# CS680: Distributed Software Development

# Project Report - Spring Term 2010

**Team Members:**

1. First Name: Jordan Last Name: Osecki E-mail: [Jordan.Osecki@gmail.com](mailto:Jordan.Osecki@gmail.com)

2. First Name: Sammie Last Name: Stahlback E-mail: [Sws28@drexel.edu](mailto:Sws28@drexel.edu)

3. First Name: Michael Last Name: Kim E-mail: [Mikim303@yahoo.com](mailto:Mikim303@yahoo.com)

**Project Name:**

* Analysis Team – Core/Periphery Analysis of MySQL from

Data Mining of an Open Source Project

# Abstract

The communication, bug tracking, and source code repositories of a project can shed light on to the communication and other functions which happen to a development team throughout the course of a project. These repositories are blueprints which enshrine all characteristics of the project, the development team, and all other aspects that typically go into software production.

Studies on the communication repositories in particular can highlight the actual communication patterns within the software development team itself. For example, is the work being done by just one person, by a small group, or by everyone equally? Are their individuals on the outside which all go to for advice? All of these types of patterns and more can be discovered through the analysis of a project’s communications repository.

This paper mines data from the mailing list repositories of an open source project, MySQL, in an attempt to learn the “blueprint” of the software development team throughout different points in time (namely, their software’s release dates). The mailing list data is put into an organized database and then re-organized and sent through third-party software that specializes in software project community structure analysis, in an attempt to see the results of this product.

Open source projects, such as MySQL, typically have the blueprint of a core/periphery structure, meaning that there is a small core which does most of the work and a large periphery which does very little (but meaningful) work. This paper studies the mailing lists to determine if MySQL follows this core/periphery structure and to see how it changes over time.

This study has serious implications not just for open source software, but also for any kind of distributed software development. Open source and distributed software development (DSD) has a lot in common, with open source being a form of DSD which is pretty successful. Knowing what makes open source projects successful, one of those things being the core/periphery structure of users, would be a valuable thing that could be applied to DSD projects of all shapes and sizes in order to improve the overall quality of the software that is produced.

# Overview

As mentioned in the abstract, there are many distributed software development (DSD) issues and implications that are embraced by this study of the core/periphery structure of MySQL and other open source projects. Some of these issues include communications and communication blueprints, repositories and tools used to communicate remotely, mining repositories, the study of open source versus DSD, and the core/periphery structure embodied by most open source projects which seems to be so successful.

Communications and their blueprints are important. One of the biggest challenges of DSD is to be able to effectively communicate with your fellow co-workers while working in remote locations. This causes issues because software projects are full of dependencies which need to be addressed, and this is much harder to do when the two or more people responsible are not geographically located together.

Because of the communication gaps which are evident and can be seen when a blueprint of the organization structure is taken, repositories and tools that are used to communicate remotely become of the utmost importance. With face-to-face communication and formal meetings lacking in DSD, most communication occurs over computers in recorded media such as e-mail, message boards, mailing lists, bug tracking, source control, etc. These repositories become the main way to interact with people with whom you have to discuss coding dependencies with on a daily basis.

Since all of these tools and repositories become so important, mining their contents can be very valuable to observe trends in projects in general, in projects at the company, and to steer that specific project if there are problems with it. One such trend which can be observed is what the communication structure of the DSD development team is like and if it is hindering things. There is no perfect formula to create a perfectly efficient DSD team, however, so this task is hard.

Therefore, the comparison is made between open source and DSD. This is because open source has many things in common with DSD, such as workers working remotely with no real communication other than the same used by DSD workers. In addition, they use the same types of repositories and are still able to perform great work with high quality software resulting.

Studies have shown that open source software typically produces high quality software with its developers organized in a core/periphery structure, described in the “Abstract” section. This structure allows for gatekeepers to the code who write most of the code and approve the rest of it, particularly to their expertise sections. In addition, the entire periphery acts as bug reporters to aid in making the software the best it can be.

The basis of this report was to explore whether MySQL, a very successful open source project, has a core/periphery structure like it is expected to have. If it does, this can be studied by software producers of all kinds, public and private, open source and commercial, to model it for DSD situations. In addition, the communication blueprint is going to be taken over the course of many releases, which will allow researchers to see the changes to the blueprint over time, correlated to how the software was performing at that time. Finally, this report will explain how the mailing list repositories of MySQL were parsed and then re-stored, which will shed light on the best practices of mining repositories and also the movement for common formats of repositories.

The main objectives that the project was supposed to achieve include the following: re-creating the entire MySQL community and social network for each major release in the last few years; compiling and studying metrics on the core/periphery structure of this community over the same time frame; and a production of charts showing trends for these two features over the same set of releases.

