

# Thesis Proposal: Vertical Interaction in Open Software Engineering Communities

Patrick Wagstrom

Department of Engineering and Public Policy

and

Computation, Organizations, and Society

Carnegie Mellon University

Pittsburgh, PA 15213

February 21, 2008

## **Abstract**

Software engineering is still a relatively young field, struggling to develop consistent standards and methods across the domain. For a given project, developers can choose from dozens of models, tools, platforms, and languages for specification, design, implementation, and testing. The globalization of software engineering and the rise of Open Source further complicate the issues as firms now must collaborate and coordinate with other firms and individuals possessing a myriad of goals, norms, values, expertise, and preferences. This thesis takes a vertical examination of Open Source ecosystems to identify the way that foundations, firms, and individuals come together to produce world class software despite their differing goals and values. At the macro level, I examine how competitors work together under open source foundations to create a stable market for each participant. Next I propose a study of interactions between firms in the community to identify to what degree the stated collaboration is actually present. At the next level down I examine how the presence of commercial firms in a community

impacts that participation of individual developers within the community. After I analyze the methods of individual communication in Open Source communities to see if the patterns of communication can be used as a predictive element for project success. Finally, I close with a set of recommendations for firms wishing to participate in Open Source.

## **Introduction**

Software engineering is evolving at a rapid pace. Firms that previously used only internally developed components are finding themselves working in new world with globally distributed development teams, proprietary off the shelf components, Open Source software, standards driving foundations, and industry consortia. Older firms with established products find themselves constrained as platforms evolve, methods change, and experienced developers retire exposing gaps in organizational knowledge. These factors are changing the face of software engineering, forcing firms to collaborate and coordinate across organizational boundaries like never before.

One of the biggest driving factors in this shift is the rise of the Internet and Open Source Software (OSS). Early descriptions of the OSS community often portrayed it as a wild west form of software development: developers work in their free time, code is donated for the good of the community, commercial use is frowned upon, and the only thing that matters is how well you code[13]. While such a view may have been appropriate for small projects in the 1990's, the reality is that OSS has shifted dramatically. Many commonly used Open Source software packages are written by developers working for commercial firms that seek exploit this software as part of their business plan. In other communities, volunteers and commercial developers come together to form foundations to steward and market other projects with little desire for direct commercial gain[14, 12, 17].

The OSS process has become so successful, that many firms are now using similar processes internally for their own development. VA Software markets an enterprise edition of the popular SourceForge.Net

software that corporations can install to manage all of their projects. IBM is developing the Jazz platform to support radically distributed and agile development practices. Like OSS, it focuses heavily on computer mediated communication, iterative development, and gives individual developers the ability to manage their own branches of code and share those branches with other developers. Even Microsoft, which has long been viewed as the antithesis of Open Source, has allowed Open Source like sharing of code between licensees of the Windows CE operating system and recently submitted to licenses for approval to the Open Source Institute – the body that owns the rights to the term Open Source and validates licenses as meeting the requirements for Open Source.

Another major change introduced by Open Source is the commoditizing force it provides[18]. With many complicated pieces of software, there exists common pieces of functionality that each piece of software must implement. For example, in the integrated development environment (IDE) market, each product needs to interface with an editor, various revision control system, and a compiler. Before the success of the Eclipse project, each firm in the market space implemented each of these tools, taking resources away that could otherwise be used to focus on core competencies. Now, various vendors that compete in the market and build on the Eclipse platform can utilize these common components provided by Eclipse and use those resource previously allocated to standard features to further differentiate and enhance their product. Commoditization also provides an opportunity for specific vendors who focused on creating excellent versions of a common component, such as a text editor, to better showcase their work by making it easier for them to create a complete product or integrate their product with others [7, 4].

All of these changes induced by Open Source require additional collaboration and coordination. This required collaboration comes on a variety of levels. At the the highest level is the firm based collaboration and coordination that is necessary for foundations and firms and consortia to collaborate on commoditized software. At a lower level firms need to coordinate their actions and consider the strategies of their competitors in the broader Open Source community when considering what to release to the Open Source community. In addition, when entering an existing Open Source community, if a firm wishes to preserve the delicate

ecosystem already existing around the project they must work within the norms and expectations of the community. Finally, individual developers must be able to communicate efficiently and directly with other programmers, architects, testers, and potentially users – a breadth of communication not previously needed.

This thesis seeks to examine interactions of foundations, firms, and individuals in software engineering. Specifically I examine how Open Source creates new opportunities for interaction, cooperation, and competition across all levels of the Open Source ecosystem. I address these issues through a vertical analysis utilizing both quantitative and qualitative methods to best understand the community, model its interactions, and generate viable models for the development process.

Formally, there are four major sections of this thesis each addressing a particular level of communication within Open Source software engineering. The first section examines the Eclipse community to identify the ways that collaborators and competitors rally around the guiding focus of the Eclipse foundation to create high quality software and extract value from an emerging market. The second section continues to look at Eclipse to understand the firm to firm interactions at various levels in the community and to verify if the communication patterns resemble those that the members of community believe occur. The third section moves further down and examines the relationship between commercial firms and the individual volunteer developers through a longitudinal analysis of the GNOME project. Finally, the fourth section analyzes individual communication patterns within Open Source by building upon the concept of Socio-Technical Congruence.

## **1 Open Source Ecosystems**

While Open Source technology and products have been around since the dawn of computers, the term itself is relatively new, having arisen in 1998 as an alternative to the historic and sometimes confusing “Free Software,” which indicates software that is both free as in speech and free as in zero direct cost. This confusion, and licensing terms around the dominant software license of the time, the GNU General

Public License, had led many firms to avoid Free software and utilize a variety of proprietary or commercial solutions. Even at this point young point, there were firms that had been eeking out a narrow existence in the field for years, but the announcement of the new name and formal definition that was friendlier to business, along with the changes in the information technology industry caused an explosion of new firms to emerge in the latter half of the 1990's. This rapid growth led to a plethora of unsuccessful business models and a handful of successful ones. Amongst the notable successful models were Cygnus Solutions' method of providing customized software development to work with existing tools in the embedded market[15] and Red Hat Software's strategy of giving away a free version of their Linux and then charging for a higher level support for commercial firms that require it[19].

This period also saw the origins of academic research on Open Source and its associated business models. In particular, Hecker described some of the more general reasons why firms might be willing to go into Open Source through his own experiences of releasing the code for the Netscape Web Browser, now called Mozilla, under an Open Source license while at Netscape[8]. Later, Martin Fink identified many of the economic and business options for individual firms working on Open Source, focusing on a more general method of utilizing Open Source strategies and the process of bringing those practices into a firm[5]. His work addresses many of the terms that are commonly utilized in Open Source and begins to address some of the community issues related to participating in Open Source from a corporate perspective. In 2003, Krishnamurthy bean to flesh out some of the more general business models that are in use by firms in Open Source. In particular, he identified four major categories of business models. The first model is that of a software distributor, which is the traditional model of firms such as Red Hat. In this model, firms take a community developed product and repackage the product, providing it for free or a fee and also providing service and support for the product. The second model is that of a derived software producer – a firm that takes the community developed product and derives a new product from it and charges for the enhanced product. A variation of the software producer is a software producer who is working with a produce under the GNU General Public License[6], which requires that derived products also be published and made avail-

able under the same license. In this model, the firm may charge for the derived product and support that goes with it, but they must also provide their modifications back to the community. The final related business model is that of the third party service provider which only provides expert service for a set of Open Source projects and does not directly provide contributions to the project[9].

While these previous efforts all provide valuable contributions to the scholarship in the field, their focus remains on the individual firm despite the fact that the landscape of Open Source has evolved. Successful projects have grown and now feature many competing firms working together on the same components of a larger project and seeking to differentiate themselves based on higher levels of functionality. In particular, most of the models cannot address the difficult issues of what happens when a commercial firm releases an internal project under an Open Source project to the community. In such a case, complete understanding of the contribution requires knowledge of the project and the method of community participation. As this thesis focuses on the interactions between commercial firms and their associated communities, it is important to develop a cohesive model that addresses the communal aspect of participation along with the business strategy for participation.

This section builds on my current work with Jim Herbsleb and Sonali Shah addressing value chains in the Eclipse Ecosystem. Our major data sources are a series of interviews with developers, managers, and executives involved in Eclipse, my personal attendance at EclipseCon in 2007 and soon in 2008, and an analysis of the business models used by all of the strategic developers and plug-in providers of the Eclipse Foundation. Much of this work has already been completed, it will be supplemented with feedback obtained from presentations to the Eclipse Foundation Members Meeting and a presentation at EclipseCon in Mid-March, 2008.

We are currently identifying the various models in use in the community, addressing how money is made as a result of the model, the necessary resources, and how this forces competition in the community. Thus far, our research has identified the following dominant models for firm participation on the Eclipse

Ecosystem:

- **Market Consolidation** - Firms that seek to use the project to compete on a higher level of the value chain. By participating at this level, firms hope to force the other participants to consolidate on their offering. Thus far, such a strategy has proven successful for IBM, the developers of the Eclipse platform. Smaller firms may also employ this model to force the community to accept a particular direction.
- **Commodity Utilization/IDE Enhancement** - Many of the firms that previously competed in the space that has been consolidated may still have products and customers they wish to support. These firms utilize the base of the Eclipse project to build new tools for development. For example, Adobe utilizes Eclipse as the basis for their Flex development studio.
- **Plugin Sales** - While much of the code in the ecosystem is written by professional developers, there is still great incentive to keep modifications proprietary. Some firms develop add-ons for the IDE that are a product unto themselves and would not exist without the IDE as a development platform. This has led to a small number of firms arising that create plugins to enhance the experience, usually by improving over the Open Source tools, as is the case with Instantiations and their GUI designer too, or to provide some new functionality.
- **Plugin Non-Sales** - As a project becomes a dominant ecosystem, vendors of support projects are pressed to create interfaces to Eclipse. This is particularly prevalent for existing source code management and build systems. In addition, hardware vendors, such as Nokia and Intel, provide plugins to ease developers in creating solutions for their platforms.
- **Nested Platform Building** - As the Eclipse ecosystem continues to grow, the code has become more modular and attractive for purposes other than just an IDE. In particular, the OSGi component model and rich client platform (RCP) models make it easy for a firm or group of firms to create solutions without a need for a heavy infrastructure. This trend is becoming more common especially as firms

seek to consolidate commodity components across an industry. For example the Higgins project seeks to provide a common framework for identity management, while the Aperi project utilizes RCP and OSGi to create a standard set of tools for storage management.

- **Customization/Consulting** - As more firms use Eclipse for internal development, there is a greater desire for customized plugins to work with existing work flows. Likewise, there is increased demand for support and easy distribution of Eclipse. A variety of firms have sprung up that provide these solutions, such as Innoopract, which develops an Eclipse “distribution” that packages up eclipse with a set of tested plugins, much like a Linux distribution does.
- **Users** - End users of a software project are often lost when examining a development community. These users play a key role in requesting features, providing testing, and often moving the market in a direction to increase the market share of the product. A wide variety of companies utilize Eclipse, and while they may not be highly visible, they play a key role in the ecosystem.

This section will continue to expand on the current work with Jim Herbsleb and Sonali Shah by expanding and enhancing the data. In our current work we have looks primarily at business models in the community, I intend to examine how the business models relate to the governance of the ecosystem through the Eclipse foundation and flesh out a complete view of the ecosystem of member companies, something which has not yet be done by academia, industry, or the Eclipse Foundation. This will take the form of additional interviews with members of the Eclipse community during and after EclipseCon 2008.

This section of the work greatly adds to the knowledge of software development and Open Source by providing a holistic analysis of a market leading Open Source ecosystem. Indeed, as more and more projects move toward Open Source methods of collaboration such an analysis will prove invaluable in strategic planning and design of such communities.



## **2 Firm to Firm Interaction in Open Source**

While organizations typically have a plan for how they participate in Open Source projects, the reality of their interaction is a very different situation. Understanding the real methods of interaction for communities and firms is beneficial in understanding not only the process of Open Source software development, but also is a valuable component in helping to identify best practices in Open Source software development. This section builds on the results of section 1 by providing an quantitative analysis of how the firms act through observable channels in the ecosystem. In particular, this section will address the following questions:

- How do firms divide work in the Eclipse ecosystem?
- How do we best model the interactions between firms?
- What are the true interaction patterns across different communication mediums in the Eclipse ecosystem?

The data for this section will be collected from publicly available Eclipse resources; CVS source code repositories, Bugzilla bug tracking, project email lists, and community newsgroups. Portions of this data are publicly available through the work of the annual Mining Software Repositories workshop while other elements will need to be coordinated with individuals in the Eclipse community, something I am currently working on.

