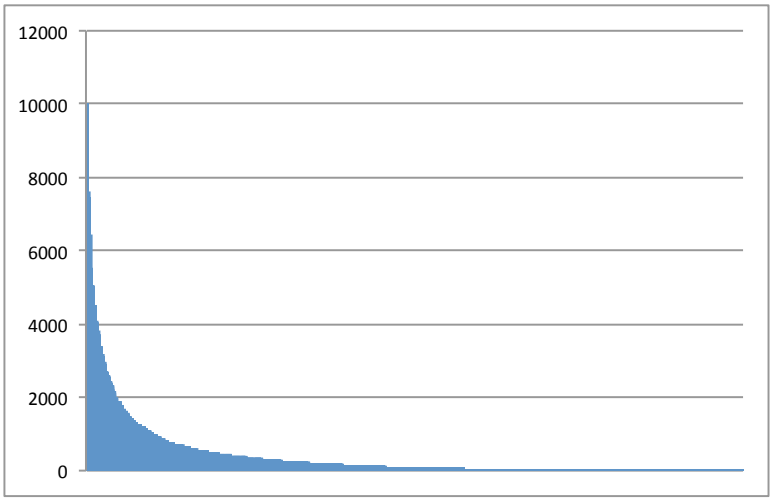

Fig. 3. Developer commits. The x-axis is individual developers, sorted by commit volume.

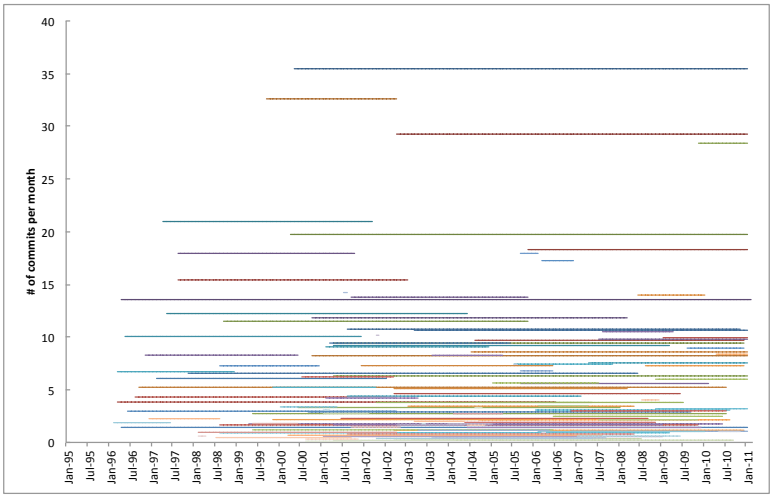

Fig. 4. Developer tenure. Each horizontal line represents a single developer. The length of the line represents the span of a developer's commits, from the first to the last.

3 Methods

In order to identify private information in the Apache HTTP Server community, we first had to associate email topics with developers. To do so, we 1) mapped the committers to email records, 2) cleaned the email records to remove extraneous information, 3) identified topics of discussion in the resulting messages, and 4) constructed a social network model from committers and topics.

3.1 Developer/Email Mapping

Mailing lists for the httpd project are not associated with SVN, and therefore are generally tied to the usernames employed by the versioning software. The only exception is contributors who send email from their Apache email addresses, which are formatted as <svn-username>@apache.org or <svn-username>@<project-name>.apache.org. In either case, the mailing lists simply identify contributors by their email addresses, most of which are non-Apache email addresses and therefore can't be attributed directly to an Apache committer. In addition, many individuals sent emails from multiple email addresses and from clients that employ differing email formatting conventions. We used a simple algorithm to map email addresses to committers:

if email address ends with *apache.org* **then**
  use all text before the "@" symbol as the committer username
else
  if from header contains a name (e.g., "John Lawrence Smith" <jls@nowhere.com>) **then**
    if first name and last name match those of a committer listed on the Apache Foundation Website[3] **then**
      map the email address to the committer
    else
      ignore the email address
    end if
  else
    ignore the email address
  end if
end if

Due to the small number of committers registered to the project, a manual analysis of a random subset of the email was sufficient to confirm that this method accurately categorized the email on the mailing list.

3.2 Email Cleaning

We preprocessed the messages to extract the information contributed by the sender and remove information contributed by others, thereby improving the accuracy of the identified topics. The email messages are in MIME format. Preprocessing consisted of seven steps:

[3] http://people.apache.org/committer-index.html

1. Remove all sections of the multipart emails that were not "text/plain."
2. Remove all header lines.
3. Remove lines that were most likely quoted text from a previous email. We identified four patterns that denoted quoted text (see Table 1).
4. Remove signatures by deleting all contiguous lines following `--\n` until encountering an empty line.
5. Remove all instances of the author's name (see Section 3.3).
6. Remove a list of stopwords[4].
7. Tokenize the messages such that words consist of groups of letters and underscores.

