The Benefits of Separating the Data Mining of

Repositories from the Research Analysis Phase

CS680 Distributed Software Development Position Paper (Spring 2010)

Sammie Stahlback

Drexel University  
3141 Chestnut Street  
Philadelphia, PA 19104

sws28@drexel.edu

Jordan Osecki

Drexel University  
3141 Chestnut Street  
Philadelphia, PA 19104

jmo34@drexel.edu

Michael Kim

Drexel University  
3141 Chestnut Street  
Philadelphia, PA 19104

mk394@drexel.edu

# ABSTRACT

There are many benefits from collecting history from software projects by mining software repositories. The data can be used to analyze which variables contributed to the success or failures of a particular task or possibly the whole project in a number of different categories. In general, much can be learned both by the developers of the software and by researchers from mining software repositories. Software repositories can give guidance to the developers regarding the current project and help researchers come to conclusions about more general results, benefitting future projects everywhere.

Historical information is typically mined from a wide range of sources. Because there is such a wide breadth of data sources to choose from, there is also a wide range of data formats that need to be handled. Each type of repository is stored in different ways and in different places. In addition, each project may change the way respositories are managed over time.

This results in historical information sources being hard to navigate and parse for any particular software project and it is almost impossible to scale research results across multiple data sources. Because of the intense effort required for data mining, the conclusions and types of research that can be performed on software projects becomes very limited and the scalability of any conclusions can be extremely questionable. As a result, research analysis is severely impacted by these constraints.

This position paper proposes that there is value in separating the data mining effort of software repositories from the research analysis process. Currently, data mining is performed alongside the analysis process, but there appears to be a need to develop data extraction techniques outside of the research itself because of the breadth and complexity truly involved in mining software repositories.

**Categories and Subject Descriptors**

H.2.8 [**Database Applications**]: Data mining.

**General Terms**

Design, Documentation, Experimentation, Management, Measurement, Reliability, Standardization, Theory.

**Keywords**

Distributed software systems, co-located software systems, mining software repositories, research analysis, repositories.

# Introduction

There are many benefits from collecting history from software projects by mining software repositories. Data repositories include code repositories, bug reports, emails, and other communication archives.

We have seen many examples [list references here] where the research starts with the the development of tools in order extract data from the various data repositories. Along with developing tools, the researches also had to make decisions regarding certain heuristics in order to

This paper will also investigate existing tools and techniques that are currently available for mining software repositories, to see if any can be incorporated into this new separate process. It will analyze what these tools try to accomplish, the effectiveness of them, and what can be learned from them as the need for mining software repository tools grows.

Treating the task of mining software repositories as its own discipline will help expand research in software development projects. Ways in which this will help include the following: creating new “best practices” for extracting data and promoting common data formats, tools, schemas, and algorithms for others to pickup and approve upon. As a result, it will allow researchers to more easily duplicate results of others or expand upon their research without having to reinvent the data extraction process.

By allowing others to more easily build upon common extraction techniques, researchers will be less hampered by this process and no longer forced to gloss over pertinent parts of the research. It will also bring additional researchers into the area of both mining software repositories and analysis.

Section three outlines the basic steps used while performing research and how data mining is large portion of that effort. Next in section four, we highlight several tools and techniques used for data mining and the various levels value they give to researchers.

TODO

# Current Method to Conduct Research

The current strategy typically used while performing research on software repositories is a simple model. This model consists of identifying the repositories, extracting the data, perform any additional heuristics on the data, perform the analysis, and then present the results.

First, identifying the repositories with the necessary data required to calculate the metrics. Since the data comes from several different repositories, the researchers must first become familiar with the data repository. The researchers need to know the standards behind the different data formats. For example, a common repository used for mining is a project mailing list. Researchers need to learn the protocols used to communicate with mailing lists.

Once it is known how to talk to the repository, then the extraction process is performed. During this step, the format of data being extracted needs to be understood. This may require research into the associated RFC(s) in order to determine what types of assumptions can be made while parsing the data. Without having the apppropriate knowledge regarding the data format, the extraction process may exclude important data or include erroneous data that may affect the final results. Continuing the mailing list example, it’s important to understand the required fields within the header information and format of those fields.