### **2.1 How is work divided in the Eclipse ecosystem?**

Before founding a project in the Eclipse community (and similarly in the Apache community), projects are placed in an incubation stage. During this incubation stage each project has a mentor or set of mentors to help guide the process in the norms of working within the community – such as open decision making, documenting modifications to the software, and true collaboration with competitors. A project must fulfill

a number of requirements before it can “graduate” from incubation – including demonstrating that a project appeals to individuals from more than one firm. This rule is designed to ensure that projects have sufficiently broad appeal and also to prevent the loss of a single entity from completely sidelining a project.

In some projects it may appear that there is a wide breadth of interest in a project, but what firms are really contributing? Even in platform building projects, such as the Aperi Storage Management framework which sees contributions from most enterprise storage vendors, the bulk of the work is led by only a handful of firms. From a strategic perspective, such inequity forces firms to overly reveal and contribute to a public good for which their competitors may be free riders. Conversely, a firm which contributes most of the work to a project may find itself in a beneficial position because of its ability to drive the direction of the project and the resultant community.

This section will examine a selection of projects within the Eclipse community to identify the distribution of work within the community. A longitudinal analysis will be performed to analyze the contributions by firms through release cycle intervals in the community to understand how projects increase (or possibly decrease) participation over their lifespan. The major elements of analysis will be an examination of contribution to project source code, both at an overall level, by module within the project<sup>1</sup>, and an analysis of contributions to project bug tracking. Such an analysis is invaluable to both firms and foundations in the Open Source community and has significant impact on the future directions for Open Source foundations regarding collaboration and distribution of work.

## **2.2 How do we best model the interactions between firms?**

The communication patterns fostered by Open Source software development typically create sparsely populated social networks with islands and a solid core-periphery structure[2]. However, such analyses typically

---

<sup>1</sup>There are multiple ways to determine module structure in source code repositories. For this work, I take advantage of Java’s explicit module structure.

look at only a specific communication medium, such as discussion on project forums, leaving out a multitude of other interactions. Furthermore, no such analysis has connected the actions of individuals to the firms they work for. This creates artificial holes in the information sharing network that exist through offline communication. In this section, I evaluate a variety of metrics for generating networks from three different communication streams:

- **Highly Technical Interactions - Source Code Networks** - there are multiple methods to construct source code networks, most of which rely on simple links between developers,  $A$  and files  $F$ , which is termed the  $AF$  matrix (in the congruence paper, this was termed the task assignment matrix[1]). Multiplying this matrix by its transpose yields links developers who have a potential for communication because they have modified the same file. However, the larger question is whether or not there should be link from developer  $a$  to file  $f$ . Should this network be built for all of time, or timeout after some predefined period? What if the network is only built for each release, how does the structure change between releases? How does the structure of the network evolve within releases?

Furthermore, relationships between files can be calculated using a variety of different methods. In the past we have used a method that groups together logical dependencies as those files that were modified together and committed in the same atomic commit. This method is particular useful because it is independent of programming language constructs, however, it is sensitive to the work patterns of individuals in the community. Static code analysis, such as that used by MacCormack in his analysis of the Linux kernel and Mozilla projects, has the potential to capture relations that pervade the environment, such as the cascading dependencies that changing a low level debugging infrastructure would have[11]. Another method to compare to is the use of logical code and package structure, such as what is exhibited in the Java and C# programming languages, to create the dependency links between files.

- **Highly Social Interactions - Mailing Lists** - perhaps the easiest method of communication to ana-

lyze. The norm on most lists is to include the original poster of a message in the CC or To fields of the messages. Barring that, most email messages contain reply-to fields that identify the original message and who posted it. However, it may be worthwhile to go beyond this simple structure and include all participants in a thread or a branch of a thread, as a group.

- **Clique** – Working at the thread level, all individuals who posted messages on a particular thread are considered to be in a clique with each other. On a physical level, this would be similar to all individuals being present a shared physical space and interacting with each other.
- **Timed** – A link is added from an individual to all individuals posting within the thread who posted before the individual regardless of what branch they're communicating on. In this method, it is inferred that the individual wishes to communicate with all those who posted before him, but not all those who post after him.
- **Thread** – A link is added from an individual to all individuals communicating on the same branch of communication along a path going back to the root of the conversation. In this way, a communication is not assumed to reach all those involved in the thread, only those along the line of communication within the thread.
- **Direct** – The most conservative method of link generation that assumes a communication only between direct pairs of individuals in the thread reply network.
- **Weighted** – A compromise for many of these methods is to use weighted links and then dichotomize the network at some threshold level.

My previous preliminary research on this topic has shown that the use of the direct method for link creation results in a substantially different ordering of nodes according to network metrics than clique, timed, and thread methods. However, such research was only a small sample mailing list and examined only eigenvector and betweenness centrality measures.

- **Semi-Technical Interactions - Bug Trackers** - traditionally the level of participation on bug trackers

has resided somewhere between mailing lists and source code, however, recent tools that walk users through the process of filing a bug have greatly simplified this process and increased participation. The most popular bug tracker system is Bugzilla, but a variety of other tools also exist. Common among these tools is the ability to receive emails when new comments are added to the bug or fields are changed on the bug. The comment streams for almost all bug trackers is flat, which means it can be difficult to discern the network structure. In general, I will use the same methods proposed in for mailing lists with slight modifications on the non-threaded bug tracker communications.

The appropriate methods of network generation will be selected by analyzing the sensitivity of a family of metrics listed in table 1 for errors caused by omission and commission of links, both of which are potential issues when understand how Open Source ecosystems actually communicate. The goal here is not to identify a single best method, but to be aware of the limitations inherent in generating networks from sparse Open Source communication data. The sensitivity analysis will be done through repeated Monte Carlo perturbation of observed networks on each of the created network structures.

### **2.3 What are the true interaction patterns across different communication mediums in the Eclipse ecosystem?**

The final portion of this section analyzes interactions between firms, grouping individuals by firm to identify how often and across what mediums do firms communicate and collaborate. For each of the projects within Eclipse I propose to generate a network of the firms participating in the project and compare these networks to the information on collaboration publicly available on the Eclipse Foundation's web pages and information obtained from interviews.

This provides an important real world verification of firm communication in Open Source software and can have potentially large impacts on the development of Open Source as a viable communication medium if it is found that there is little communication between firms, which means that a single firm could easily

Table 1: Metrics analyzed on various network constructions

Node Properties	Network Level Properties
<ul style="list-style-type: none"> <li>● Node Degree</li> <li>● Betweenness Centrality</li> <li>● Eigenvector Centrality</li> <li>● Closeness Centrality</li> <li>● Network Size</li> <li>● Network Density</li> <li>● Task Exclusivity</li> </ul>	<ul style="list-style-type: none"> <li>● Density</li> <li>● Generalized Network Structure</li> <li>● Network Level Congruence</li> <li>● Average nodal metrics</li> </ul>

steer the project in a way that is not beneficial to the entire community, or if there is little cooperation on projects, which means that firms may be gaming the rules of the foundation to push projects that have little interest. From a community maintenance standpoint, if such tools become standard in the community, a project with little communication would be susceptible to business pressures on a single firm.

### 3 Corporate Involvement in Volunteer Communities: There Goes the Neighborhood

While much of the publicity on Open Source arises from commercial entities releasing previously proprietary projects as Open Source – for example the release of the Eclipse, Netscape, and Solaris source code – a frequent mode of interaction by firms with Open Source is to contribute to an already existing project with a stable community. In this alternative model, firms must work within the bounds of the community,

surrendering some elements of control in exchange for the goods of the community. Often times firms participating on such projects continue to rely on the contributions of the individuals to the projects for core functionality and complete control of the project is not desired.

While many projects see corporate involvement as a validation of the success of the project and community they have worked so hard to create, there exists possible negative consequences to volunteer communities that welcome commercial developers with open arms. This section identifies and identifies several key issues related to commercial participation in existing Open Source software communities.

In the context of this section, an Open Source software community is a community which produces several different projects, each of which may have different sets of developers. Typically these communities surround larger organizations and foundations which oversee a set of related projects, such as the Eclipse foundation and Apache Software Foundation. Within these communities, most of the software serves a set of related needs and have similar work practices and licensing structures. This allows a single developer to easily change between projects in the community or to work on multiple projects in the community.

The work patterns of commercial developers tend to differ dramatically from those of volunteer developers – not only do commercial developers typically spend much more time working on projects, they also have the ability to collaborate more closely with other developers working at the same firm. These actions allow developers to write more code in a shorter period of time, and may allow for code to be less modular because of the ability to extract information from co-workers at the same site. At the micro level, these changes would be manifest within various smaller modules of individual projects. We reason that the increased **cognitive complexity** of software projects as a result of co-located or full time commercial developers working on a volunteer Open Source project will drive volunteers from those areas of projects populated by commercial developers.

At a higher level, the participation of commercial developers in an Open Source project is likely seen as a mark of validation for the project providing **momentum**, increasing the profile of the project, and

attracting developers who wish to signal their attractiveness to potential employers[10] along with volunteer developers who desire to work in a stable and successful project. In this context, we expect that commercial developers participating in an Open Source project lead to additional new volunteers working on the project.

Commercial developers, in addition to bringing resources and momentum also introduce an element of **heterogeneity** to the project. When a previously all volunteer project attracts commercial developers, individuals may feel as though their work is being exploited for commercial gain or that commercial interests are trying to push the project in a direction that is not suitable for volunteer developers. This causes tension in the community and may force volunteers to leave the project.

It is also unrealistic to group all commercial firms into the same broad categories, while some firms have a very good reputation for working with volunteer communities, other firms are known for being difficult because of their goals or work policies. To account for this, firms are broken into two different categories which may be seen as proxies for the **norms and values** of the firm:

- **Community-focused Distributors** – Firms that focus on taking the complete output of the community and marketing it as a product. For example, Linux distributors such as Red Hat and Mandriva, and Eclipse distributors such as Genuitec and Innooact, take complete products from Open Source communities, provide additional packaging and clean up the software some, and create a marketable product.
- **Product-focused Manufacturers** – Firms that don't need the entire output or work only on a particular aspect of the software package. For example, IBM utilizes the Apache Web Server as part of the Web Sphere application server, but not the complete output of the Apache Project. Likewise, the One Laptop Per Child contributes to portions of the GNOME project, but only needs the results of some portions of the project.

These issues are addressed through the longitudinal analysis of a large and successful volunteer origi-



nated community, the GNOME project. A set of interviews with 18 of the developers helped to confirm our firm classification scheme and provide qualitative confirmation of our hypotheses. This was followed up with analysis of 10 years of longitudinal data collected from project source code, bug tracking, and mailing list repositories.

### 3.1 Preliminary Results

I've attached an advanced draft paper on this issue that will be going to ISR in the near future as appendix A in this proposal. In summary, we have found the following:

- **Cognitive Complexity** - There is no support for the hypothesis that commercial developers will drive volunteer developers to less complex areas of a project. This was tested by examining the effect of commercial developers at the module level.
- **Heterogeneity** - There is no support that the heterogeneity introduced by commercial developers causes volunteer developers to leave projects.
- **Momentum** - The presence of commercial developers was found to have a positive relation to increased volunteer participation.
- **Norms and Values** - When broken up by business model, community focused firms, which share the norms and values of the community were found to have a high and positive relationship with increased volunteer participation, while product focused firms had no statistically significant relation.

### 3.2 Proposed Future Analysis

This work has established that there is a temporal relationship between the presence of commercial developers and subsequent presence of volunteer developers, but did little aside from an examination of the business

models of the firms to identify possible other observable elements that may lead to increased future volunteer participation. I propose to expand this analysis by performing more in-depth analysis of communication in the community. I have previously done some analysis which found that for the most part, community and product focused developers showed little preference in their targets of communication. However, this analysis was at the aggregate level and could have missed important relationships.

- **Preference of Volunteer Responses** - One thing we have not directly tested for is signalling in the community[10, 3]. If individuals are attracted by the presence of a job, they would be more likely to perform actions which can be seen as increasing their profile to commercial developers. I will generate the response networks for messages in the community and test if volunteers preferentially communicate with commercial developers and if such communication leads to sustained participation in the project.
- **Participation in Feature Development** - Another conjecture put forth in the work is that volunteer developers are attracted to community focused developers because the features they develop are more directly applicable to the small user, rather than many product focused firms that focus on enterprise class software development. Utilizing information out of the CVS repository and bugzilla databases this can be tested by clustering the networks to observe the propensity of volunteers to be grouped with community and product focused developers.

## **4 Individual Communication and Socio-Technical Congruence**

Previous research by Cataldo et. al. developed the metric of socio-technical congruence for use in evaluating the fit of an organization's communication patterns to task dependencies [1]. As an organization reaches higher overall congruence between task dependencies and communication, time to resolution of defects was reduced, indicating an improvement in organizational efficiency. As proposed, this metric is a matrix based

algorithm that produces a single number for the entire organization. While such information is useful for a workgroup manager, who is responsible for understanding the overall status of a project, the usefulness of such a metric to individual developers is questionable.

