



Final Presentation

Position Paper

Final Project Report

Presented on: 6/07/2010

Sammie Stahlback

Jordan Osecki

Michael Kim

● ● ● | 0 – Agenda

➤ Position Paper

- Presentation
- Discussion

➤ Final Project

- Presentation
- Discussion



Position Paper Presentation

The Benefits of Separating the Data Mining of
Repositories from the Research Analysis Phase

Sammie Stahlback

Jordan Osecki

Michael Kim



1 – Abstract

- MSR – Mining Software Repositories.
- There is a field of study devoted to MSR.
- There are many benefits with MSR.
 - Data can be analyzed to determine causes for success and failure.
 - Gives guidance on the current health of a project.
- Challenges with MSR:
 - Historical data can be stored in a variety of formats. For instance, you can have CVS repositories, news groups, bug tracking, etc.
 - Need a certain amount of knowledge to work with each of these formats.



1 – Abstract (cont)

- Because of this, historical repositories are very hard to navigate and work with.
 - Becomes even more difficult when spanning multiple data sources.
- This may limit the type of research and conclusions that can be performed.
- It may also cause inconclusive or even erroneous results for research projects.
- What do we suggest?
 - Currently, data mining is performed along with the analysis process.
 - Suggest that there is value in developing data extraction tools and techniques outside of the research analysis.
 - Goal: treating MSR as a separate process from the analysis phase.



2 – Introduction

- Many research papers that analyze software projects tend to start by mining various repositories, depending on the research subject.
 - The researchers need to develop their own tools to extract the necessary data.
 - While the extraction process is not important for analysis, it may affect the results.
 - This is a time consuming task and it would benefit by having a common tool set as well as working from a common data format.
- Researchers also make decisions on certain heuristics to make further connections with the extracted data.
 - These heuristics may make it difficult for future research efforts to build upon.
 - Having a common repository for extracted data can benefit researchers.



2 – Introduction (cont)

- Look at current MSR tools that are available.
 - Investigate their effectiveness as well as be useful in our proposed solution.
- By treating MSR as its own discipline, can expand research in software development projects.
 - Wish to create “best practices” for MSR.
 - Promote common data formats, tools, schemas, and algorithms to be used by others as well as improve upon.
- By having these tools and techniques in place, it will allow researchers to get off the ground quicker in their research by not having to have to duplicate effort that has already been done.
 - Will also allow researchers to concentrate more on their analysis than worrying about getting their data set up properly.



3 – Current Model

- The current model for performing research on software repositories generally has the following steps:
 - Identify the repositories to extract data from, ie. CVS, bug tracking, new group, etc.
 - Researchers need to understand the format of the data in the various repositories they are using. For instance, protocols used in communicating in mailing lists.
 - Extract the data.
 - Knowledge of the data is needed to make sure nothing important is left out.
 - Perform any heuristics on the data.
 - Look for any additional patterns in the data.
 - These heuristics are specific to the research, which may make it difficult to reproduce.
 - Perform analysis.
 - Present results.

● ● ● | 4 – Our Experience

- Our analysis project consisted of a core/periphery analysis on the open source database MySQL.
- Consisted of data extraction and analysis.
 - These were two distinct tasks that could have been completed independently of one another.
 - Extraction process ended up taking up the majority of the time.
 - Took away from analysis.
- Further discussion of the project to follow after the position paper presentation.



5 – Current Tools

- MSR tools can help hide details that researchers do not need to know.
- They can also present a uniform, clean, and easy to understand format for the researchers.
- There are already tools out there that provide this functionality.
- CVSgrab:
 - Web-based tool used to pull data from a CVS repository.
 - Data can be stored in a database or used as input for another tool for analysis.
 - Researchers that need to pull data from a CVS repository now have an easy to tool to do this.

● ● ● | 5 – Current Tools (cont)

- Without it, researchers would need to develop intimate knowledge of CVS repositories.
- One limitation of CVSgrab is that it may not pull all the data that you need. In which case, CVSgrab would need to be modified to pull this data or a whole separate tool would be needed or created.

➤ iSPARQL:

- A framework that extends past interfacing with any of the software repository data and introduces an example of using exchange formats.
- Uses SPARQL for retrieving and modifying data in Resource Description Framework (RDF).
- Web Ontology Language (OWL) is the format used to define the semantics of the RDF.
- Data gets mined from the repository and exported to an OWL file.