To pursue these objectives, the researchers will first mine the mailing list of the MySQL project, put it into a common format in a database, and then use this database to construct whatever interface formats are necessary to feed third-party tools which will be used to analyze and re-create the MySQL social network and also obtain core/periphery metrics at each release date the team is interested in. More information on the pursuit of these objectives can be found in the “Approach” section of this paper.

# Background (1 page max – excluding figures)

TODO

Provide any background information that is useful to better understand the context in which you developed your project. – Repository context, transformation of data, discuss class, open source aspect

May have to do with the description and characteristics of the project you have worked with. – describe mysql (source)

The techniques that were used. – Mining in Perl, gathering in Java, use of two products

The environments and tools that you employed. – PostgreQL, Mysql, Java, perl, NodeXL, UCINET

If there is any related work that you think is relevant to the work you have done and should be referenced, discuss it here. - Borgatti

# Approach (2 pages max – excluding figures)

Describe in detail the work you have done, as a team or individuals. – Step by step using the info below

The methods you have employed to achieve the intended results. – Go into a few steps in detail, including the schema used and how releases were determined

Detail whatever you have built and how it works – detail the code, point to the repository (source)

Discuss here also any issue of process and collaboration if relevant. – MINING ISSUES, ISSUE with NodeXL (source)

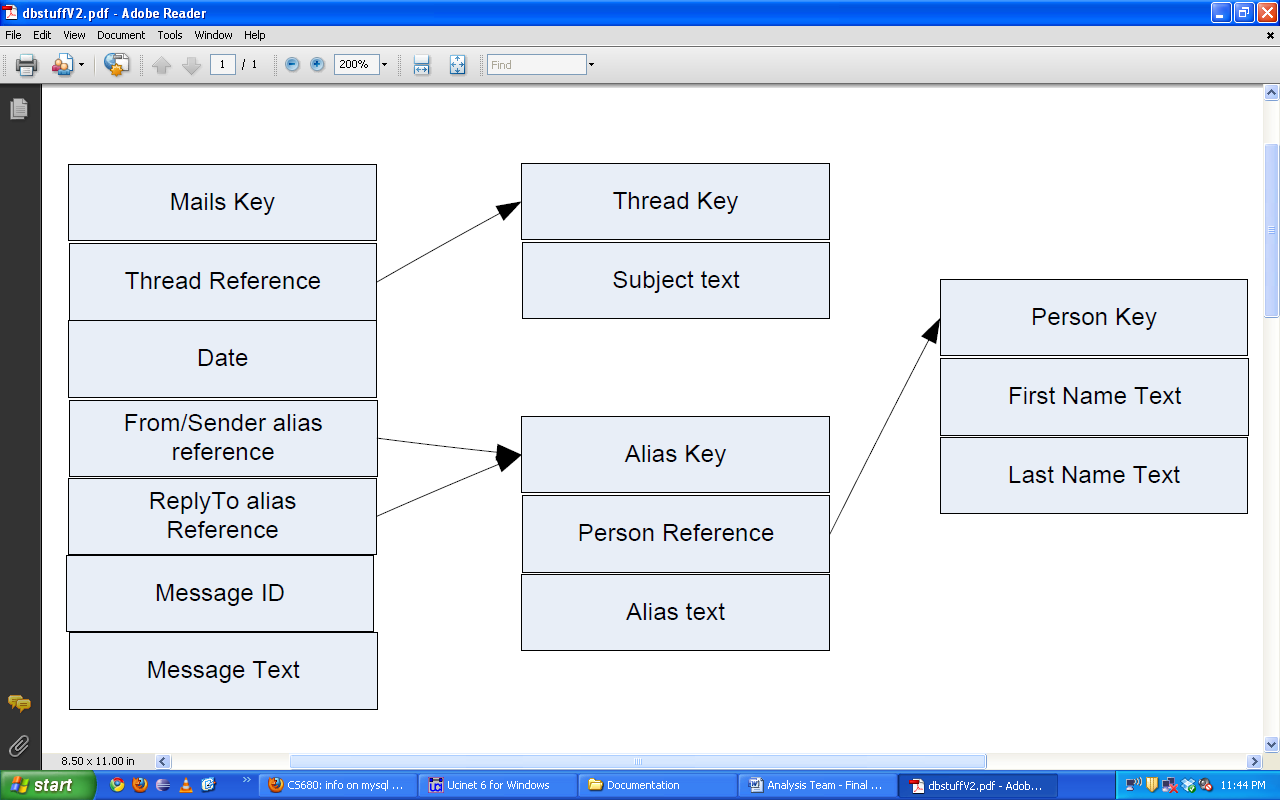


Figure : PostgreSQL Database Schema

High level overview of project tasks:

1. Collect data from (<http://lists.mysql.com>) NEWS GROUPS, CHANGE LOG
2. Create Postgres/PostgreSQL database using schema from sql.tar.gz
3. Process data collected from (http://lists.mysql.com)
4. Populate Postgres/PostgreSQL database - SCHEMA
5. Analyze communication (reference previous papers discussed in class)
6. Analyze one particular MySql Database Server:  
       - Mine archived developer's mailing list for last 10 years on lists.mysql.com
7. (One list from Server links, have to determine mainone)  
   - An assumption will be in the tree of messages, that a reply is  
   meant for the author of the previous message in tree.  
     