Identification	Action
Lines that started with ">"	Line removed

Sections that started with
 	
`-------- forwarded message`
 	
or
 	
`---------- forwarded message`
	Following text removed

Lines like `<so-and-so> (wrote|writes):`	Line removed
Lines like `On <date> <so-and-so> (wrote|writes):`	Line removed

Table 1. Quoted text identification.

Wang and McCallum [10] used a similar cleaning scheme but indicated that doing so may remove inline responses. However, we found that in this dataset inline responses were generally made to lines of text that started with ">," and therefore only the quoted text was removed. Of a random sample of 100 emails, this process did not remove any lines of text that were not quotes, and only failed to remove all of the quoted text in three instances.

3.3 Topic Identification

In order to identify latent roles and groups in the httpd community that may suggest private information, we extracted topics from messages sent by developers on the mailing list. These topics were identified in an unsupervised/unbiased

[4]Removing stopwords improves the accuracy of LDA by removing commonly occuring terms that are not topic specific.

regex match pattern regexp string cmd regex_t expression preg regular register pcre posix compiled regexec regcomp library pmatch rm_so ap_pregcomp

Fig. 5. The top twenty terms in the regular expression topic. Term importance decreases from left to right, where "regex" is the most important term.

fashion using Latent Dirichlet Allocation (LDA), a Bayesian probabilistic topic model that clusters words from the corpus into topics [4]. "Topics" are probability distributions over the vocabulary words (terms) in the corpus, based upon word collocation within messages. In the process of topic identification, words in the emails are attributed to topics and a mixture of topics is deduced for each message in a manner that depends upon collocation of words within messages. We utilized the MALLET [9] implementation of LDA to identify 200 topics from the email corpus and assign message topic probabilities. Figure 5 shows the top twenty terms in the regular expression topic.

Our initial tests using LDA to identify topics from the email messages resulted in topics that contained author names. We removed the author's name from each email to avoid identifying these name-heavy topics. Although grouping author names with terms provided interesting insight into developer prominence, including the names diluted the topics and decreased the likelihood that less prolific developers would be associated with a given topic.

3.4 Social Network Analysis

From the topics assigned to email messages by LDA we created a two-mode social network consisting of topic nodes and committer nodes. Connections only exist between topic nodes and committer nodes: no inter-topic or inter-committer edges exist. Edge weight between a topic committer and a topic is determined by the weighted proportion of the messages attributed to the committer that refer to the topic, where weighted proportion is

$$\sum_{i=1}^{M_c} p_t(m_i) \tag{1}$$

M_c: All messages attributed to committer c,
i: Message index,
m_i: Message i from M_c,
$p_t(x)$: Proportion of message x attributed to topic t.

This calculation allows committers whose communications on the mailing list are voluminous to dwarf those who contribute less frequently. This seems

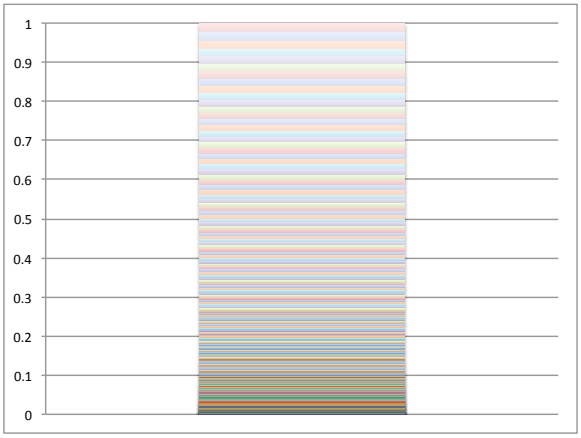

Fig. 6. Proportion of communication for which each topic is the dominant topic in a message.

appropriate in light of the intent of the metric to identify those who have or control knowledge about individual topics. However, it is worth noting that the metric does not capture subtleties, such as committers who specialize in a particular subject, but whose contributions to the project are relatively minor.

4 Results

If specialization exists within the httpd community, we should see distinct communities develop around topics. In addition, unique groups of developers should congregate around specialized subtopics. We examined the data from both angles: topical affinity and topic communities.