5 – Current Tools (cont)

- The OWL file is then exchanged in RDF format using iSPARQL queries.
- Useful because it uses existing standards developed by the World Wide Web consortium to exchange data.
- Easier to extend the framework to pull data you need.
- This enables researchers to develop heuristics that can be easily duplicated by others.

➤ PROMISE:

- A community site where researchers can publish tools and data collected from experiments.
- This collaboration allows researchers to learn from one another's experiences.

● ● ● | 6 – Advancements and Techniques

- Developing standards in MSR is an important advancement.
 - A standard for exchange languages, similar to what is seen with iSPARQL, will help integrate data from many different sources.
 - These exchange languages need to be extensible to be able to allow new paths of research.
- Communities for MSR are also a growing field.
 - Allows data to be collected and shared.
 - PROMISE.
- Extraction tools to help hide protocol details that do not add any value to the research.

● ● ● | 7 – Benefits of Separation

- Inviting for researchers to get started with projects without having to become experts with whatever repository format they wish to explore.
 - Having standards will also encourage others.
 - Allows researchers to focus on their data and analysis.
- Easier to have reproducible results created by a standard set of tools and shared mining techniques.
 - Benchmark tests are easier to perform as well if everything is produced in the same way.
- By having exchange languages, only need to develop tools once to get the data, in whatever format, that researchers need.
- Finally, will allow a different set of participants to become experts in the field of MSR to build better tools.



8 – Closed Source Projects

- A lot of research is conducted on Open Source Projects.
 - Data repositories are easy to access.
- Researching Closed Source Projects is also valuable, but presents some extra challenges.
 - Data repositories are stored internally within the corporation. Harder to get access to.
 - These repositories may also contain sensitive information that the corporation would not want to release to outside research teams.
 - Having tools for mining and storing data within a corporation will help give access to researchers.



9 - Conclusion

- Many research projects have used data repositories for their analysis.
 - Each of these projects have to perform both the extraction and analysis process.
 - This leads to extraction tools being re-created for each project, which may lead to inaccurate or difficult to use data.
- By separating these two processes and having a standard set of tools and techniques, we can eliminate a lot of the duplicated work and effort that goes into each of these projects.
- Allows researchers to focus on their analysis and get more meaningful and more accurate results.



10 – Future Work

- What are ways to help develop MSR as a separate field and increase research in software engineering?
- Continue to use and develop standards.
 - Build on standards presented with TA-RE.
 - Authors of TA-RE cover many of the requirements for sharing data and tools within the software research community.
- Build acceptance within the industry. How?
 - Keep it simple and inviting.
 - The use of standards.
- Build the community.
 - Use sites like PROMISE.
 - Populate data with more projects.



10 – Future Work (cont)

- Define and expand data sources to increase interest.
 - Open Archive Initiative presents a model that separates data harvesters and data set exchanges.
- As more data becomes available, MSR needs to scale to larger repositories.
- Continue to build the relationship between researchers and MSR.
 - Understand the needs of each. We do not want to automate the whole process as data the researcher needs may be hidden from them.
- And finally, need to keep transparency by stressing documentation standards.



11 – Discussion Topics

- Does anyone have any first-hand experience with any of the technologies discussed here?
- What is more promising – common formats, common tools, communities of tools, communities of repository sharing, or a combination of some of the above?
- Can data extraction ever be automated?
- Would implementation of some of this encourage more private companies to make their repository information available?



Final Project Presentation

Analysis Team – Core/Periphery Analysis of
the Open Source Project MySQL

Sammie Stahlback

Jordan Osecki

Michael Kim



1 – Introduction: Overview

- Data Mining of an Open Sourced Project to discover Core/Periphery structure.
- Communication, bug tracking, and source code repositories contain valuable information about the project structure.
- Analysis on communication repositories, in particular, can shed light on the structure of the developers and give a blueprint of the team.
- In DSD, communication is very important and is mainly done in an asynchronous manner.
 - In typical DSD, communication is done through emails, message boards, bug tracking, etc.
 - Poor communication can lead to misunderstandings of requirements, missed features, release delays, etc.
- Because communication is recorded, it can be analyzed to find weak and strong aspects.