   High level overview of project tasks:  
   1) Decide on discussion/blog area  
     a) Private group in bb vista  
   2) Collect data from mailing list archives (<http://lists.mysql.com>)  
   3) Create Postgres/PostgreSQL database using schema from sql.tar.gz  
   (Use initMails.sql and person and alias table from initBase.sql)  
   4) Process data collected from (<http://lists.mysql.com>)  
   5) Populate Postgres/PostgreSQL database with the data  
   6) Analyze communication (reference previous papers discussed in  
   class, see Project Goals above)  
   7) Produce a final report analyzing our findings  
     
   Immediate TODO's:  
   -Get pointers to the relevant papers  
   -Get pointers to tools that will help us calculate c/p  
   -Determine if mySQL has its mailing list archived  
   -Determine correct mailing list to use  
   -Get a database set up on tux  
   Timeline:  
   -Data Collection:  Approximately 2 Weeks  
   -Data Analysis:  Approximately 4 Weeks

# Evaluation (max 2 pages – excluding figures)

Here is where you present all of your results and discuss them as in depth as you can. – present the table and graphs of results and discuss them in-depth

Analyze and discuss what those results mean with respect to the issue chosen for the project. – Explain what this means in general for mysql and how open source and dsd compare.

Table of Results:

|  |  |  |  |
| --- | --- | --- | --- |
| Release Name | Release Date | C/P Metric 1 | C/P Metric 2 |
| mysql-3.20.32a.tar.gz | N/A |  |  |
| mysql-3.21.33b.tar.gz | 6/1998 |  |  |
| mysql-3.22.32.tar.gz | 2/2000 |  |  |
| mysql-3.23.57.tar.gz | 6/2003 |  |  |
| mysql-3.23.58.tar.gz | 9/2003 |  |  |
| mysql-4.0.26.tar.gz | 9/2005 |  |  |
| mysql-4.0.27.tar.gz | 5/2006 |  |  |
| mysql-4.1.21.tar.gz | 7/2006 |  |  |
| mysql-4.1.22.tar.gz | 11/2006 |  |  |
| mysql-5.0.45.tar.gz | 7/2007 |  |  |
| mysql-5.1.22-rc.tar.gz | 9/2007 |  |  |
| mysql-5.1.23-rc.tar.gz | 1/2008 |  |  |
| mysql-5.2.0-falcon-alpha.tar.gz | 1/2007 |  |  |
| mysql-5.2.3-falcon-alpha.tar.gz | 2/2007 |  |  |
| mysql-6.0.0-alpha.tar.gz | 4/2007 |  |  |
| mysql-6.0.2-alpha.tar.gz | 9/2007 |  |  |
| mysql-6.0.3-alpha.tar.gz | 11/2007 |  |  |
| mysql-6.0.4-alpha.tar.gz | 3/2008 |  |  |

What I'd like you guys to do is to figure out a date for each of the   
above releases, which will be the moment we'll use to take snapshots of the Social Network mined from the ML and compute its Core/Periphery metrics.

If you look at the references of last week's Crowston's paper, you'll find a paper by Borgatti + 1 co-author which described the C/P metric. About the MLs, I was wrong: the relevant ones (developers' communications) are "internal" ones: specifically Internals and  possibly also Bugs. Let's start with the former.

Cover both strengths and shortcoming of your approach. – Strengths – got data into common format first, reproducible, uses proven products, Weaknesses – errors in mining, dependent on other’s,

# Conclusions (max 1 page)

Summarize the lessons you have learned from the project. – Mining is hard, point to position paper. Open and dsd have similar things

What is the most significant contribution? Full c/p analysis of mysql at plenty of different releases, see trends as the product developed

How would you do it anew, if you had the hindsight you have now? – devoted more time to the mining process, worked with own statistics so weren’t limited in the end by UCINET

Is there any promise? – definitely promise, shows the results of c/p studies on many releases of mysql

How can this work be further extended? – more releases studied, mining more extensive, more c/p statistics taken

# References

Borgatti, S.P., Everett, M.G. and Freeman, L.C. 2002. Ucinet for Windows: Software for Social Network Analysis. Harvard, MA: Analytic Technologies.

Crowston/Borgatti Paper