Even after data is extracted and parsed, it’s common for researchers to perform additional heuristics on the data. Heuristics are often an important step in order to show additional patterns in the extracted data or perform additional mappings that help generate additional metrics. This is a step that may make if difficult for other researchers to extend on the result and conclusions of the previous researchers.

Finally, analysis can be performed on the data using the metrics of interest and the results presented or possibly stored in another repository.

While understanding every detail in this process can be important and help researchers establish potentially new avenues for research, the effort will extend the amount of time required to complete the research. Each research effort will end up duplicating the same effort.

TODO

# Available Methods of Data Mining

To help reduce amount of effort required during the extraction process, it will help if there were a group of standard tools that can be used so intimate details of the repositories can be hidden from the researchers. Some of the details that can be hidden are the network protocols being used, APIs into the repository, and the exact syntax of the data in the repository.

The following tools and techniques show a few different ways that information can be hidden from researchers and will allow them more flexibility while analyzing data from software repositories.

Also, seperating the data mining process can help individual research projects and help the the software engineering research community grow as a whole.

TODO

## CVSgrab

The first step towards building a community around data mining is establishing a set of tools that don’t have to be written for each research project. CVSgrab is tool presented in [8] is an example of such a tool.

In a nutshell, CVSgrab is a web based tool that can be use to pull data from CVS repositories. Data about that can be entered into a database for further analysis or used as input into another tool to display the results.

CVSgrab is an example of a tool that frees up researchers from having to understand how to talk with a CVS repository by hiding the network details. CVSgrab also hides internal details of the repository and, therefore, the researcher can narrow their focus towards details of the CVSgrab tool itself. If CVSgrab wasn’t available then the researcher would have to develop a strong knowledge about how code is stored in CVS repositories and perform all the necessary data parsing.

While a set of tools like CVSgrab are valuable in saving time during data extraction, there are still limitations in research growth. These limitations are mainly due to the fact that a researcher would have to modify CVSgrab in order to extract any additional information that the tool couldn’t provide.

The next tool presented will illustrate a framework that helps the mining community further will hide details about the repository even further and allow for easier ability to extend the data model.

TODO

## iSPARQL

Research done by [6] presents an interesting framework that extends past interfacing with any details of the software repository data and introduces an example for using exchange formats.

Much of the details presented in [6] are beyond the scope of this document but it provides a concrete example of a framework that uses several standards developed from the World Wide Web consortium. The standards highlighted are SPARQL for retrieving and modifying data in Resource Description Framework (RDF) format and OWL to define the semantics of the RDF format.

The initial steps follow the basic steps required for extraction in that the data mined from the software repository is exported to OWL format. After the original data is an OWL file, data is exchanged in RDF format using iSPARQL queries. It’s a powerful concept for a couple reasons. First, by using existing standards to exchange data, there is less additional software that has to be written in order to retrieve data once it has been exported to OWL. This perceivably opens the door for more researchers to duplicate results and perform research without having to write a bunch of additional code to extract the software repository data and convert it into a usable format. Second, now it becomes open and easier to develop heuristics that can be duplicated by others by extending OWL semantics.

This framework helps reduce complexities mentioned by [5] that inhibit researchers from paticipating in software mining.

The next section describes an example where both the tools and framework can be organized into a single community that can be shared and extended.

TODO

## RESEARCH COMMUNITY: PROMISE

Research by [9] uses a tool called TracExtractor and applying the Cox model in order to predict faults in software. While interesting, the item of importance in this research was that the researchers used data collected through a common site called PROMISE [10].

PROMISE is community site where researchers can publish tools and data collected from experiments. This site hosts a collection of data, tools, and results that other researchers can use for comparison with their own research. This type of collaboration if you will, helps build experience around what models work best during analysis and whether the tools being used are accurate.

TODO

# Researcher’s Method of Data Mining

As a case study, we mined the mail list archives of an open source database called MySQL. This is an example where we had to do both the data extraction and analysis. Although it was a single research project there were two distinct tasks that could have been done individually or even independent of one another. We extracted the data from the mailing list and imported the output into a relational database.