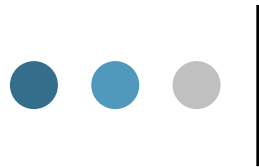


1 – Introduction: Overview (cont)

- OSS is a type of DSD.
- OSS projects tend to have a good success history and result in high quality software.
- OSS projects typically exhibit a core/periphery structure, meaning there is a small group that is responsible for the majority of the code, and a larger group that contributes a smaller amount.
 - The core/periphery members may also change over the course of time, depending on which release they are on.
- What can we learn from OSS that makes them so successful that we can apply them to other DSD projects to give them a higher chance of success?

● ● ● | 2 - Objectives

- Take a successful OSS project and recreate the community and social network for specific releases over the past few years.
 - Mined a mailing list to get the data.
 - Mailing lists will be stored in a common format database.
- Compile metrics on the core/periphery structure of the community during each of these releases.
 - Metrics will be computed using a third-party social networking analysis tool.
 - Create social network diagrams to show trends during these time periods.



3 – Background

- Which OSS project to choose?
 - Needed to be a successful project.
 - Has been around long enough for a community to be established.
- MySQL was selected.
 - MySQL is a popular and successful Relational Database Management System.
 - Initial Release: May 23, 1995
- Communication mainly took place through mailing lists.
 - Multiple categories: Servers, Connectors, Internals, etc.
 - Developers mainly communicated in the Internals category which contained the “Internals” and “Bugs” mailing lists.

● ● ● | 3 – Background (cont)

- Mailing lists were mined using a Perl script.
 - Perl is a great scripting language to parse and manipulate data from the mailing list.
- The Perl script stored all the data into a PostgreSQL database stored on tux.
 - Storing in the database made the data more accessible and flexible for analysis.
 - Important to have everything in a common format.
- Java was used to extract data from the database.
 - Being an object-oriented language, it was very easy to map the database schema to objects.
 - Once data was retrieved, it was converted into a file format that would be used with the social networking analysis tool.

● ● ● | 3 – Background (cont)

- There are many tools (free and shareware), that are available to be used for social networking analysis.
- There were two that were considered for use: NodeXL and UCINET.
- NodeXL is just a template for Microsoft Excel 2007.
 - Pros: Very simple and easy to use. Takes in an edge list (could be from a CSV file or Excel worksheet) and easily generates a graph of the network and computes metrics about the graph. Very customizable.
 - Cons: No core/periphery statistics, only polar layouts and centrality statistics.

● ● ● | 3 – Background (cont)

- UCINET is a shareware tool (30 day trial period).
 - Developed by Steve Borgatti, Martin Everett, and Lin Freeman.
 - Works with a separate graphing program called NETDRAW.
 - Pros: Powerful and does compute core/periphery statistics, as well as other metrics.
 - Cons: Not as easy to use as NodeXL.
- Ultimately, the decision was made to use UCINET for its core/periphery statistics. While the polar layouts and centrality statistics from NodeXL are useful, it would take much more calculation and analysis to get core/periphery metrics from it.