4.1 LDA Results

We used LDA to identify 200 topics that summarize the communication on the mailing list. The most prevalent topic described 2% of the text contained in the messages. Figure 6 shows a stacked bar chart of topic proportions over the entire corpus, where each segment of the stacked bar represents a single topic. The median topic described 0.3% of the text contained in the messages. Notably no topics dominated the discussion.

Despite stripping developer names from the emails, a few topics exist that are centered around prominent committers to the project. These topics likely result from other members of the community referring to these more prominent members. We ignored developer-centric topics in our analysis. Although they may indicate private information about a developer, they do not indicate private information about the project.

4.2 Topical Affinity

One measure of developer communication is topical affinity—how much a developer discusses a particular topic. In Figure 7 we see a network that contains the top 22 developers by topical affinity. Note how clusters of topics gather around certain developers. Note also that a small number of developers discuss these topics in such volume that most of the developers are filtered out of this network. This threshold was discovered visually. There is a clear threshold at which the network stabilizes into a small subset of prolific authors.

Regardless of the edge weight threshold, the network looks the same, albeit more or less connected. This trend directly contradicts experience in the three reference organizations (see Section 1.3). If private information exists we would expect to see distinct groups appear within the network as the lower threshold for edge inclusion increased. Instead, we see a core group of developers to whom all topics are connected more heavily than to any specialized group.

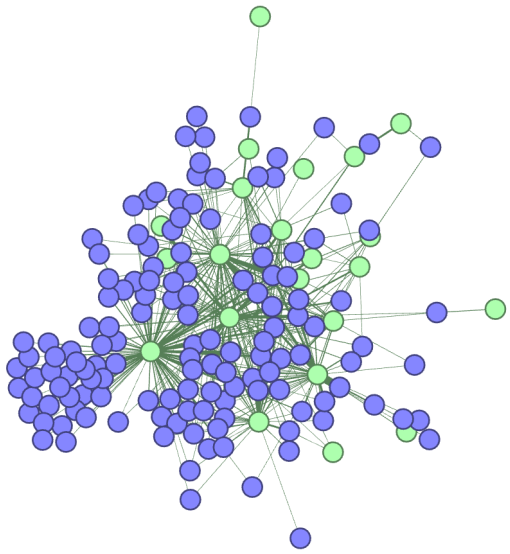

Fig. 7. The top 22 developers on the mailing list by topical affinity. Dark nodes are topics, light nodes are developers. Note that there are strong clusterings of topics around certain developers.

4.3 Topic Communities

We examined the communities that form around major topics, such as voting, SSL, licenses, security, patching, module development, CGI, configuration, regular expressions, and error handling. In each case, the resulting community was

comprised almost exclusively of the core developers. Figure 8 is indicative of the groups that form around topics. In the network, all of the top ten developers are attached to the topic. This pattern repeats itself for all non-developer specific topics. In several cases concepts were split among multiple topics. However, the communities that congregated around the combined topics were also composed of the core developers.

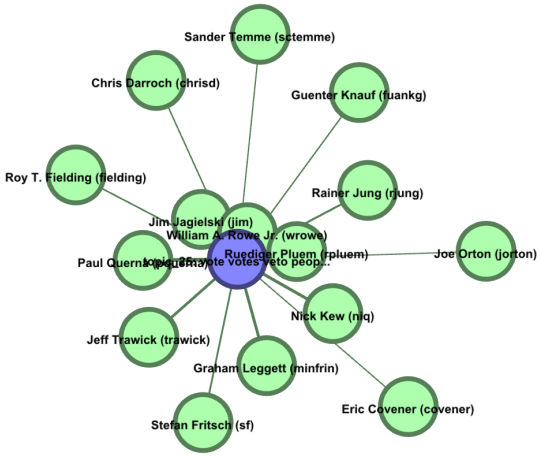

Fig. 8. Developers who discuss the voting topic. Proximity indicates volume of communication.

4.4 Specialization

Because there are no prominent groups in our network of topics and developers, we must conclude that there is little specialization among the core group of developers in the Apache HTTP Server project. This result is especially surprising in light of our reference organizations which exhibit clear specialization. Instead, all of the core developers discuss all major topics with varying degrees of prominence. This suggests that instead of specializing, the core group of developers discuss all of the topics and make decisions as a whole.

In Figure 9 each bar represents a committer and the magnitude of the bar indicates the number of topics for which the committer was either the first, second, or third most prolific commenter (200 topics, 600 total first, second, or third ranks). A small number of core developers dominate the discussions on each topic.