The extraction process requires one to almost exclusively focus on learning the protocols and semantics of the mailing list. Otherwise, it became difficult to verify that the data was in fact accurate and the extraction process ran cleanly.

The extraction process presented several challenges related to parsing mail files and understanding what fields could be grabbed. Parsing the files was interesting since the data is more or less free form text. Therfore there was quite a bit of time dedicated to checking that data being read was being output correctly.

In our research, we imported the extracted data into a relational database. Using a database instead of analyzing the data during extraction has a couple benefits.

First, the speed in which you can query a large volume of data from a database versus a newsgroup or mailing list is much faster. This allowed us to collect a large amount of data for analysis.

Second, the data is one step closer to being in a format that can be used to caculate other metrics outside of our own research. Once the schema is published, the next group of researchers can skip the process of learning the details about collecting mailing list archives and begin analysis on data stored in the databases.

Third, rather than being limited to tools that can read data in mailing lists; we’ve expanded the number of tools that we can use to analyze the data. Finding tools that read from a database is much easier than finding tools that read mail headers.

Clearly, we could have saved ourselves time and effort during analysis if the extraction process was already done and we started analysis with the database already populated. Or at the very least undstand that the extraction process was not a trivial task and maybe plan accordingly.

Along with saving time, we could have possibly developed more models that could have been applied to our data.

TODO

# Advancements and Techniques for MSR

One of the more important advances in MSR is developing standards similar to those presented in [11]. First, a standard for an exchange language should be defined in order to integrate data from many different sources.

The exchange lanugages that have been defined, attempt to cover a wide range of information. They have several common requirements that will help increase adoption within the MSR community. A few of the more inteteresting requirements are that the exchange language needs to be extensible in order to aticipate new paths of research. Also, the exchange language should support such flags as quality codes to identify data that is more trust worthy than data that might known have know issues or created from special heuristics. Along this same topic, the exchange lauage be able to support what types of algorithms were used for inferred information.

The next advancement in MSR is the formation of common areas or communities where extracted data is collected and shared. There are several collections of data that cover specific areas. PROMISE [10] contains data used in predictive models while FOSSology project [12] maintains data about software dependencies in order to organize the complex license dependencies within open source software.

Finally, extraction tools are being developed that hide much of the protocol details needed to interface with repositories but provide no additional research value. As a result, this frees up researchers to focus on defining data and developing models.

TODO

# The Benefits of Separation

More module

Adopting common standards in order to invite more researchers into the field without having to become email experts, cvs experts.

The benefits of an exchange language is that researchers only need to develop tools once in order to import/export data. Rather than every research having to develop tools for the platform that they are storing data. Rdbm, xml, etc..

Allow people to foucs and become experts that can focus on extraction and develop better tools that accurate and cover a larger spectrum of data sources.

Open research to more people by removing the amount of knowledge required for entry. Menaing, researchers can focus on the data they are trying to model and not have to spread their focus on learning email, cvs, or bug reports in order to extract the data.

Reproducible results by having data sets together and standards around how that data was collected. Along with reproducing results, researchers can also perform benchmark tests. Performing benchmark tests or comparing results is difficult if researchers are using different methods and heuristics while extracting data.

TODO

# Business Preparation

This section describes how a business can prepare itself and its projects to implement our position and be able to set-up their stuff to make MSR easier to do.

Develop cost benefits and how accurate defect predictions are. Being able to model development for an industry.

Anonymous data

Enable as much logging as possible

Build a framework that will collect data from differenet sources and be analyzed over time from project to project.

TODO

# Conclusion

This section will review the major points of our position and summarize everything else discussed, as well as reflect on any new conclusions or points discovered throughout the process.

There has been a lot of valuable research that has taken advantage of the data repositories containing a diverse set of software project data. In general, each research project is performing both the extraction process and the analysis effort. This is model is less than optimal for a couple reasons. First, extractrion tools are being reinvented for each project and this leads to results that are hard to extend or reproduce.