4 – Approach

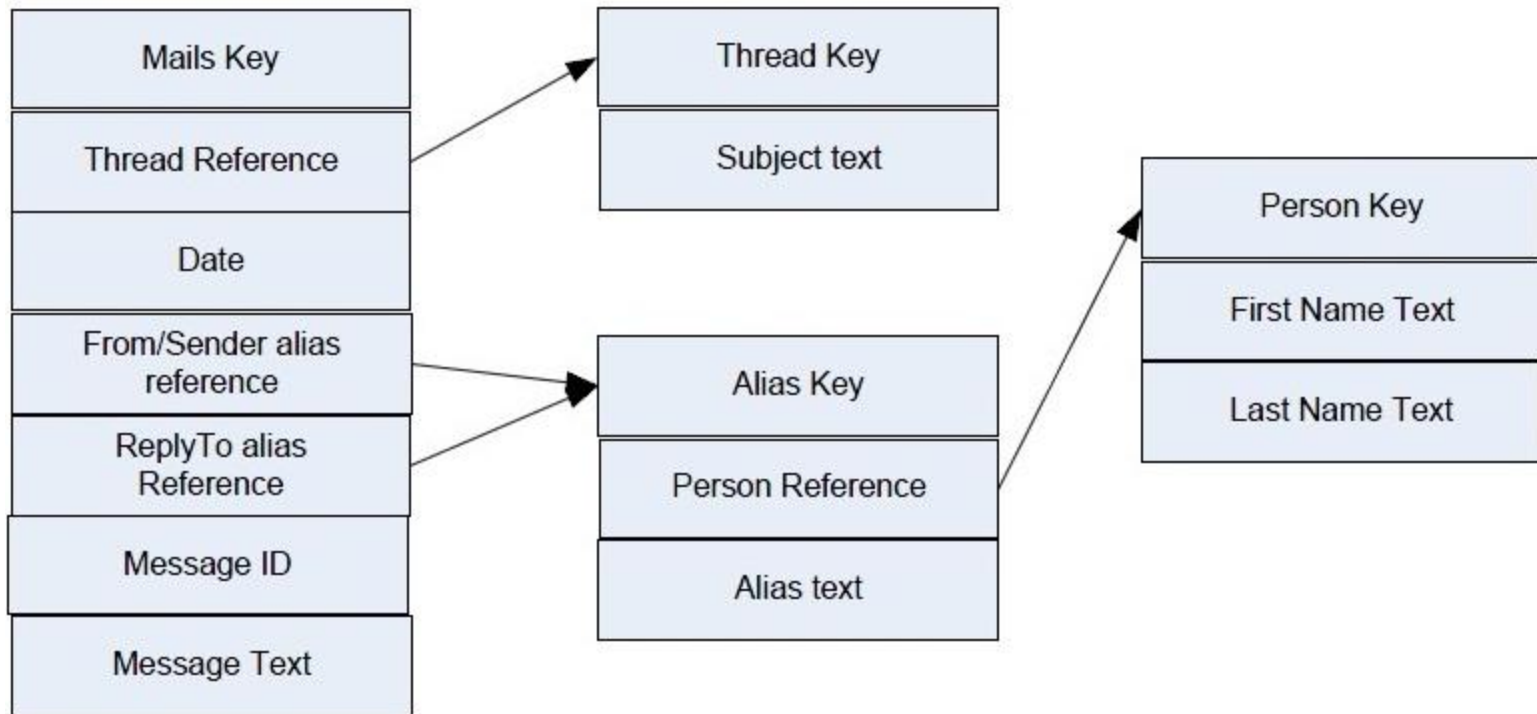
- Here were the necessary steps that were taken for this project:
- 1. Analyze MySQL Mailing Lists.
- 2. Write Script to Collect Data from MySQL Lists.
- 3. Create PostgreSQL Database Using Common Mailing List Schema.
- 4. Populate the Database with the MySQL Mailing List Data.
- 5. Choose a Tool to Analyze the Data.
- 6. Choose the Date Ranges / Releases to Analyze the Data.
- 7. Create a Program to Put the PostgreSQL Information in the Correct Format for the Tool.
- 8. Analyze the Communication Data in the Tool and Produce Final Results.

● ● ● | 4 – Approach: (cont)

- Step 1: Analyze MySQL Mailing Lists.
 - This step consisted of analyzing the mailing lists for MySQL (<http://lists.mysql.com>) and determine which categories to mine.
 - The Internals and Bugs lists were chosen because they were most likely to contain information about the true core/periphery structure.
- Step 2: Write Script to Collect Data from MySQL Lists.
 - Needed to determine best way to mine the data.
 - Discovered a newsgroup that contained the same information as the mailing lists.
 - Created a Perl script to mine this newsgroup and write to the PostgreSQL database.

4 – Approach (cont)

- Step 3: Create PostgreSQL Database.
 - Schema for the database:



● ● ● | 4 – Approach (cont)

- Step 4: Populate Database.
 - Executed Perl script which parsed the mailing lists from the newsgroup and inserted into the database.
 - Verify that database was populated correctly.
- Step 5: Choose a Tool to Analyze the Data.
 - Initially, NodeXL was chosen for its ease of use.
 - UCINET eventually replaces NodeXL.
- Step 6: Choose Timeframe to Analyze.
 - Determine which date range or releases to analyze.
 - The snapshots that were chosen were more recent or interesting MySQL releases, which included 3-5 different versions.

● ● ● | 4 – Approach (cont)

➤ Step 7: Prepare Data for Analysis.

- Need to be able to pull the data from the database based off the dates for the releases that were chosen in Step 6.
- Prepare this data into a format suitable for the analysis tool to read in.
- UCINET takes in a text file which contains an edge list in matrix form.

➤ Step 8: Analyze Data.