Recently, Helander et. al. have extended the algorithm as a graph theoretic model with improved runtime efficiency and reduction of memory requirements[16]. Chief among the accomplishments of these extensions is the weighting of individual edges and their contribution to organizational congruence and the introduction of the concept of communication gaps to the analysis. In the case that all edge weights are set dichotomized to 0 or 1 then the this algorithm produces identical results to the congruence metric of Cataldo et. al.

I propose extending these metrics to create a family of metrics that individuals can better comprehend and adjust their behaviors accordingly. An individualized version of congruence is necessary to provide incentive for developers to take action to improve their own personal congruence, and thus improve the organizational congruence. A preliminary version of this work was presented at the 28th Sunbelt Conference in St. Pete Beach, Florida in January of 2008.

There are two classes of individualized congruence which I propose - unweighted and weighted individual congruence. Unweighted individual congruence,  $IC_u$ , is very similar to overall network congruence. A coordination requirements matrix  $C_R$  is generated for the entire network as is the actual coordination matrix  $C_A$ . The unweighted for individual congruence for an ego is then calculated by observing the congruence between  $C_R$  and  $C_A$  only for those edges in  $C_R$  that are incident upon the ego in question.

Weighted individual congruence is slightly more complicated, a weighted coordination requirements matrix  $C_R$  is generated either by not dichotomizing the matrix, or through the graph theoretical model of Helander. The actual coordination matrix,  $C_A$  is assumed to be dichotomized.

Formally the individual weighted congruence,  $IC_w$  for ego  $i$  can be expressed as:

$$IC_{w,i} = \frac{\sum (\mathbf{C}_R [i, ] \times dichot(\mathbf{C}_A [i, ])) + \sum (\mathbf{C}_R [ , i] \times dichot(\mathbf{C}_A [ , i]))}{\sum dichot(\mathbf{C}_R [i, ]) + \sum dichot(\mathbf{C}_R [ , i])} \quad (1)$$

Where  $dichot(\mathbf{x})$  represents dichotomizing matrix  $x$  to 0,1 such that all cells  $> 0$  are set to 1, the multiplications are element wise multiplications, and we assume that there are no diagonals in the coordination requirements matrix ( $\forall i : \mathbf{C}_R [i, i] = 0$ ).

## 4.1 Preliminary Results

Using data from the GNOME project with a dependant variable of time to resolve software defects, un-weighted individual congruence was found to have no statistically significant relation. However, this data produces very sparse matrices, which may be overly or underly sensitive. Contrary to our hopes, high weighted individual congruence was found to be related to an increased time to resolve bugs. This result also mirrors analysis done on a proprietary project while at IBM Research. Also, consistent with research from IBM, we found that the more communication required by an individual, as shown by the number of edges in the  $\mathbf{C}_R$  matrix for ego  $i$ , the lower the time to resolve software defects. Like the previous work, we found that high levels of organizational congruence were predictors of faster time resolve bugs. Thus, we have a situation where the organizational and individual level metrics predict results in different directions. The regression for the model can be found on page 14 of appendix B.

This is a rather counter-intuitive result as in the edge case it suggests that highly coupled code with many dependencies and developers who rarely communicate is best for individual performance, but worst for organizational performance. From a tool development perspective such a find is disconcerting because individuals may be wary of contributing toward an overall team goal if it will hurt their individual evaluation. Recently, this result has been found by a team from the MIT Media Lab when testing performance of individuals and teams in German a bank. High levels of overall communication were found to increase overall team performance, but lower individual performance (Waber et. al., submitted to Academy of Management).

I will continue to examine this phenomenon by looking at more data within the GNOME community to examine if this result is consistent across more than the ten sample projects previously presented. In addition, if resources and data avail themselves, I would like to compare the results found in the GNOME community to the results found in Eclipse, which has a more formal operations and participation model.

## **4.2 Uncertainty and Sensitivity in Socio-Technical Congruence**

Despite best efforts in the field, social networks generated from archival data are passable at best, and quite frequently have severe errors and inconsistencies. This problem is magnified in measurements such as congruence where we are concerned not only with the presence of a link, but also the underlying semantic meaning of the link. For example, it may be possible to use email archives to infer that two developers, Alice and Bob, communicated during a period of study. However, absence a content analysis of all communication, it is not possible to know if the communications between Alice and Bob were relevant to their work assignments or not. This uncertainty hits at the core of socio-technical congruence and should it be found that the metrics are overly sensitive to noise, could seriously damage the validity of the metrics.

I propose two different models to understand the uncertainty in this model. In the first model, we assume that each link has a chance of error of omission or error of commission with various probabilities and re-analyze the network outcome. In the second model we assume a prior probability for communication between each ego and alter in the network, and when a communication is found, we update the probability. Uncertainty is accounted for by instantiating the actual communication network according to the new probabilities and calculating the network stats. Currently, due to technical constraints, I plan to only analyze uncertainty and sensitivity on the  $C_A$  matrix, which lends itself to more subjective methods of creation, as previously addressed in section 2.2 of this thesis.

#### **4.2.1 Errors of Commission and Omission**

This model of uncertainty addresses systemic errors in the data collection method, either caused by faulty data, or missing data on communication methods. I propose to run a set of Monte Carlo simulations on multiple sets of real world data. This is a two axis experiment where the rate of omission is scaled from 0 to 25% and the rate of commission is also scaled from 0 to 25%. At each point I will run 100 samples with each data set and generate a distribution of overall socio-technical congruence, along with weighted and unweighted individual socio-technical congruence.

From a technical perspective, such analysis requires significant amounts of computational power, however I have some prototype tools that utilize Amazon's EC2 computing service which can provide hundreds of computational nodes for approximately \$0.10/hr/node – reducing weeks worth of runs to a few hours.

#### **4.2.2 Probabilistic Communication**

Creating a probabilistic communication model is even more complex, as rather than utilizing a uniform probability for the presence of each link in  $C_A$ , the probability is based on the value of that cell and the base probability for a link – in essence a slightly randomized dichotimization of the data. In many situations similar to this, Bayesian inference would be the ideal method for updating priors, yet the data in  $C_A$  represents only positive observations, with no concept of negative observations for decreasing a prior. It is possible that I could proceed using only a low prior and positive observations, and such an option will be explored.

This section of the work will provide valuable insight into the stability of individual socio-technical congruence and also provide additional insight into generating networks from observed data.

## 5 Timeline

- **March 3rd, 2008** - Submit preliminary workshop paper addressing Socio-Technical Congruence at the individual level (section 4) to STC 2008 Workshop
- **March 5th, 2008** - Propose
- **Early March - Mid March** - Collect data on Eclipse Project for section 2. Retool framework for Congruence to allow for multiple dependency network generation methods.
- **March 17th-20th** - Present preliminary findings for section 1 at EclipseCon. Conduct and schedule interviews to further research on sections 1 and 2.
- **March 20th - Mid April** - Conduct follow-up interviews. Parse and load the data related to Eclipse. Explore the implications of network structure and dependency generation on the congruence metric. Generate data from more projects to better understand the “communication overload” phenomenon.
- **April 18th** - Submit more advanced version of individual congruence to CSCW 2008.
- **Mid April - Early May** - Implement and analyze probabilistic model for congruence.
- **May 1st** - Sloan Industry Studies conference provides additional feedback on section 1.
- **May 10th** - STC 2008 workshop provides additional feedback on section 4.
- **Mid May - Early June** - Evaluate methods of network construction using Eclipse data. Perform large scale sensitivity analysis on data of network construction method.
- **Mid June - Early July** - Generate structures for Eclipse to evaluate how firms really work together within the Eclipse ecosystem.
- **Summer** - Hopefully get reviews from ISR for “There Goes the Neighborhood” paper, providing feedback on section 3.

- **Mid July - Early August** - Put final touches on writing of thesis. Convince my wife that she really wants to read thesis to help proofread it.
- **Mid August** - Prepare slides, fudge room of a week or two.
- **Late August 2008** - Defend

## References

- [1] CATALDO, M., WAGSTROM, P., HERBSLEB, J., AND CARLEY, K. Identification of coordination requirements: Implications for the design of collaboration and awareness tools. In *2006 Conference on Computer Supported Cooperative Work* (Banff, Alberta, Canada, Nov. 2006), ACM Press, pp. 353–362.
- [2] CROWSTON, K., WEI, K., LI, Q., AND HOWISON, J. Core and periphery in free/libre and open source software team communications. In *Hawaii International Conference on System Sciences* (2006), p. 118a.
- [3] DAHLANDER, L., AND MAGNUSSON, M. G. Relationships between open source software companies and communities: Observations from nordic firms. *Research Policy* 34 (May 2005), 481–493.
- [4] DES RIVIERES, J., AND WIEGAND, J. Eclipse: A platform for integrating development tools. *IBM Systems Journal* 43 (2004), 371–383.
- [5] FINK, M. *The Business and Economics of Linux and Open Source*, 1st ed. Prentice Hall PTR, Sept. 2002.
- [6] FOUNDATION, F. S. Gnu general public license, June 1991. available at <http://www.gnu.org/copyleft/gpl.html> – Visited April 28, 2007.
- [7] GEER, D. Eclipse becomes the dominant java ide. *IEEE Computer* 38 (2005), 16–18.



- [8] HECKER, F. Setting up shop: The business of open-source software. *IEEE Software* 16 (1999), 45–51.
- [9] KRISHNAMURTHY, S. *An Analysis of Open Source Business Models*, 1 ed. MIT Press, Cambridge, MA, June 2005.
- [10] LERNER, J., AND TIROLE, J. Some simple economics of open source. *Journal of Industrial Economics* 50 (June 2002), 197–234.
- [11] MACCORMACK, A., RUSNAK, J., AND BALDWIN, C. Y. Exploring the structure of complex software designs: An empirical study of open source and proprietary code. *Management Science* 52 (July 2006), 1015–1030.
- [12] O’MAHONY, S. Guarding the commons: how community managed software projects protect their work. *Research Policy* 32 (July 2003), 1179–1198.
- [13] RAYMOND, E. S. *The Cathedral and the Bazaar*. O’Reilly & Associates, Sebastapol, CA, Oct. 1999.
- [14] SHAH, S. K. Motivation, governance, and the viability of hybrid forms in open source software development. *Management Science* 52 (July 2006), 1000–1014.
- [15] TIEMANN, M. *Future of Cygnus Solutions: An Entrepreneur’s Account*. O’Reilly Media, Inc., Sebastapol, CA, 1999, pp. 71–91.
- [16] VALETTO, G., HELANDER, M., EHRLICH, K., CHULANI, S., WEGMAN, M., AND WILLIAMS, C. Using software repositories to investigate socio-technical congruence in development projects. In *Mining Software Repositories 2008* (Minneapolis, MN, USA, May 2007).
- [17] WEST, J. How open is open enough? melding proprietary and open source platform strategies. *Research Policy* 32 (2003), 1259–1285.
- [18] WEST, J., AND GALLAGHER, S. *Patterns of Open Innovation in Open Source Software*. Oxford University Press, 2006, pp. 82–108.

[19] YOUNG, R. *Giving it Away: How Red Hat Software Stumbled Across a New Economic Model and Helped Improve an Industry*. O'Reilly Media, Inc., Sebastapol, CA, 1999, pp. 113–126.

## Appendices

### A “Won’t You Be My Neighbor”

This paper is the preliminary draft for section 3

### B Understanding Communication, Coordination, and Congruence at an Individual Level By Harnessing Baseball Statistics

This is a copy of the presentation that I gave at Sunbelt in January 2008 corresponding to section 4 of the thesis.

# **“Won’t you be my neighbor?” The Impact of Commercial Organizations on a Volunteer Community**

Patrick Adam Wagstrom, James D. Herbsleb, Robert Kraut

School of Computer Science

Carnegie Mellon University

5000 Forbes Ave

Pittsburgh, PA, 15213

## **Abstract**

Early Open Source software development consisted of loosely coupled volunteers contributing ideas and code to create software for a common good. Today, however, a variety of commercial firms often pay employees to contribute to and enhance Open Source projects. While these developers from commercial firms address a variety of needs within the projects, volunteers continue to play key roles in most open source projects making it critical understand the impact the actions of commercial firms and their employees have on volunteers in these communities. We approach this problem by addressing four possible ways that commercial firms can influence volunteer participation. On a purely technical level, commercial developers, through their increased focus and close collaboration with other developers, may increase the complexity of the project rendering volunteers unable to keep up with the changes and driving them away. On a social level, the presence of the commercial developers may be viewed as a sign of success and recognition for an Open Source project, or contrarily that the project has been taken over by

“outsiders”. Recognition and success are likely to make projects more attractive to volunteer developers, leading to an increased rate of participation. On the other hand, however, commercial participation may drive volunteer developers away if overall goals for a commercial firm are at odd with the volunteers, causing tension in the community and increasing the rate at which established volunteer developers choose to leave the project. Lastly, the nature of firms’ participation in the community is influenced by the general business model of the firm, so we break up the firms by broad business models and re-evaluate the previous effects. We explore these issues in a large Open Source community with more than 1,000 contributors. Data sources include semi-structured interviews with 18 highly active participants and a quantitative 7-year longitudinal analysis of development and communication logs. Both types of data show that participation by paid developers attracts volunteers rather than driving them away. The effects are largest when the paid developers work for community-focused firms, whose business strategy is to offer support services and products which complement the broad Open Source project, rather than product-focused firms that focus on niche projects from a broader Open Source community. Developers employed by community focused firms tend to be more broadly involved with the community and provide valuable assistance with the information seeking actions of potential new developers, although these factors do not by themselves mediate the effects of commercial developers on volunteer participation.