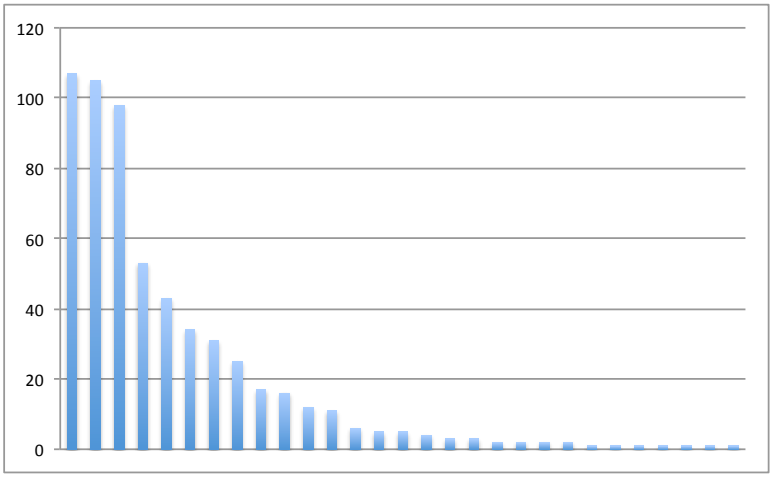

Fig. 9. Each bar represents the number of times an individual was the first, second, or third most prolific communicator in a topic in 2010. There is one bar per developer.

5 Validation

In order to validate our findings, we contacted Dr. Justin Erenkrantz who served as Director of the Apache Software Foundation from 2005 to 2010 and as President from 2007 to 2010. Dr. Erenkrantz, a substantial contributor to the httpd project, corroborated our findings that suggest that little specialization exists among the core developers. This validated our interpretation of the distribution of topic usage in the project. In addition, his corroboration was in stark contrast to the results obtained in the study by Krein, et al [8], where the stakeholders were well aware of specialization and private information within the organization.

6 Threats to Validity

Here we mention two threats to the validity of our study: minor contributors and email text selection.

6.1 Minor Contributors

While our edge weight metric identifies those who are most vocal about a particular topic, it does not identify lesser commenters who specialize in that topic. Further work is required to determine whether minor contributors to the project likewise lack specialization. We suspect that developers who commit infrequently must specialize because they don't have enough familiarity with the system to change multiple components without unintended consequences.

6.2 Email Text Selection

In this study we used only the "text/plain" sections of the emails. However, developers can create emails that only contain "text/html" sections. Although HTML was not common in this project, it could slightly bias the results against an individual who always creates emails this way. However, this was not the case with the most prolific contributors, and therefore should not bias the study significantly.

7 Future Work

This analysis has unearthed a multitude of additional questions, some of which we list here.

7.1 Hidden Private Information

In this study we used topics gleaned from email messages to determine that there was little specialization in the community and corroborated our results through member checking. However, our results were based upon communication and members' impressions. To more fully explain specialization in the httpd project we must look at the commit patterns of developers. If the commit patterns indicate that the core developers work on everything, then our results are confirmed. If they indicate the opposite, then there is an assumption by the developers of a lack of specialization when in fact they do specialize. This result would have fascinating social implications.

7.2 Developer Tenure

During the dot com boom, conventional wisdom maintained that the average developer tenure in a job was 18 months. A study by Fallick, Fleischman, and Rebitzer [5] seems to indicate that average tenure in Silicon Valley has grown to around 4.2 years. However, their work revolved around developer mobility, not tenure, and extrapolating 4.2 years is statistically unsound[5]. Nevertheless, the numbers used in their study still cast heavy doubt on the previous figure of 18 months.

In contrast, the median tenure of developers on the Apache HTTP Server project is 3.7 years. Is this tenure significantly different from the median tenure of developers in traditional, closed source, closed development organizations? Also, when a developer is employed by a company, the company can expect the developer to spend a certain amount of time each week on assigned projects. Is the effort exerted by volunteer[6] developers analogous to that of an employee?

[5]Fallick, Fleischman, and Rebitzer did not make this extrapolation in their paper.

[6]We realize that most of the prominent developers in an open source projects are employed by external organizations and paid to contribute to the project. However, the question is still legitimate.

7.3 Topic Analysis in a Mid-Sized Organization

Only 134 developers have ever committed to the Apache HTTP Server project. Of those 134 developers, the top 30 are responsible for 75% of the commits. Comparing the httpd project to more traditional, closed source organizations of comparable size would provide insight into whether or not the behavior in this FOSS community differes significantly. Ideally this study would involve both collocated development organizations and distributed organizations in order to identify any effect that geographic distribution may have.