We’ve shown several areas where seperating the mining process from the analysis will benefit the growth of software research. Increased participation occurrs by removing the amount of knowledge required in order to perform analysis on data collected from software project. Researchers no longer have to understand details about the software repositories and instead can focus on creating reproducible research that can be extended by using standard data exchanges.

TODO

# Future Work

In recent years there has been a push in advancing the field of mining software repositories (MSR). [5] Outlines the challenges of MSR and why it’s important to approach this task seperately as a stand-alone field.

Continue to develop standards bulding TA-RE. the benefits of establishing standards is proven through the RFC process. apply to exchange lanugage.

Most important. Build acceptance within the industry. How can this be done? keep it simple and inviting.

Continue to develop efficient and accurate tools but move towards stressing exchange language standards.

Determine why ta-re didn’t succeed.

Continue to populate data with more projects.

Define and expand data ousrces.

Open archive initiative has a good model that sepereates data harvesters and data set exchange.

Scaling MSR to larger repositories as more information becomes available.

Continue to develop good relationship between MSR and researchers. The goal isn’t to automate the whole mining process, otherwise there could be data not visible to the researchers that could be valuable.

TODO

# ACKNOWLEDGMENTS

Our thanks to ACM SIGCHI for allowing us to modify templates they had developed.

The team would also like to thank Professor Valetto for helping to guide this position paper in the right direction and raise issues for the team to consider.

# BIBLIOGRAPHY

1. Alonso, O., Devanbu, P. T., Gertz, M., “Database Techniques for the Analysis and Exploration of Software Repositories”, *First MSR Workshop*, ICSE 204, Scotland UK.
2. Bird, C., Gourley, A., Devanbu, P., Gertz, M., Swaminathin, A., “Mining Email Social Networks”, *International Conference on Software Engineering, Proceedings of the 2006 international workshop on Mining software repositories*, 2006, ACM.
3. Canfora, G., Cerulo, L., “Fine Grained Indexing of Software Repositories to Support Impact Analysis”, *METRICS, Proceedings of the 11th IEEE International Software Metrics Symposium*, 2005, IEEE Computer Society.
4. German, D., Cubranić, D., Storey, MA, “A Framework for Describing and Understanding Mining Tools in Software Development”, *International Conference on Software Engineering archive, Proceedings of the 2005 international workshop on Mining software repositories*, 2005, ACM.
5. Hassan, Ahmed e., “The Road Ahead for Mining Software Repositories”. *16th Frontiers of Software Maintenance,* FoSM 2008, Sept 30 – Oct 2 2008, IEEE Computer Society, pp. 48-57.
6. Kiefer, C., Bernstein, A., Tappolet, J., “Mining Software Repositories with iSPARQL and a Software Evolution Ontology”, *Proceedings of the 29th International Conference on Software Engineering Workshop*, 2007, IEEE Computer Society.
7. Robbes, R., “Mining a Change-Based Software Repository”, *ICSEW, Proceedings of the 29th International Conference on Software Engineering Workshops*, 2007, IEEE Computer Society.
8. Voinea, L., Telea, A., “Mining Software Repositories with CVSgrab”, *International Conference on Software Engineering, Proceedings of the 2006 international workshop on Mining software repositories*, 2006, ACM .
9. Wedel, M., Jensen, U., Göhner, P., “Mining Software Code Repositories and Bug Databases using Survival Analysis Models”, *Empirical Software Engineering and Measurement*, 2008, ACM.
10. [*http://promisedata.org/*](http://promisedata.org/)
11. Kim, S., Zimmerman, T., Kim, M., Hassan, A., Mockus, A., Girba, T., Pinzger, M., Whitehead, E. J., Zeller, A., “TA-RE: An Exchange Language for Mining Software Repositories”, *Proceedings of the 2006 international workshop on Mining software repositories*, May 22-23, 2006, Shanghai, China.
12. Gobeille, R., “The FOSSology Project”, *In Proc. Fith International Workshop on Mining Software Repositories*, 2008, pp. 47-50.
13. Gonzalez-Barahona, M, J., “Collaboration Plan and Report”, <http://www.qualoss.org/deliverables/D6_5b_CollaborationReport_submitted.pdf>, 2009.

# APPENDIX

TODO

TODO