## **1. Introduction**

The core of early Open Source software projects were distributed volunteers who freely exchanged ideas and code to create software for the common good of contributing individuals.

Over time, many of these projects became robust enough to attract a wide variety of contributors

and end users resulting in the creation of a community of both developers and users. These communities were held together by a common set of norms and expectations. Central to most communities were the norms of sharing modifications of the software with the greater community and governance by a meritocracy – a system that gave those who made the greatest contributions to the community the ability to directly modify the project source code and control the overall direction of the software.

Today Open Source projects have evolved and many projects have a variety of commercial firms with full-time developers contributing to the project. In the Firefox web browser, Linux operating system, and OpenOffice suite of office programs, volunteer and paid developers from numerous firms collaborate to plan and develop key features of the software. These firms and their associated developers may have different goals from those of volunteers and may not be familiar with the Open Source development process and community norms. Their presence in Open Source projects could either foster or disrupt the original volunteer communities. While previous research has addressed the motivations and actions of individual commercial developers in an Open Source software environment, there has not been any analysis of the overall community impact of commercial participation (Roberts et al 2006). The primary goal of this article is to determine how volunteer developers react to commercial participation in Open Source communities and to better understand what attributes of commercial firms lead to successful commercial/volunteer partnerships. In the following sections we describe four distinct routes through which commercial firms could influence volunteer participation in Open Source projects.

**Comment:** Highlighted references indicate references that do not yet appear in the bibliography. For some reason Zotero is being picky. If I have to, I'll type up the references by hand at a later date.

## **1.1. Commercial Participation and Project Momentum**

At a project level commercial participation in Open Source communities brings increased overall visibility to the community and its participants. Commercial firms often add the project to their existing offerings – providing wider distribution, issue press releases about the software, talk about the community at trade shows, and encourage their employees to become active within the community through the use of mailing lists, wikis, bug tracking, and weblogs. In addition to providing a forum for developers to discuss, plan, and share what they're working on, these websites provide easy access for individuals outside the community to see and learn about what is going on in the community, including new releases, new developers, and new corporations involved in the community. An individual who is only marginally involved with the community and looking for ways to get involved may see such investment by commercial firms as validation of the project and seek to participate and contribute to such a project because of the possibility for future rewards, such as increased technical know-how or the possibility of career advancement (Lerner & Tirole, 2002).

If commercial developer participation validates the importance of the project and increases the momentum, then an influx of commercial firms and paid developers should attract volunteer developers and increase their participation in the community. The number of changes that full time commercial developers can make, and their high level of skill may speed up the development process, increasing the utility of the project to community members and contribute to volunteers identification and attachment to a successful project. Such attachment with projects, communities, and movements is an important factor in volunteers remaining active in a

community and overall community success both in conventional volunteer organizations and Open Source communities (Kleidman 1994) (Lakhani & Wolf, 2005).

**Comment:** Yes, I know the cite is messed up here.

**Hypothesis 1:** The participation of commercial developers on an Open Source project will increase volunteer participation in the project.

### **1.2. Negative Impacts of Heterogeneity**

However, just as participation by commercial firms can provide resources to a community, they also introduce heterogeneity into the pool of developers. Whereas initially all of the volunteer developers may have been able to rally around the primary focus of the community, developers employed by commercial firms may be working just for a pay check, with little concern for overall community health and well-being. In the long run, such heterogeneity in workgroups decreases overall productivity and increase tension within teams (Pelled et al 1999, Williams and O'Reilly 1998). Open Source communities have additional issues of heterogeneity which result in decreased performance, such as personal ideology for community participation (Stewart and Gosain, 2006).

**Hypothesis 2:** The participation of commercial developers on an Open Source project will decrease volunteer participation in the project.

### **1.3. Business Models and Community Norms**

In the late 1990's when Open Source was first attracting interest from commercial firms, most has similar business models. These firms followed the model of Linux distributors, such as Red

Hat, that took the output of the community as a whole, packaged it with documentation and additional software to make it easier to use, and then sold the complete collection of software with enhanced support(Young, 1999; Tiemann, 1999). These companies tied their financial success to the success of the Open Source community as a whole. More recently as the market has matured, additional business models have arisen that allow firms to isolate and derive revenue from a single component from a larger community or start their own communities around small niche products (Krishnamurthy, 2005).

A key element in determining which business models may be successful in a community is the choice of license for the software. Open Source licenses vary in their restrictiveness. Some like the GNU General Public license require that anyone who distributes a modified version of the software release all their changes under the same license. Others, like the Apache and MIT licenses allow firms to customize projects without the need to Open Source their modifications. Previous research on participation in Open Source found that the choice of license had an impact on the participation in the community. Relative to projects with restrictive licenses, projects with more liberal licenses, such as the MIT, BSD, or Apache licenses, were found to have greater interest and often see large amounts of commercial interest because of the ability of commercial firms to appropriate the software into a new commercial product with few restrictions (Stewart, Ammeter, & Maurping, 2006).

The choice of license also helps restrict business models related to a community. Licenses that require that modifications remain Open Source typically are difficult, but not impossible to build



a business around, especially if the firm is not the copyright holder of the software. More liberal licenses allow any firm to begin commercializing a project without the need to Open Source modifications. We identify two broad classes of commercial firms based on their business model and interactions in the community: community focused firms that package the entire output of a community, such as Linux distributors, and product focused firms that utilize only a portion of the output from the community for their products. Product focused firms typically enhance particular components from a community, such as a component library, or focus on a particular application in their business model. As many of the volunteers identify with the entire community of projects, rather than a small component in a project, it is likely that community focused firms will be viewed slightly better than product focused firms, leading to an increased power in attracting new volunteer developers.

**Hypothesis 3:** Community focused firms will have a more positive relation to the number of volunteer developers than product focused firms.

#### **1.4. *Cognitive Complexity at the Module Level***

At the heart of Open Source projects is source code – sets of files written in various programming languages that enable the primary functionality of the project. Techniques for managing Open Source have continually evolved, with more advanced programming languages and improved software engineering practices. The code for complex projects may consist of hundreds or thousands different files, each performing a specific task. To assist in developer logistics and comprehension, within large projects code and responsibilities are typically broken

up into smaller components, called modules (Parnas, 1972). For example, a simple email client may have three modules: receiving mail, sending mail, and user interface. Each module may have a set of developers who are responsible for maintaining the module and overseeing development. Organizationally, modules often replicate the structure of the larger project – complete with their own mailing lists, bug tracking, and social norms.

Because of the distributed nature of most Open Source software development, projects have adopted strong norms of open communication and decision making. For example, the Apache project requires that all decisions reach consensus on publicly accessible mailing lists. However, collocated developers employed by a commercial firm who work closely together have decreased incentive to post to the project mailing lists and create a transparent decision and documentation process. Such a process increases the cognitive complexity of code and prevents volunteers from fully understanding the logic of the new code. The loss of open discussion allows collocated developers to create code that is less modular making future changes more difficult further decreasing participation (MacCormack, Rusnak, & Baldwin, 2006). Because commercial developers work full time, they change project code much faster than volunteer developers. A survey of volunteer Open Source developers found volunteers average 14 hours a week on Open Source projects only a third of a standard 40 hour work week (Lakhani & Wolf, 2005). These issues posit a real danger that as developers from commercial firms modify code within a module of a project, it will become increasingly difficult to for volunteer developers to comprehend the set of the changes, forcing the volunteers previously working on the module to migrate to alternate modules within the project or leave the project completely.

**Hypothesis 4:** The participation of commercial firms in modules of an Open Source project will reduce volunteer participation in those modules.

Our goal in this paper is to identify the net effect of commercial developers on volunteer developer participation and to understand the mechanism through which this occurs.

## **2. Research Method**

Open Source software projects have rich historical archives of communication. For many projects, every communication, idea, and decision is automatically recorded by project support tools. While it is possible to gain useful insights into a community using just the information, the myriad of interactions possible in addressing the role of commercial firms in Open Source requires qualitative as well as quantitative analysis to best understand the nuances of how communities and commercial firms interact in addition to such quantitative analysis. Our research utilizes a multi-pronged approach to understanding these questions around commercial participation in Open Source software communities

We conducted two studies focusing on a single large Open Source community. The first study was a qualitative study to identify the views of developers toward commercial participation and to provide additional background context about the community. The second study analyzed quantitative data obtained from the community against a set of hypotheses regarding commercial participation in Open Source suggested by previous research and the results of the qualitative study.

## **2.1. Community Background**

Our research focuses on the GNOME project, a large and successful Open Source desktop environment started by volunteer developers in 1997 as a response to the lack of a completely free and Open Source desktop environment for Linux and other free computer operating systems. By many metrics, this has is a highly successful project: more than 10 years of history, stable releases every six months, and a continually growing user base. It is the desktop environment for computers from Sun Microsystems, software from the project is in use in a myriad of devices like the One Laptop Per Child and Nokia n800 series of Internet tablets, numerous startup firms have created solid businesses around the project, and it recently was made available direct from Dell computers as part of their option to provide Linux on new computers. In our period of analysis, which goes from the origins of the project in 1997 to late 2006, there were over 1200 individuals who had gained “commit” status, the ability to directly modify the project source code without needing to go through an intermediary, and almost 1000 different components in the shared repository. The community coordinates most of their activity through Internet enabled tools such as a shared bug tracker, mailing lists, and real time chats. The community has been previously studied to understand some of the aspects of software development in the community (German, 2004) and earlier efforts have identified some of the coordination issues related to the project (Koch & Schneider, 2002).

One of the key elements of the GNOME project is that it is comprised of many smaller projects of varying size, complexity, and maturity. For our purposes, when we refer to the GNOME project as a whole as the “community”, and a “project” is one of these smaller projects in the

community. The community acts in a federated manner, providing each project with the opportunity to control their own outcomes and chart their own roadmap subject to some broader constraints and goals developed within the community. When a project has reached sufficient maturity, the developers may apply to have that project included as part of the main community software distribution. Most projects in the community have their own mailing lists and bug trackers and are generally managed by individuals working on those projects. Most community participants are active on multiple projects, but because there are hundreds of projects within the community, there are no developers who are active in, or able to monitor all the projects.

The community has a track record of commercial investment – during the dot-com boom of the late 1990's several firms were created to customize the project, develop components, and provide support for users of the software. However when the bubble burst in 2000-2001, many of these firms went bankrupt or left the market, leaving critical components largely unmaintained. The community slowly built up commercial support again and now has significant corporate investment from firms that distribute software as a component of the Linux operating system, and from other firms that utilize the software as a base toolkit that can be used for the design and manufacture of embedded devices such as PDAs and mobile phones.

### **3. Study 1: Developer Interviews**

Our first study was a set of qualitative interviews designed to better understand the community and their relations to the commercial firms. The first author attended one of the two major annual face to face meetings for both volunteer and commercial community participants. These events are generally considered to be one of the highlights of the year for the project and take

place shortly after the major releases of the software, approximately every six months. To encourage participation by volunteers in the conferences, the GNOME Foundation provides travel stipends to volunteers in the community to attend the conferences. While this helps volunteers attend the conference, because of issues with getting time away from work or school and the limited number of stipends, volunteers are typically underrepresented at the conferences.

Before attending the conference, key individuals were identified and contacted to schedule the interviews, and the most active firms in the community were researched and classified according to their business model. During the course of the conference, a total of eighteen individuals were interviewed over the course of three days. Interviews were semi-structured, lasted twenty to forty minutes, and were conducted during breaks in the schedule. Each interviewee was asked for the relevant professional background, how they got involved with the community, where they currently participate in the community, how they relate to commercial developers in the community, and if they believed our firm classifications were accurate.

General descriptive statistics about the interviewees can be found in Table 1 below. Of interest is that only fourteen of the eighteen individuals could directly commit to the project source code – two of the newer developers, one commercial and one volunteer, still needed to contribute through intermediaries and neither of the individuals who self-described their role as community support could make changes directly to the source code. As is typical in Open Source all the interviewees were male.