7.4 Cross Project Comparison in Apache

If knowledge homogeneity is a symptom of open development, we would expect to see the same kind of knowledge distribution in other projects in the Apache Foundation. All projects within the foundation are governed by a similar set of rules, and all are maintained by volunteer developers. Future work should explore the degree to which specialization appears within other projects.

7.5 Knowledge and Email Communication

In this paper we have been primarily concerned with topic specialization in email messages in the Apache HTTP Server project. However, further work is required to determine the degree to which analysis of topics in email messages identifies knowledge distribution in an open source organization.

8 Conclusions

Lack of specialization within this community may result from a myriad of reasons. Here we explore several factors that may perpetuate this homogeneous knowledge base.

8.1 Organizational Resiliency

Development of the Apache HTTP Server is performed by volunteers, most of whom are paid by external organizations to develop the product. Incentives for the external organizations vary, but one common element holds: developers are potentially transient. At any point a sponsoring entity may decide that a developer is required for an internal project either temporarily or permanently. Although the developer may continue to contribute on personal time, it is unlikely that he or she will be able to contribute anywhere near previous levels. If that developer holds unique specialized knowledge, the knowledge is essentially lost.

By spreading the knowledge across the core members of the organization, this organizational threat is mitigated. We doubt that the organization consciously promulgated a policy of knowledge homogeneity, but rather reacted to organizational fluidity.

8.2 Small Group of Core Developers

A second, related factor may be the relatively small size of the core group of developers. The question becomes: on what are the developers working? If the core group spends much of their time working on bug fixes, then the management of bug priority could homogenize the developer communication. If the developers work together on new features, feature timeline could lead to a homogeneous workgroup—all developers work on the same feature to get it out the door for the next release.

8.3 Small Project Size

A third, trivial, reason, may be that the project is simply too small to require developer specialization. It may be the case that the conceptual domain of the project is small enough that a single developer can, and does, comprehend that entire space. If so, then there is no need for specialization in a project of this size.

However, this reason seems highly unlikely. The conceptual domain contains complicated subdomains such as SSL, user authentication, OS specific tailoring, and transfer protocols. We don't believe that the required depth of knowledge in these subdomains is sufficiently shallow that developers can operate with a broad overview of knowledge.

8.4 Voting

Lastly, organizational culture within the httpd project could *require* homogeneity. Releases and major commits to the project must be approved by a vote of the Project Management Committe (PMC), a group of the more seasoned developers. Informed votes require that members of the PMC are familiar with all aspects of the project and ensure that they are kept abreast of developments.

8.5 Software Engineering Taxonomy

Regardless of the reasons behind the observed knowledge homogeneity, the implications are fascinating and motivate further exploration. By better understanding the similarities and differences between various types of development organizations—be they open or closed source, open or closed development, geographically distributed or collocated, large or small, embedded, desktop, or cloud—we continue to develop a working taxonomy of the software engineering landscape. This taxonomy informs our analysis of future projects and enables better comparison between domains.

9 Acknowledgements

The authors would like to thank Dr. Justin Erenkrantz for his willingness to provide validation and insight regarding the Apache HTTP Server community.

References

1. 2010 form 10-k, international business machines corporation. United States Securities and Exchange Commission.
2. Apache http server project, April 2011.
3. January 2011 web server survey, January 2011.
4. D.M. Blei, A.Y. Ng, and M.I. Jordan. Latent dirichlet allocation. *The Journal of Machine Learning Research*, 3:993–1022, 2003.
5. B. Fallick, C.A. Fleischman, and J.B. Rebitzer. Job-hopping in Silicon Valley: Some evidence concerning the microfoundations of a high-technology cluster. *The Review of Economics and Statistics*, 88(3):472–481, 2006.
6. F.A. Hayek. The use of knowledge in society. *The American Economic Review*, 35(4):519–530, 1945.
7. Jonathan L. Krein, Alexander C. MacLean, Daniel P. Delorey, Charles D. Knutson, and Dennis L. Eggett. Impact of programming language fragmentation on developer productivity: a sourceforge empirical study. In *International Journal of Open Source Software and Processes (IJOSSP)*, volume 2, pages 41–61, June 2010.
8. Jonathan L. Krein, Patrick Wagstrom, Stanley M. Sutton Jr., Clay Williams, and Charles D. Knutson. The problem of private information in large software organizations. In *International Conference on Software and Systems Process*, New York, NY, USA, 2011. ACM Press.
9. Andrew Kachites McCallum. Mallet: A machine learning for language toolkit. http://mallet.cs.umass.edu, 2002.
10. X. Wang and A. McCallum. Topics over time: a non-Markov continuous-time model of topical trends. In *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 424–433. ACM, 2006.