- Input data file created in Step 7 into UCINET.
 - Run core/periphery analysis to produce metrics.
 - Analyze metrics that were produced.
- All source code (Java, Perl Script) and documentation are stored in an SVN at the following URL:
<http://code.google.com/p/mysqldatabasecoreperiphery/>



5 - Issues

- During our research, we had certain obstacles that we had to overcome. The two main issues encountered:
- Issue 1: Mining the mailing list:
 - The news groups contains over 10 years worth of data.
 - There was no standardized way for storing these messages.
 - It took a few iterations of parsing the data to make sure it all went in correctly and in the proper format.
 - Just one malformed message could ruin the whole structure and would have to be re-run. Each run would take 10+ hours.
- Issue 2: Tool choice:
 - NodeXL was the original tool choice.
 - Later discovered the metrics it produced would not be sufficient for our core/periphery analysis.
 - The switch to UCINET, while it did make core/periphery metrics easier to produce, had a harder interface to use and the late switch caused some scrambling on the team's part.

6 - Evaluation

- Below is the core/periphery results for each of the given releases:

Release Name	Release Date	Categorical C/P Metric	Continuous C/P Metric
mysql-3.20.32a.tar.gz	N/A	No Data	No Data
mysql-3.21.33b.tar.gz	6/1998	No Data	No Data
mysql-3.22.32.tar.gz	2/2000	No Data	No Data
mysql-3.23.57.tar.gz	6/2003	Jammed	Link
mysql-3.23.58.tar.gz	9/2003	*Link*	Link
mysql-4.0.26.tar.gz	9/2005	*Link*	Link
mysql-4.0.27.tar.gz	5/2006	Link	Link
mysql-4.1.21.tar.gz	7/2006	Link	Link
mysql-4.1.22.tar.gz	11/2006	Link	Link
mysql-5.2.0-falcon-alpha.tar.gz	1/2007	Link	Link
mysql-5.2.3-falcon-alpha.tar.gz	2/2007	Link	Link
mysql-6.0.0-alpha.tar.gz	4/2007	Link	Link
mysql-5.0.45.tar.gz	7/2007	Link	Link
mysql-5.1.22-rc.tar.gz	9/2007	Link	Link
mysql-6.0.2-alpha.tar.gz	9/2007	Link	Link
mysql-6.0.3-alpha.tar.gz	11/2007	Link	Link
mysql-5.1.23-rc.tar.gz	1/2008	Link	Link
mysql-6.0.4-alpha.tar.gz	3/2008	Link	Link