**Table 1: General Description of Interviewees**

Total Interviewees	18
--------------------	----

Commercial Developers	50%
Volunteer Developers	50%
Student Volunteer Developers	17%
Commit Access	78%
Self-Described as Developer	89%
Self-Described as Community Support	11%
Longest Participation	10 years
Shortest Participation	11 months
Median Participation	3 years

The nine volunteer participants had varied backgrounds. Three of the volunteers identified themselves as students who primarily participated during their free time. The other six volunteers indicated their use and participation in the project was related to their roles at work. All six of these non-student individuals admitted to contributing to the project while they were “on the clock”, even though participation in the community was not an official part of their jobs. These participants believed their participation was relevant to their jobs and participation improved their performance at work. The participants came to the project through a variety of routes. Most first became interested in the community because of their general interest in Linux and technology, but their reasons for changing from a passive community member who only uses the software to an active, contributing, member varied. Three of nine volunteer developers indicated that another individual working in the community had played a very large role in bringing them in to work on the community. Two of the developers indicated they started submitting changes to a project in the community and were later offered the chance to become maintainers of the project. The remaining four volunteer developers could not identify a specific reason they became more active in the community. Five of the commercial developers were active as volunteers in the community before they were hired. The remaining four commercial

developers were hired by the firm for other projects and later shifted to projects in the community.

“If it weren’t for [commercial developer name], I wouldn’t be involved in the community. He saw my postings on the mailing list and encouraged me to get more involved. About a month later he asked if I would like to maintain the project.”

– Volunteer developer speaking about how he became involved

### **3.1. Views of Commercial Participation**

Both commercial and volunteer developers thought commercial developers provided manpower and the focus necessary to accomplish tasks that volunteer developers lacked the skill or motivation to accomplish. Additionally, the commercial developers believed their firms provided a marketing force for the community, increasing the appeal and bringing in more individuals to work on and participate in the community.

Volunteers generally welcomed the expertise and effort that commercial developers provided. One volunteer explicitly stated he hoped that his participation would be noticed by commercial developers and they would offer him a job, as they had for one of his friends. Three of the volunteer developers believed there were times when the heterogeneity introduced by commercial developers was beneficial – in particular the skills of commercial developers were sought for highly technical areas such as system performance and low-level libraries that volunteers often could not develop. None of the developers, volunteer or commercial, ever mentioned intentionally treating another individual differently because they worked for a



different firm or were a volunteer, although a commercial developer did indicate that he believed code written by volunteers wasn't always as useful as code from his firm. At a modular level such a comment highlights the differing directions and goals of commercial firms and volunteer developers and could cause developers to move to other areas of the project, or other projects entirely.

Two of the commercial developers who began working in the community as volunteers expressed a small amount of frustration in aligning the goals of their firm and the community – possibly alienating volunteers in the community, but generally thought their firms had found ways to succeed. In one case the firm adopted a set of dual processes for participation in Open Source, where developers had individual responsibility for ensuring their participation was congruent with the values and norms of both their firm and the project. This caused problems with documenting work and assigning work – especially in the case of volunteers working on a component that the firm needed designed differently. The other developer indicated that when his firm first became involved with Open Source, they intentionally restructured the norms of the development team to align with the community and not the firm. Although this caused contention within the firm, it was thought to be best for the community.

“I certainly would not want to see commercial participation go away. But I think there are things that some companies should be more careful of when working in the community. At [firm name], we've been very careful how we work with the community.”

The need to be careful when choosing how to participate was echoed by a commercial developer who had been active in the community for more than five years and had worked with multiple firms. He was currently employed by a product focused firm, was critical of his firm's participation, believing that his current firm had little respect for the founding value of the community: freedom. Rather he believed his firm was involved only the sake of exploiting the community for their own products, and had little interest in the health and values of the overall community. This view was in sharp contrast to his previous experience at a community focused firm that he described as fostering involvement within the community. This developer left the product focused firm and the entire community shortly after the conference and his departure stirred up debate within the community about how firms should interact with each other and volunteers.

### **3.2. Classification of Firms**

The interviewees were asked about their views of the nine largest firms, as measured by the number of changes made to the community source code repository. As researchers, we had previously classified the firms according to business model – five of the firms were product focused firms, which worked primarily within smaller areas of the community code, and four were community focused with contributions to many projects within the community. A brief description of each of these firms can be seen in Table 2. Each of the interviewees was provided a description of our classification scheme and asked to classify each of the nine firms. Out of the 162 classification tasks across interviewees, only two were not in agreement with our

classification (Fleiss'  $\kappa=0.953$ ). Incidentally, the two points of disagreement were both employees of a product focused firm who believed their firm was community focused.

**Table 2: Major firms participating in the community**

Product Focused Firms		Community Focused Firms	
Firm A	A large American IT firm that became involved in the last five years through the purchase of Firm B. Migrating from a community focused to product focused firm.	Firm F	A large American Linux distributor. Long time supporter of community.
Firm B	A medium firm that developed enterprise class software and provided services for the community. Purchased by Firm A.	Firm G	A large American IT firm that uses the community software to compliment hardware offerings.
Firm C	A small European firm that assists in application development for the embedded market.	Firm H	A European Linux distributor that historically shipped a similar product from a competing community and had small participation in the community.
Firm D	A small European firm that produced enterprise class applications for the community. Ceased operations in 2002.	Firm I	A medium European Linux distributor that historically supported the community and that of its competitors.
Firm E	A small venture capital funded American firm that developed software and sold integrated services for the community. Ceased operations in 2001.		

However, when asked about perceptions of specific firms the views of interviewees varied. In particular, most of the volunteer developers believed that product focused firms had more difficulty working with the community than community focused firms. Attitudes were almost

universally favorable of community focused firms. In contrast, product focused firms had mixed perceptions from developers and, in the words of one volunteer were guilty of “not caring about volunteers.” Another volunteer who maintained a project within the community that had contributions from about ten developers was extremely skeptical of participation by a major firm, despite being good friends with many of their developers and contributing to other projects maintained and stewarded by the firm. He expressed concern about the method of participation by the firm and the fact that they didn't require everyone to go through the same community socialization process before gaining committer status. This led him to be wary of contributions to his component from the commercial firm. Later in the interview process, when five of the other volunteer developers were asked specifically about this firm, they all echoed similar concerns about the firm's participation.

“I don't think the commercial firms have the same interests as volunteers. If they submitted code to my project, I'd accept it, but if they started to submit lots of code, I'd start to look a lot more at where the project was going.”

– Volunteer developer and project maintainer

These interviews paint a mixed picture with regards to commercial development. While most developers indicated that they appreciated commercial development in the community, a substantial portion of the developers were skeptical about the behavior of these commercial firms. The views expressed toward commercial firms by the developers indicate that there may

be a relation between business model and perceived attractiveness of commercial firms in the community.

## **4. Study 2: Quantitative Analysis**

Theory surrounding the issue of commercial participation in Open Source communities and the interviews conducted in the first study provide a foundation for our second study, a longitudinal analysis of historical data obtained from the community. We begin by further validating the classifications proposed to the interviewees through an analysis of the behavior of commercial developers at a highly technical level that requires significant dedication to participate in (interactions in project source code), a focused technical level open to anyone but with a moderate learning curve (bug tracking system), and an open potentially non-technical forum with little learning curve (mailing list). This data is then used to develop profiles of developers and projects and test for the effects of commercial participation on volunteer participation at the project and module levels.

### **4.1. *Data Collection and Analysis***

Most Open Source projects follow a set of norms that are generally referred to as “the Open Source process.” A key component of this process is the collection and archival of nearly all communication data as a form of organizational memory and as a tool for developers and users to later reference. In most communities public mailing lists are archived where they can be easily indexed and searched, bug tracking systems provide a complete audit history of every

change made to each bug report, and a version control system manages and records all changes made to the software. When using a version control system, each developer downloads a complete copy of the code for the project, makes and tests their modifications, and then sends information about the files that were modified back to a main server in a single action called a commit. Each change is tracked in the system, allowing developers to revert to a previous point in the development process or “roll back” changes that may have been detrimental to overall development while providing a method of providence for all code modifications (Fogel, 2005).

Working with the system administrators in the community, archival copies of mailing lists, bug databases, and version control system were obtained. During the period of study, the community utilized the concurrent version system, CVS, software package as their version control system. CVS access was only granted to approved developers based on a request, limiting the number of individuals who could contribute to those who demonstrated significant dedication to the project. Bugs were managed and tracked using the Bugzilla software package, allowing anyone with an account, available instantaneously through a web form, to submit and comment on issues related to the project. The mailing lists were managed using the Mailman software, and most lists were open to anyone with an email address.

Each of the tools utilized different account and identity management solutions, all accounts belonging to developers were manually unified and linked together within the data set. This allowed us to simply and directly obtain all contributions for developers across different projects and mediums. Information about developers was augmented with information about which

companies they had worked for and the approximate dates of their employment. This was used in classifying a developer's participation as either volunteer or commercial allowing the analysis of firm level behaviors in the community.

#### **4.2. *Product Focused vs. Community Focused Developers***

The interviewees supported our idea that there were two different types of firms in the community – community and product focused. Initially it was thought, based on some of the comments of the interviewees that community focused developers may have more experience and thus be seen as experts in the community – a reasonable assumption as the first firms to invest in the community were community focused firms. However, there was no statistical difference between the experience in the community for product focused (5.24 years) and community focused (5.51 years) developers – however both had more experience in the community than volunteers (3.93 years).

Several interviewees also believed that there were observable behaviors that differed between community and product focused developers. Based on interview responses and personal experiences in the community we identified and analyzed a set of behaviors that may be seen as pro-social and community building. These behaviors have the primary characteristics of showing an interest in the community beyond the narrow focus of products the developer is paid to work on or are behaviors which have a high probability of interacting with individuals in the community who are not already developers. As developers were active in the community for widely varying amounts of time, we normalized each level activity by the number of years the developer was active in the community.

The first way that individuals outside the community are likely to interact with commercial developers is through project mailing lists. Individuals that start many new discussions and reply to a variety of messages are likely to interact with a variety of volunteers and address issues raised by the community. Beyond being highly active, the number of mailing lists a developer posts to also increases the sense that the developer is building community, especially if the developer is active on mailing lists of projects for which they have not written code. We examined the mailing lists from the community and for each developer counted the number of messages posted, new discussion threads started, projects mailing lists they were active on, and the number of projects they posted messages to for which they had never contributed code. Each of these values were normalized by the number of years the developer had been in the community, as measured by the duration from their first observable contribution in any project to their last observable contribution (or the end of the data set if still active). We then took the mean across each of the classes of developer; volunteer, product focused, and community focused; and performed an ANOVA between the three means. The results as shown in Table 3 indicate that after normalizing for experience in the community, there is a significant difference in participation patterns between the three classes of developers. In particular, commercial developers were found to be much more active on mailing lists. However, when tests were performed analyzing just the difference between product and community focused commercial developers, a difference was found only in the number of messages posted to project mailing lists.



**Table 3: Mean Activity per Year on Mailing Lists by Class of Developer (superscripts indicate statistically different groups of means in each row)**

<b>Variable</b>	<b>Volunteer Mean</b>	<b>Product Focused Mean</b>	<b>Community Focused Mean</b>	<b>P Value</b>
Messages	39.04 <sup>A</sup>	87.10 <sup>B</sup>	135.80 <sup>C</sup>	<0.001
Threads Started	13.85 <sup>A</sup>	34.42 <sup>B</sup>	56.37 <sup>B</sup>	<0.001
Mailing Lists	0.37 <sup>A</sup>	0.68 <sup>B</sup>	0.53 <sup>B</sup>	<0.001
Extra Mailing Lists	0.20 <sup>A</sup>	0.23 <sup>A</sup>	0.20 <sup>A</sup>	0.400

Further content analysis of all email messages sent to public mailing lists identified elements that support information seeking behavior in the community – posting email addresses of contacts and providing pointers to web pages. The results were then aggregated by whether the author of the message was employed by a community or product focused firm. This analysis found that community focused developers posted references to email addresses 85% more frequently than product focused developers, and web addresses 140% more often. Both of these behaviors are pro-social and may help new community members become acquainted with the project and eventually contribute as developers.

Mid-level technical interactions on the Bugzilla bug tracking system may have similar affects in building community as posting to message to a mailing list. In particular, users are encouraged to post any bugs encountered to the Bugzilla. These bugs are periodically triaged by a group of community members who then assign the bugs to project developers. At the simplest level, each bug is given a small message form that allows developers to post messages which are sent back to the original submitter and any other individual with interest in the bug. Often times, developers post messages indicating that a bug has been verified as present, asking for more information, or provide a workaround for the user experiencing the bug. Individuals also may

submit patches to bugs that address the bug, a behavior that could be seen by a volunteer as taking a significant interest in their issue. When working with Bugzill, developers can mark bugs as “fixed”, indicating that the patch has been submitted and accepted to the upstream code base and that it will most likely be in the next major release. Finally, we can count the number of distinct projects a developer was active on within the Bugzilla system to get an idea of overall activity and also cross-reference this against activity in the source code repository to see where developers contribute to bug management but do not write code.