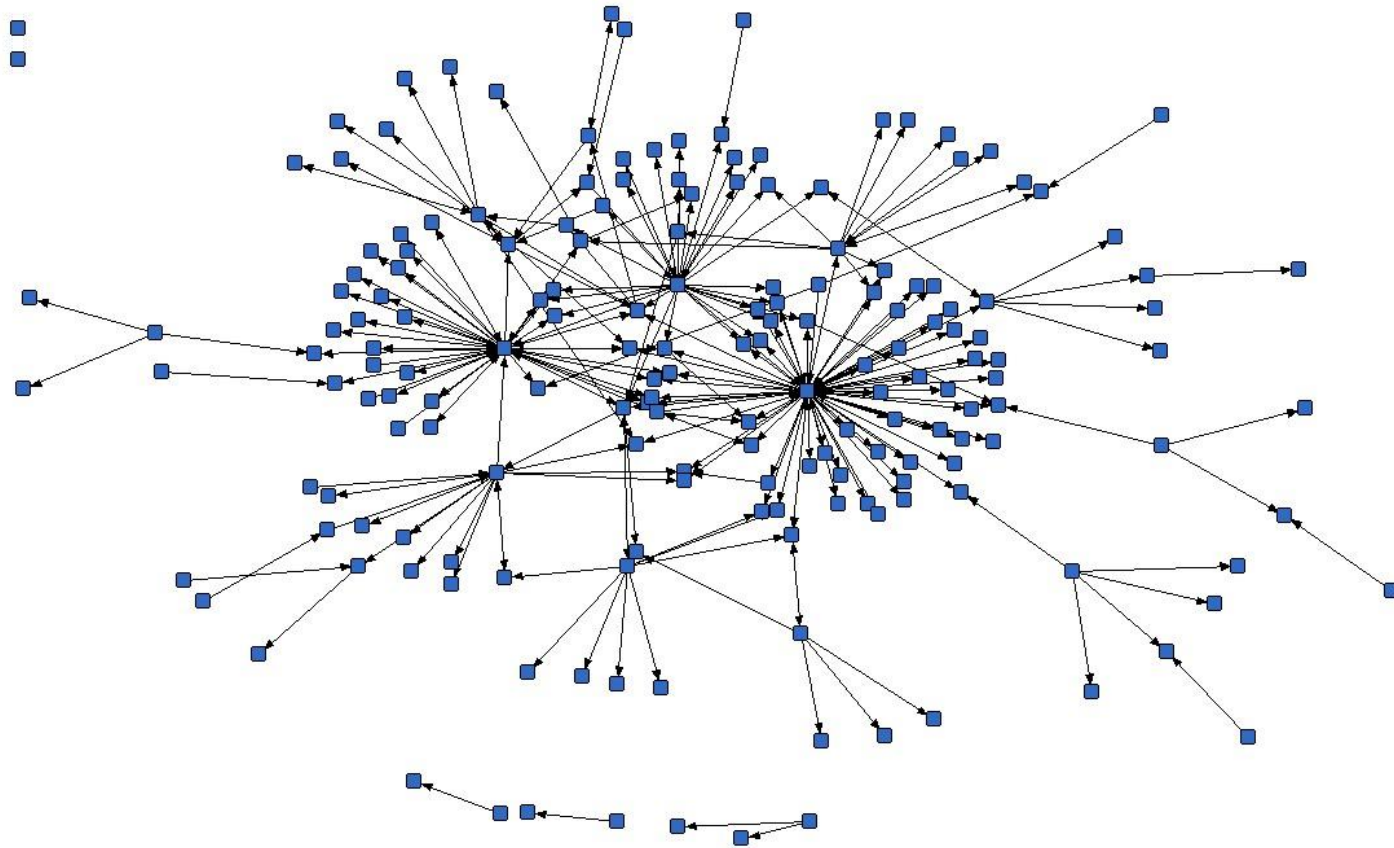


6 – Evaluation (cont)

- The first column represents the specific name of the release.
- The second column shows the date for each of those releases.
- The third column displays the results of the categorical analysis of the core/periphery structure of the MySQL communication structure as a snapshot at that date associated with each release.
- The fourth column is the results of a continuous analysis of the core and periphery structure of the MySQL communication structure as a snapshot at that date associated with each release.

6 – Evaluation (cont)

- Here is an image of the social network for 9/2003:





6 – Evaluation (cont)

- As seen from the results and diagrams, MySQL exhibits the typical core/periphery pattern that is evident in many OSS projects.
- See how the structure of programmers changes from one release to the next.
 - This may be due to issues or changes related to a specific release.
- Because of these results and the close relation between OSS and DSD projects, a core/periphery blueprint should seriously be considered with a DSD environment.

● ● ● | 7 – Lessons Learned

- During this research experiment, the team learned many important lessons.
- The first is that mining a repository, and a mailing list in particular, is a cumbersome process.
 - Having a common format and tools would make this process much easier and streamlined.
 - Most of the teams time was spent in mining the repository and making sure that the data that was collected was in the proper format.
- Also affirmed the similarities between OSS and DSD.
- Some more upfront analysis and decisions, such as picking UCINET from the beginning, would have saved more time for analysis.
- The use of the Perl script and database as a common tool may help future research extend this work much quicker. And the benefits are not just with MySQL, but any other mailing list repository.



8 - Conclusions

- The most significant contribution of this project is the full core/periphery analysis that was performed.
- Shows trends in the structure of the development team throughout the different releases.
 - Changes in core/periphery between different releases could be due to specific changes or issues with that release.
- Further research can examine other releases besides the ones that were studied.
 - Can also use other core/periphery or centrality metrics besides the ones that were used in this project.
- Finally, again we see a core/periphery structure being exhibited in a successful OSS project. This result cannot be a coincidence and may be a source for success for traditional DSD projects.



9 - Discussion

- Further research could involve looking at beginning releases of MySQL:
 - Do you think that core/periphery will be present from the beginning? If not, at what point will we see it?
- How would a core/periphery structure look in a DSD environment?
 - Does it have to be DSD? Can core/periphery be used in co-located environment as well?
- Do you see research being coupled to third-party software as a strength or a weakness?
- Common formats again – How would it have helped here? How would it affect reproducibility and extensibility?