We collected each of the previously described metrics for every developer and used the same method as the mailing lists to average for the length of time the developer was active in the community. The metrics were aggregated by class of developer, and summarized in Table 4. Surprisingly, while commercial developers had a greater depth of activity, as measured by the number of comments, patches, and bugs fixed, they had the same relative amount of breadth in the system as volunteer developers. Contrary to our initial belief, when accounting for tenure in the project, product focused developers were active on significantly more projects and projects for which they had written no code than community focused developers. However, overall community focused developers were still more active in the community thanks to their longer average tenure.

**Table 4: Mean Activity per Year in Bug Reporting Database by Class of Developer (superscripts indicate statistically different groups of means in each row)**

Variable	Volunteer Mean	Product Focused Mean	Community Focused Mean	P Value
Comments	74.92 <sup>A</sup>	156.50 <sup>B</sup>	133.40 <sup>B</sup>	0.010
Patches	4.93 <sup>A</sup>	9.60 <sup>A</sup>	6.14 <sup>A</sup>	0.042
Bugs Fixed	1.44 <sup>A</sup>	3.80 <sup>B</sup>	7.30 <sup>B</sup>	<0.001
Projects	2.60 <sup>A</sup>	3.54 <sup>B</sup>	2.43 <sup>A</sup>	0.141

Extra Projects	1.56 <sup>A</sup>	2.09 <sup>B</sup>	1.11 <sup>A</sup>	0.556
----------------	-------------------	-------------------	-------------------	-------

Certain actions by commercial developers in the CVS code repository may also be construed as pro-social community oriented behavior. Such actions may not be overtly visible to the community at large, but volunteer developers would be very likely to see the number of commits a commercial developer makes as a sign of dedication to the community. Also, working on a variety of projects within the community shows that the developer has a greater interest in the overall health and well-being of the community. Analysis of the logs indicates that developers employed by community focused firms contribute more modifications to the source code repository and also are active on significantly more projects, as shown in Table 5. However, of note is that, when accounting for tenure in the community, the average number of commits per project does not exhibit a statistical difference between the two groups of commercial developers. The increased participation across projects mirrors the responses by many of the volunteer interviewees who believed the product focused firms worked only in narrow niches within the community and that community focused firms spread their effort across multiple projects.

**Table 5: Mean Activity per Year in CVS Repository by Class of Developer (superscripts indicate statistically different groups of means in each row)**

Variable	Volunteer Mean	Product Focused Mean	Community Focused Mean	P Value
CVS Commits	117.30 <sup>A</sup>	143.20 <sup>A</sup>	177.40 <sup>A</sup>	<0.001
CVS Projects	13.72 <sup>A</sup>	5.24 <sup>B</sup>	15.29 <sup>A</sup>	0.002

Although the interviewees agreed with the general classifications of firms in the community, most were unable to provide suggestions for what attributes outside of business model may differ

between the groups. The analysis just presented shows that employees of community focused firms are much more active in the broader community and across all three mediums; broad focused mailing lists that allow any individual to participate, the technical bug tracking system that focuses on specific issues within the software, and the highly technical source code repository that is open only to approved developers in the community.

This analysis shows that there are sometimes dramatic differences in the patterns of participation between volunteer, product focused, and community focused developers. While, in general, all commercial developers are more active in the community than volunteer developers, community focused developers are much more active and visible on community mailing lists and within the project source code. However, product focused developers tend to be more active in the moderately technical realm of bug tracking, indicating that such work may be critical to the success of their business model.

### ***4.3. Quantifying the Impact of Commercial Developers on Volunteer Participation***

The community makes it very easy for developers to start a new project, leading to a variety of projects that contain only small amounts of code, or are the work of only a single developer working on a very specific tool. To select projects that had a substantial community around them, we filtered the data selecting only projects with more than 20 developers, bugs filed in the Bugzilla bug tracking system, and mailing lists associated with the project. These requirements

yielded 14 projects from the community. The data was broken up into 8 week periods and at each period the number and identity of volunteer and commercial developers committing code during the period and the number of commits to the project during the period was recorded. Summary statistics and correlations can be seen in Table 6 and Table 7 below. While there are several variables that are highly correlated, indicating possible issues with multi-collinearity, each of the forthcoming models was analyzed and it was found to not be an issue. Additional metrics for project activity, such as total number of files modified and lines of code changed were also considered, but consistently were highly correlated ( $> 0.92$ ) with number of commits to the project.

**Table 6: Summary Statistics of Data Collected from 14 projects at 8 week intervals (601 total observations)**

	Mean	Median	Skewness	Kurtosis	Std Dev	Max	Min
$VolDevs_{i,t}$	4.01	3	1.24	0.96	3.94	18	0
Number of volunteer developers contributing code to project $i$ at time $t$							
$ComDevs_{i,t}$	3.57	2	2.19	4.70	4.87	26	0
Number of commercial developers contributing code to project $i$ at time $t$							
$ComDevs_{CF,i,t}$	1.12	1	2.16	4.77	1.66	9	0
Number of commercial developers from community focused firms contributing code to project $i$ at time $t$							
$ComDevs_{PF,i,t}$	2.65	1	2.84	8.22	4.49	26	0
Number of commercial developers from product focused firms contributing code to project $i$ at time $t$							
$Commits_{i,t}$	114.97	46	2.98	11.54	175.64	1407	0
Number of commits made by all developers to project $i$ at time $t$							
Observations per Project, $N$	42.92	49	-1.66	1.74	12.72	53	14

**Table 7: Correlations of Data Collected at Project Level after  $\log_2$  transformations.**

	$VolDevs_t$	$VolDevs_{t-1}$	$ComDevs_{t-1}$	$ComDevs_{CF,t-1}$	$ComDevs_{PF,t-1}$	$Commits_{t-1}$
$VolDevs_t$	1.0000					
$VolDevs_{t-1}$	0.8263	1.0000				

$ComDevs_{t-1}$	0.3755	0.3921	1.0000			
$ComDevs_{CF,t-1}$	0.4346	0.4258	0.6272	1.0000		
$ComDevs_{PF,t-1}$	0.2733	0.2773	0.9272	0.3555	1.0000	
$Commits_{t-1}$	0.6655	0.7331	0.6400	0.4208	0.5504	1.0000

The distribution of the number of commercial and volunteer users is highly skewed toward the lower end of the range and can be approximated with a log-normal distribution. To a lesser degree the distributions of the number of community focused developers, product focused developers, and commits are also skewed. To accommodate for this in our models, the logarithm (base 2) of these variables are used.

We begin with a simple model that predicts the number of volunteer developers contributing source code to project  $i$  at time  $t$  as a function of the log transformed number of volunteer developers, commercial developers, and commits at time  $t-1$ . A simple linear regression is not appropriate for such a model, because of lack of independence between observations within a project and the differing levels of inherent attractiveness to volunteer developers between projects. To account for this difference, a multi-level model was utilized that allows that intercept, corresponding to base project attractiveness, to vary between projects. Within the model shown in Equation 1 this difference is represented by a constant error term,  $v_i$ , for all observations of project  $i$ .

**Equation 1:**  $VolDevs_{i,t} = \beta_0 + \beta_1 VolDevs_{i,t-1} + \beta_2 ComDevs_{i,t-1} + \beta_3 Commits_{i,t-1} + v_i + \epsilon_{i,t}$

The results of the model can be seen in Table 8 and show that an increase in the number of commercial developers results in a small increase in the number of volunteer developers when controlling for the number of volunteers already active in the project and overall project activity. We note, however, while the effect is significant at the  $p < 0.05$  level, the magnitude of the effect is quite small – a doubling of commercial developers working on a project yields only an 8% increase in the number of volunteer developers. This small effect indicates that volunteers tolerate commercial developers working in their communities, but their presence plays little role overall in increasing the number of volunteer developers. Thus, we conclude there is support for hypothesis 1 and can reject hypothesis 2.

**Table 8: Hypothesis 1 and 2 – Multi-Level Regression Coefficients Predicting Number of Volunteer Developers by Project**

<b>Variable</b>	<b>Estimate</b>	<b>Std Err</b>	<b>P Value</b>
Intercept	0.5643	0.1397	0.001
$VolDevs_{t-1}$	0.4562	0.0442	<.001
$ComDevs_{t-1}$	0.0817	0.0389	0.036
$Commits_{t-1}$	0.0601	0.0242	0.013

As we have shown there is a marked difference between the methods and magnitudes of participation of commercial developers when divided by whether their firm is product focused or community focused. In Equation 2 we expand on the model to differentiate between participation of developers for community focused firms,  $ComDevs_{CF,t-1}$ , and product focused firms  $ComDevs_{PF,t-1}$ . The same data are used with the new multi-level longitudinal model, with the regression coefficients presented in Table 9.

**Equation 2:**

$$VolDevs_{i,t} = \beta_0 + \beta_1 VolDevs_{i,t-1} + \beta_2 ComDevs_{CF,i,t-1} + \beta_3 ComDevs_{PF,i,t-1} + \beta_4 Commits_{i,t-1} + v_i + \epsilon_{i,t}$$

**Table 9: Hypothesis 3 – Multi-Level Regression Coefficients Predicting Number of Volunteer Developers by Project Broken Up By Firm Model**

Variable	Estimate	Std Err	P Value
Intercept	0.6032	0.1381	<.001
$VolDevs_{t-1}$	0.4212	0.0443	<.001
$ComDevs_{CF,t-1}$	0.2050	0.0432	<.001
$ComDevs_{PF,t-1}$	-0.0433	0.0388	0.264
$Commits_{t-1}$	0.0711	0.0234	0.003

These results indicate a significant difference in behavior by business model of the firm. Participation by developers from community focused firms tends to have a moderate positive impact on the number of volunteer developers, while participation by developers of product focused firms has no statistically significant impact. This difference between developers at community and product focused firms lends support for hypothesis 2.

To better identify if the observed difference in the relations of community focused and product focused developers to volunteer developers was due to factors previously observed as differentiation factors between the groups, a mediating analysis was performed that included the metrics found to have statistically significant variation between community and product focused developers. For each of the eight week periods the total number of messages written by that projects developers across all mailing lists, the number of projects the developers worked on bugs for, and the number of additional projects the developers wrote code for were calculated.



Descriptive statistics of these mediating variables can be seen in Table 10 below. Correlations of the variables can be seen in Table 11. Although there are several points of high correlation, analysis found that the maximum variance inflation was 3.8, concluding that multicollinearity was not an issue in the data.

**Table 10: Summary statistics of mediating variables collected for each project at 8 week time intervals**

	Mean	Median	Skewness	Kurtosis	Std Dev	Max	Min
<i>BugProjects<sub>i,t</sub></i>	13.54	3.00	2.03	4.99	20.03	138	0
Number of bugzilla projects for commercial developers contributing code to project <i>i</i> at time <i>t</i>							
<i>DevMailMessages<sub>i,t</sub></i>	250.22	120	2.48	7.87	347.61	2405	0
Number of messages posted to all community mailing lists for developers contributing code to project <i>i</i> at time <i>t</i>							
<i>CVSProjects<sub>i,t</sub></i>	37.20	36.00	0.65	0.07	31.48	189	0
Number of total projects for commercial developers contributing code to project <i>i</i> at time <i>t</i>							

**Table 11: Correlations of mediating variables**

	<i>VolDevs<sub>i,t-1</sub></i>	<i>ComDevs<sub>CF,i,t-1</sub></i>	<i>ComDevs<sub>PF,i,t-1</sub></i>	<i>Commits<sub>i,t-1</sub></i>	<i>BugProjects<sub>i,t-1</sub></i>	<i>DevMailMessages<sub>i,t-1</sub></i>
<i>BugProjects<sub>i,t-1</sub></i>	0.2905	0.2747	0.3401	0.3711	1.0000	
<i>DevMailMessages<sub>i,t-1</sub></i>	0.3124	0.5831	0.7014	0.4759	0.3596	1.0000
<i>CVSProjects<sub>i,t-1</sub></i>	0.4839	0.4240	0.5580	0.3945	0.3660	0.5144

**Equation 3:**

$$VolDevs_{i,t} = \beta_0 + \beta_1 VolDevs_{i,t-1} + \beta_2 ComDevs_{CF,i,t-1} + \beta_3 ComDevs_{PF,i,t-1} + \beta_4 Commits_{i,t-1} + \beta_5 BugProjects_{i,t-1} + \beta_6 DevMailMessages_{i,t-1} + \beta_7 CVSProjects_{i,t-1} + v_i + \epsilon_{i,t}$$

The addition of the control variables produces the model seen in Equation 3. Evaluating the model, as seen in Table 12 shows that mediating variables, which had previously been identified as differences between community and product focused developers, do not cause a significant change in the original regression variables. The pattern remains that community focused

developers have a positive relation to the number of volunteer users and product focused developers have no statistically significant relation. We do, however, note that the direction of the coefficients for  $BugProjects_{i,t-1}$  and  $CVSProjects_{i,t}$  are in the opposite direction of what would be most likely predicted – indicating that the presence of developers who are active on a number of different projects is related to fewer volunteer developers in the future.

**Table 12: Model with addition of mediating variables**

Variable	Estimate	Std Err	P Value
Intercept	0.6122	0.1387	<.001
$VolDevs_{i,t-1}$	0.4527	0.0471	<.001
$ComDevs_{CF,i,t-1}$	0.2165	0.0453	<.001
$ComDevs_{PF,i,t-1}$	-0.0177	0.0437	0.685
$Commits_{i,t-1}$	0.0939	0.0247	<.001
$BugProjects_{i,t-1}$	-0.0030	0.0001	0.046
$DevMailMessages_{i,t-1}$	0.00005	0.0001	0.692
$CVSProjects_{i,t}$	-0.0028	0.0012	0.025

Open Source projects may be comprised of over 100,000 lines of code, rivaling the complexity of large scale commercial software development projects. When projects attain such size and have so many developers collaborating, developers form sub-groups to work on modules within the project. These modules typically focus on a particular domain of the project, for example, a developer working on an email client may specialize in working on providing encryption support for email, which may be spread across dozens of different files. While language specific methods exist to specify explicit module structures and infer implicit structure through static

source code analysis, the code in the GNOME project is written in a variety of different languages and programming language specific methods are impractical. As an alternative, we propose using social network analytic clustering methods on the network of source code to determine modules within the project. The CONCOR algorithm was used to produce eight clusters per project as it requires no additional information beyond link information when generating the groupings. Functionally, the computation views the network of files as a matrix and then attempts to rearrange the rows and columns of the matrix so entities that are structurally equivalent, meaning they link to the same set of other files, are grouped together (Boorman & White, 1976). While a variety of unsupervised clustering exist that determine an optimal number of clusters (Newman, 2004) these methods often produce more clusters than practical, leaving many clusters with only a single active developer.

Our method of inferring code modules within a project utilizes a network structure where nodes in the network are files, and edges are added between nodes if they were committed back to the central repository in a single commit (Cataldo, Patrick Wagstrom, James Herbsleb, & Kathleen Carley, 2006). Community work practices encourage developers to commit changes back feature by feature, rather than waiting and sending all, possibly unrelated, changes for a whole week or more back as a single commit. In this way, we obtain a network where highly related files are densely clustered together. The CONCOR algorithm was run on this network for each project and configured to generate 8 clusters, each representing a module within the project, as a compromise value that typically yielded a realistic number of developers in each cluster. The same summary statistics shown in Table 6 were generated for each module in each time period,

yielding a total of 6360 observations. It should be noted that this generates a very conservative clustering estimate as many of the larger projects have more than eight distinct components. By using this strategy we are careful to avoid type 1 error, while focusing less on type 2 error.

The analysis at the project level was then replicated with the new data based on the clusters within each project. For this analysis, a new subscript  $j$  is added to the model indicating the cluster within project  $i$ . In addition, a new error term is added to the model,  $\zeta_{i,j}$ , which is the correction for the attractiveness of module  $j$  within project  $i$ .

**Equation 4:**

$$VolDevs_{i,j,t} = \beta_0 + \beta_1 VolDevs_{i,j,t-1} + \beta_2 ComDevs_{i,j,t-1} + \beta_3 Commits_{i,j,t-1} + v_i + \zeta_{i,j} + \varepsilon_{i,j,t}$$

**Table 13: Hypothesis 4 – Testing for issues of cognitive complexity through the analysis of effect of commercial developers at the module level**

Variable	Estimate	Std Err	P Value
Intercept	0.2341	0.0802	0.012
$VolDevs_{i,t-1}$	0.3424	0.0177	<.001
$ComDevs_{i,t-1}$	0.0363	0.0165	0.027
$Commits_{i,t-1}$	0.1123	0.0094	<.001

The new model testing for cognitive complexity issues was analyzed and the results can be seen in Table 13. We find sufficient evidence to justify the addition of the module level analysis as measured by the improvement in the fit of the model through the -2Log Likelihood statistic ( $p < 0.001$ ). This indicates that the significant variation exists not only between projects, but also between modules within a project. We also note evidence for this within project variation when observing the decrease in the magnitude of the effects in the final model versus the model

without intra-project variation found in Table 8 and Table 9 as much of the previous values for the effects has been shifted to the new module level random variable,  $\zeta_{i,j}$ . An examination of the results indicate that the presence of commercial developers has a positive and significant effect on the number of volunteer developers even at the module level, which is the opposite of what Hypothesis 4 predicted. Therefore we reject the argument that the increased cognitive complexity associated with colocated commercial developers drives volunteer developers away from project modules.

## 5. Discussion

This work shows that participation by commercial firms can have a positive impact on the participation of volunteer developers in most situations. At an overall level, it was found that benefits gained from overall momentum and being identified with a successful project, as proposed in Hypothesis 1, outweigh any negative effects from heterogeneity in the community proposed in Hypothesis 2. However, the relatively small magnitude of these effects, especially relative to effect of existing volunteer developers, indicates that not all volunteers believe the benefits of commercial developers outweigh the negative issues of workflow, norms, and values. This different echoes what has been observed in the community where some projects are founded under very utilitarian goals and seek only to create quality software, which would be amenable to commercial involvement, while other projects focus on the philosophical ideals for using Open Source software and would be more sensitive to the heterogeneity introduced by commercial developers. The GNOME project straddles this line, with some projects being very utilitarian while others are clearly founded and ran with free software ideals in mind.

Another major contribution is the identification and validation of two distinct classes of business models for firms participating in Open Source communities. These models closely aligned themselves with broad community values or a desire to focus on a single project in the community. As expected, those community focused firms were found to be highly visible on project mailing lists and wrote code for more projects than developers from product focused firms. Yet, contrary to our assumptions, product focused firms were more active in the moderately technical domain of bug fixing. Interestingly, the different business models were found to have differing impacts on volunteer developers – with community focused developers have a statistically significant impact while product focused developers had very little. Given the predominant view of the interviewees that community focused firms were more aligned with the values and norms of the community, this supports the hypothesis that the communities are sensitive to the values and norms of commercial participations and should provoke caution for firms wishing to participate in Open Source projects.

Finally, we analyzed whether or not the collocation of commercial developers and the possible increase in cognitive complexity of projects as a result, had an effect on volunteer participation. Much to our surprise it was found that this was not the case. Thus, either commercial firms are very good about ensuring that their participation does not increase the cognitive load of the developers, or the self-selection process in Open Source that requires developers be able to figure the project source code with little assistance, draws developers who are able to compensate for such situations.

Our research does not clearly indicate that commercial firms should be able to take the work of a community and productize without regard to the community standards. Rather, we found that in spite of the heavy participation of product focused firms, the magnitude of the positive relation between their participation and the level volunteer participation relatively low. Given volunteer developers frequent negative comments toward product focused firms in the interviews process, this is quite worrisome. When working with an open source community, the firms should always take care to ensure that they appear as good citizens and contribute to the community, reinforcing the sense of community amongst the participants, rather than having volunteers view their participation as exploitation.

From a volunteer perspective, it could be that volunteers view all commercial developers as beneficial to the community and are more likely to participate, but this effect is mediated by negative effects in the code of commercial developers that we were unable to observe. Alternatively, volunteers may take a selfish view and only have interest in functionality they use. In such a selfish view, many of the features added by product focused commercial developers would be ignored by volunteers because they are targeted for different markets, such as enterprise installations, and require infrastructure or have use cases that volunteers have little interest in. We also note that in a federated community, such as GNOME, the actions of commercial firms within a project are visible to volunteers even if they have no involvement with the project. This broader recognition of commercial firm actions by volunteers may result

in negative events with product focused firms in a project having repercussions across the entire community.

At a more broad level, all firms involved in Open Source may wish to evaluate how and where they participate in Open Source communities. In particular, we found that broad pro-social community building behaviors, such as being active on a variety of projects, fixing bugs in the software, and posting messages that assist with information seeking were associated with firms that had the strongest relation to volunteer developers. Firms may wish to dedicate a portion of engineer time to ensuring that questions raised on project mailing lists are responded to promptly, or even consider employing a community liaison to provide assistance to new members of the community. Typically many Open Source communities have focused almost exclusively on the code for the project, and less on the social and management roles in a project. Recently, however, as many projects have grown and seen wider adoption, these communities have adopted more traditional structures and often are in need of organizational resources that a commercially sponsored community liaison could provide. The GNOME community has seen this evolution happen, several members of the GNOME Foundation Board of Directors no longer write code, but spend their time managing the project and ensuring that everything flows smoothly.

No matter what the reasons for the increased success of community focused firms in attracting and retaining volunteer developers, it appears that more and more firms will adopt Open Source projects, practices, and contribute to communities in the near future. We have seen that in this



case, the dual worlds of volunteer and commercial can co-exist in an Open Source project with little danger of the commercial firm dramatically damaging the incumbent volunteers. Going forward, understanding the methods by which these firms attract and retain volunteer developers is an open research question that will yield great benefits for firms seeking to utilize this revolutionary software development model.

### **5.1. Bibliography**

- Boorman, S. A., & White, H. C. (1976). Social structure from multiple networks. ii. role structures, *American Journal of Sociology*, 81(6), 1384-1446.
- Cataldo, M., Patrick Wagstrom, James Herbsleb, & Kathleen Carley. (2006). Identification of coordination requirements: implications for the design of collaboration and awareness tools In (pp. 353-362). Banff, Alberta, Canada: ACM Press.
- Fogel, K. (2005). *Producing open source software*, 279. Sebastapol, CA: O'Reilly & Associates.
- German, D. (2004). The gnome project: a case study of open source, global software development, *Software Process: Improvement and Practice*, 8(4), 201-215. doi: 10.1002/spip.189.
- Koch, S., & Schneider, G. (2002). Effort, co-operation and co-ordination in an open source software project: gnome, *Information Systems Journal*, 12(1), 27-42. doi: 10.1046/j.1365-2575.2002.00110.x.
- Krishnamurthy, S. (2005). An analysis of open source business models In J. Feller, B. Fitzgerald, S. Hissam, & K. R. Lakhani (Eds.), *Perspectives on Free and Open Source Software* (1). Cambridge, MA: MIT Press.
- Lakhani, K., & Wolf, R. (2005). Why hackers do what they do: understanding motivation and effort in free/open source software projects In Joseph Feller, Brian Fitzgerald, Scott Hissam, & Karim R. Lakhani (Eds.), *Perspectives on Free and Open Source Software*. Cambridge, MA: MIT Press.
- Lakhani, K. R., & von Hippel, E. (2003). How open source software works: "free" user-to-user assistance, *Research Policy*, 32(6), 923-943. doi: 10.1016/S0048-7333(02)00095-1.
- Lerner, J., & Tirole, J. (2002). Some simple economics of open source, *Journal of Industrial Economics*, 50(2), 197-234. doi: 10.1111/1467-6451.00174.

- MacCormack, A., Rusnak, J., & Baldwin, C. Y. (2006). Exploring the structure of complex software designs: an empirical study of open source and proprietary code, *Management Science*, 52(7), 1015-1030. doi: 10.1287/mnsc.1060.0552.
- Newman, M. (2004). Detecting community structure in networks, *The European Physical Journal B*, 38(2), 321-330. doi: 10.1140/epjb/e2004-00124-y.
- Parnas, D. (1972). On the criteria to be used in decomposing systems into modules, *Communications of the ACM*, 15(12), 1053-1058.
- Raymond, E. S. (1999). *The cathedral and the bazaar*. Sebastapol, CA: O'Reilly & Associates.
- Stewart, K. J., Ammeter, A. P., & Maurping, L. M. (2006). Impacts of license choice and organizational sponsorship on users interest and development activity in open source software projects, *Information Systems Research*, 17(2), 126-144. doi: 10.1287/isre.1060.0082.
- Tiemann, M. (1999). Future of cygnus solutions: an entrepreneur's account In C. DiBona, S. Ockman, & M. Stone (Eds.), *Open Sources: Voices from the Open Source Revolution* (pp. 71-91). Sebastapol, CA: O'Reilly Media, Inc.
- Young, R. (1999). Giving it away: how red hat software stumbled across a new economic model and helped improve an industry In C. DiBona, S. Ockman, & M. Stone (Eds.), *Open Sources: Voices from the Open Source Revolution* (pp. 113-126). Sebastapol, CA: O'Reilly Media, Inc.



# Understanding Communication, Coordination, and Congruence at an Individual Level By Harnessing Baseball Statistics

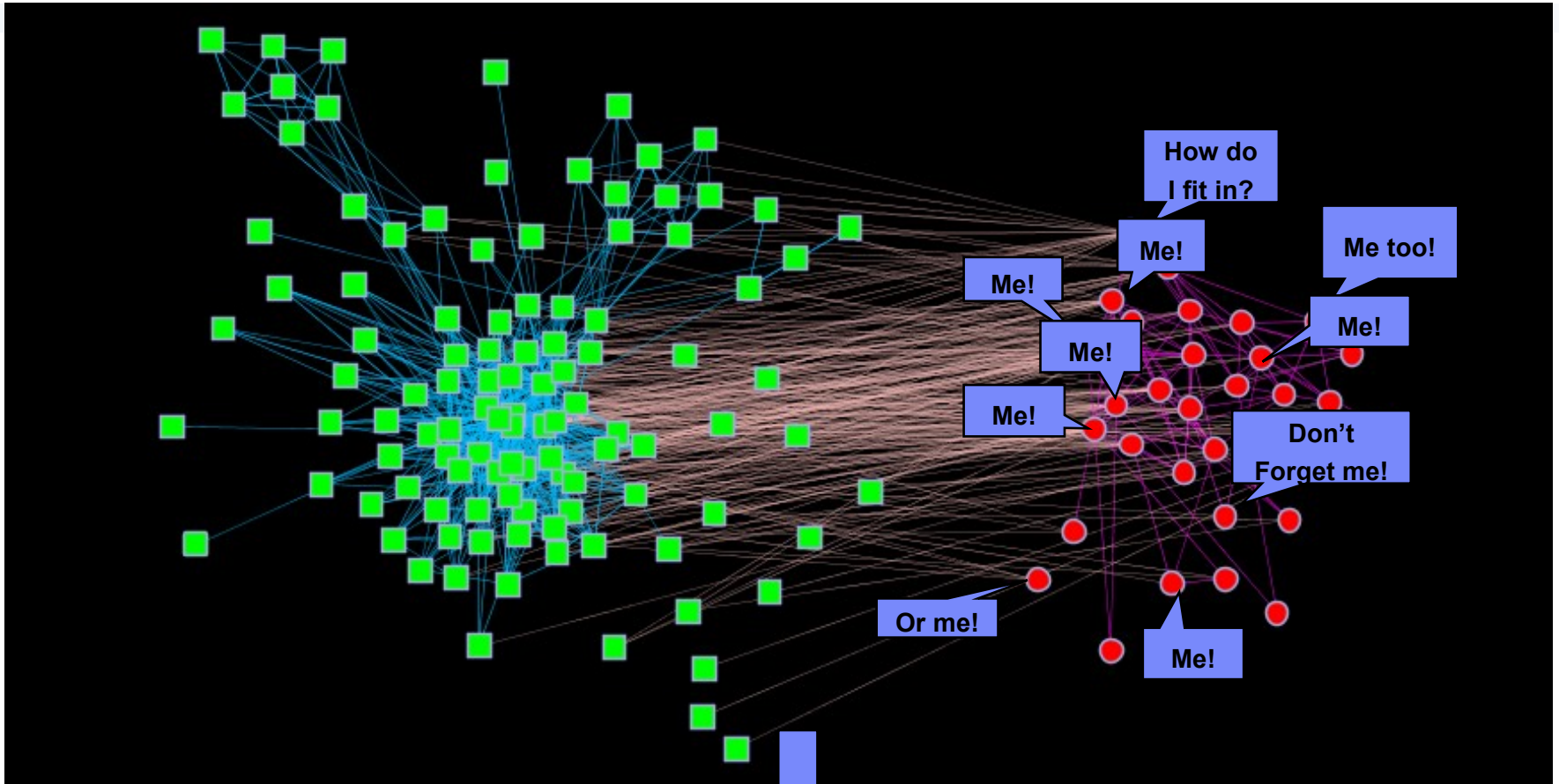
Patrick Wagstrom

patrick@wagstrom.net

Joint work with Mary Helander, Kate Ehrlich, and Clay Williams



# Congruence In a Nutshell



0.24

# Goal

- Take congruence down to an individual level
- Develop a set of metrics based on congruence that:
  - Have meaning to an individual
  - Are related to performance

# Outline

- Introduction
- Brief Introduction to Baseball
- Linking Baseball to Social Networks
- Description of Study
- Presentation of Results
- Potential Shortcomings

# Baseball Quickies



- Team sport, 9 players on each side
- A manager sits off-field and calls plays
- Primarily a struggle between a batter and pitcher
- Batter attempts to hit the ball and get on base to score
- 3 failed attempts to hit the ball is a strikeout
- Baseball lives on statistics

# Baseball and Congruence

- In measuring congruence, we have a coordination requirements network, and actual coordination network
- Congruence is the percentage of links in coordination requirements that are also present in actual coordination
- Thus:
  - An edge in the coordination requirements network is an opportunity to communicate
  - We term this an “At Bat”



# Hits and Strikeouts

- If the corresponding edge is present in the actual coordination network, we have a “hit”
- If the edge is missing in the actual coordination network, we have a “strikeout”
- We can also calculate batting average:

$$\frac{hits}{atbats}$$

# Edge Weights and Home Runs

- Work by Helander et. al. provides meaning for weights in coordination requirements matrix
- We say a **hit** is worth the number of **bases** of the corresponding edge in the coordination matrix
- A strikeout results in a number of runners being **left on base** equal to the corresponding edge in the coordination matrix

# Slugging Percentage

- In baseball:  $\frac{\sum bases}{atbats}$
- Our **slugging percentage** is the same
- We also create **anti slugging percentage**

$$\frac{\sum leftOnBase}{atbats}$$

# Software Development is a lot like Baseball...

- Team sports that require lots of coordination
- Managers shift lineups and player positions in the middle of the game
- Success and failure determined by a single play
- Frequently enter extra innings
- Even though a team sport, good pitchers and hitters still make a huge difference
- Physical fitness is optional for the game

# Study Population

- Distributed, open-source software development project
- 154 Individuals contributed code
- Usually about 10 core developers
- Five Years of data
  - Email Messages
  - Bug Reports
  - Source Code Modifications
- Data aggregated at approximately six month intervals

The Official Site of Major League Baseball: News: Game Wrapup - Mozilla Firefox: IBM Edition

File Edit View History Bookmarks Tools Help

http://mlb.mlb.com/news/wrap.jsp?ymd=20070819&content\_id=2158189

ML IBM Business Transfo... IBM Standard Softwar... IT Help Central Join World Community... Windows Marketplace IBM

### Texas blanked by Santana

By T.R. Sullivan / MLB.com

Despite walking five batters, Kevin Millwood gave up just four hits and one earned run in eight innings in the series finale at the Metrodome. Sammy Sosa got the Rangers' only two hits.

### Twins 1, Rangers 0




### Twins ride ace to victory

By Kelly Thesier / MLB.com

Johan Santana set a new club record with 17 strikeouts, fanning every Rangers hitter at least once and giving up two hits in eight shutout innings. Michael Cuddyer put the Twins' only run.

**IMPACT PLAY OF THE GAME**



**Patrick Wagstrom Sends an email to Mary on Tuesday**

**SPR Resolution Time**

3.2 Days **BEFORE PLAY** ↓ 2.1 Days **AFTER PLAY**

SEE MORE IMPACT PLAYS

MLB.TV | Condensed Game | Highlight Reel

AUDIO

KRRL - Gameday Audio

17 hitters on Sunday. (Paul Battaglia/AP)

Series at a glance:

- Fri. 08/17: MIN 2, TEX 1
- Sat. 08/18: TEX 5, MIN 0
- Sun. 08/19: MIN 1, TEX 0

MLB.TV | Condensed Game | Highlight Reel

AUDIO

TRN - Gameday Audio

Santana's 17th strikeout

**IMPACT PLAY OF THE GAME**



**Joe Nathan Strikes out batter Top of 9th**

**Win Probability**

78% **BEFORE PLAY** ↑ 90% **AFTER PLAY**

SEE MORE IMPACT PLAYS

**PROTRADE LIVE**

Where fans follow games as each team's chances of winning in real time.

Texas (54-69)  
Lost 1

August 19, 2007

	1	2	3	4	5	6	7	8	9	R	H	E
Texas	0	0	0	0	0	0	0	0	0	0	2	0
Minnesota	0	1	0	0	0	0	0	0	X	1	4	1

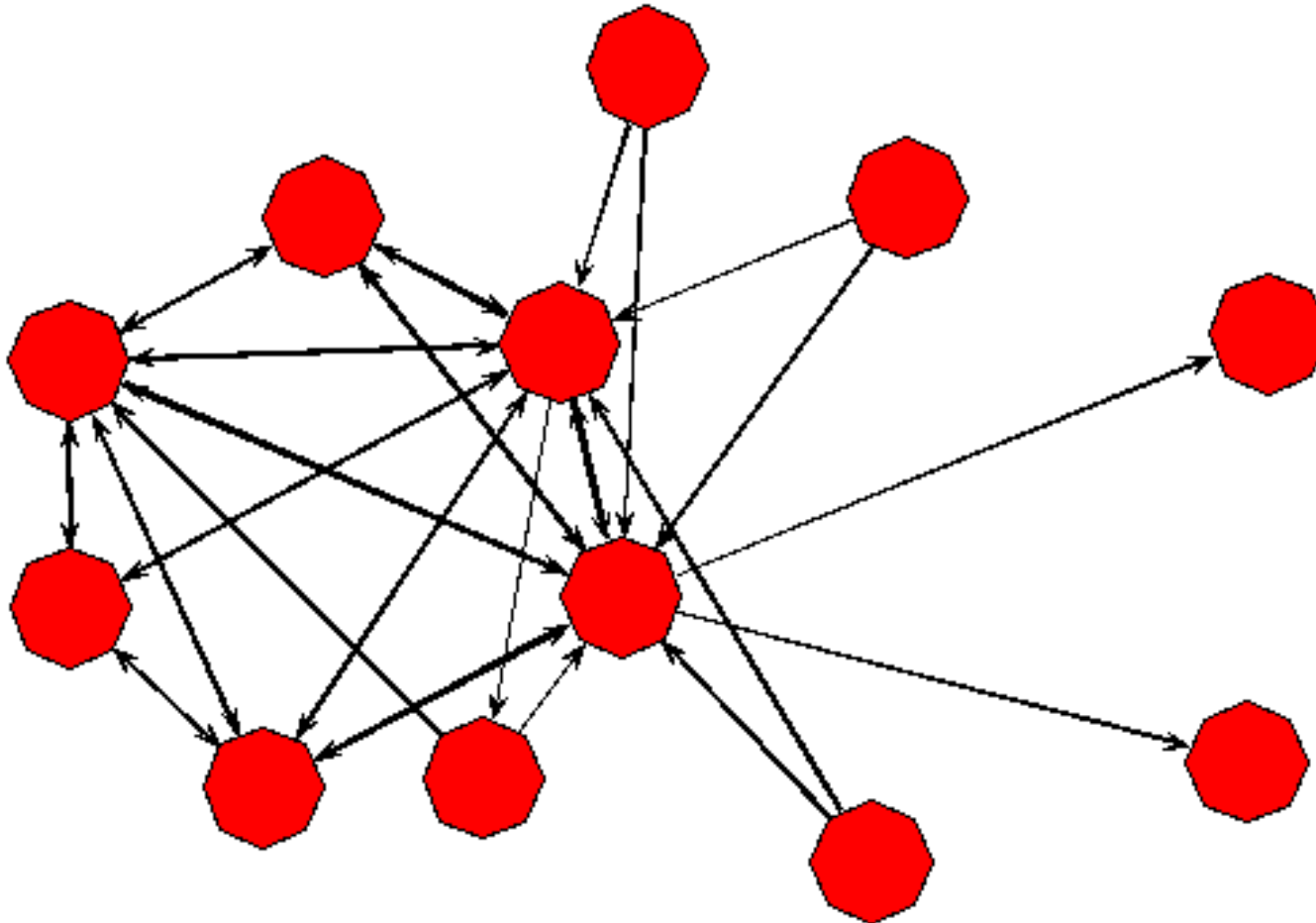
Minnesota (62-61)  
Won 1

Transferring data from gdx.mlb.com...

zotero



# Visualizing Congruence



# Modeling Task Performance

- Collected information on 3079 bugs
- Dependent variable:  $\log(\text{time for bug resolution})$
- Best model was sought out and found to be:

$$\log(\text{resTime}) = \beta_1 \text{numDevs} + \beta_2 \text{atBats} + \beta_3 \text{slgPct} + \varepsilon$$

- Model Results

	Estimate	Std. Error	t value	Pr(> t )	
(Intercept)	5.952134	1.250519	4.760	< .0001	***
numDevs	1.032183	0.196526	5.252	< .0001	***
atBats	-0.847243	0.135770	-6.240	< .0001	***
slgPct	0.011765	0.001818	6.470	< .0001	***



# Potential Issues

- Sparse data causes issues
  - Many developers don't have congruence metrics each period
- Identifying software dependencies can be tricky
  - Different methods yield radically different results
- High correlations between resulting metrics
- Don't have enough of the other control variables to predict resolution time
- Very small data set in a very specific field

# Conclusions and Thanks!

- Successfully created a set of metrics relating individuals to congruence
- Found that **batting average** doesn't mean much, but **slugging percentage** is significant
- Unclear why **slugging percentage**, which is related to high congruence, raises the time to completion

This work originated at CMU, was brought to IBM, and now continues at CMU where it is being generously sponsored by the National Science Foundation and the Office of Naval Research.

