Advances in the SourceForge Research Data Archive

Matthew Van Antwerp
Department of Computer Science and Engineering
University of Notre Dame
mvanantw@nd.edu

Greg Madey
Department of Computer Science and Engineering
University of Notre Dame
gmadey@nd.edu

ABSTRACT

The SourceForge Research Data Archive (SRDA), located at http://zerlot.cse.nd.edu, is a collection of Open Source Software (OSS) data and resources [6]. Over 100 researchers worldwide use the archive for research in many fields. In this paper, we describe the recent changes, the work in progress, and future plans for making the archive easier to use and for allowing more advanced research to be done with the data available.

1. INTRODUCTION

We receive monthly database snapshots from SourceForge. They are about 11GB uncompressed. Each dump is a snapshot of SourceForge's back-end database and is loaded into the *timeline* database as a new schema associated with that month (of the form *sfMMYY*). In early 2008, our PostgreSQL database eclipsed 600GB total. Project data available in the monthly dumps includes descriptive and statistical data on projects and users. Two notable missing kinds of data are mailing list and versioning (CVS/SVN) data, which will be addressed in this paper.

We make this data available for researchers who have completed a license form. The data can be accessed through use of a web form as well as through a web service interface. A documentation and instructional wiki is provided as well as a schema browser. Query history and autocompletion are two features that help users make correct queries to retrieve the data they desire.

2. OTHER OSS RESEARCH WEBSITES

In addition to the SourceForge Research Data Archive, there are other websites hosting OSS data or resources for performing research. FLOSSMole (ossmole.sf.net) [12], LibreSoft (libresoft.es), related projects FLOSSMetrics [3], the CVSAnalY tool [14], and SQO-OSS [4] which aims to help analysis and benchmarking of OSS projects. We have the benefit of getting the data directly from SourceForge, saving us much time and energy screen-scraping the data or acquiring it by other means. The FLOSSMole project releases data sets every month and they gather the data from multiple sources. They have the advantage of obtaining the data directly from the SourceForge pages that your browser sees. It can be difficult to find data in *timeline* that is easy to locate directly on a project's SourceForge page. Also, occasionally data in *timeline* appears to be incongruous with project page data. FLOSSMole also has data from other sites besides SourceForge. Many of our users also use FLOSSMole to supplement their research. There are occasionally differences between FLOSSMole data and SRDA data, for numerous reasons. While an examination of such differences is warranted, that is outside the scope of this paper. The CVSAnalY tool helps a researcher manually inspect CVS metadata from OSS projects. An examination of the SRDA along with other research repositories can be found in [13].

3. DATA ACCESS

Data access is restricted to registered users. A query form (found in figure 1) is a guided form that helps a researcher form the appropriate query to obtain they data they need. The most recent addition to this page is autocompletion of text fields. Every potential valid string for the associated SQL field (`select`, `from`, and `where`) is listed that begins with the text already entered in the field by the user. This feature is shown in the aforementioned figure. Data format options are a few record separators as well as XML format (all illegal characters get encoded), and the option to include the query itself as part of the result set.

Additionally, a SOAP (formerly Simple Object Access Protocol) web service interface is available. The SOAP request must authenticate over SSL so this is also restricted to registered users. This programmatic access is more versatile for researchers and more efficient. Users can script as many requests as they desire and obtain results from thousands of queries in the same amount of time it would have taken them to hand craft and submit a few dozen queries using the web form.

Upon successful query completion, that query is saved to that user's query history and automatically loaded into the fields the next time the query page is accessed. The most recent queries are displayed on the query form page in a possibly abbreviated form (they are truncated after a certain length to keep the page from being stretched) and all queries are available in their entirety on the query history page,

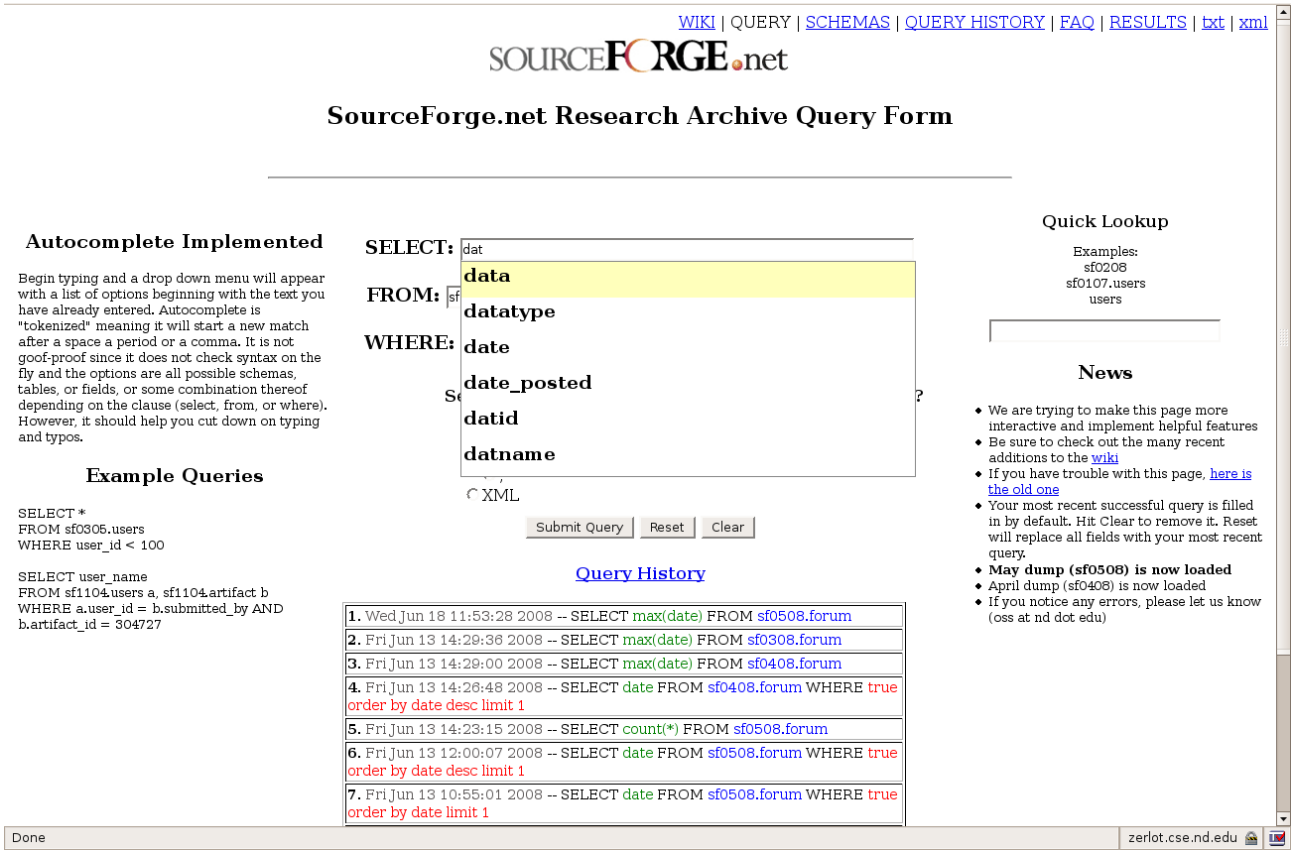

Figure 1: The query form page. The autocompletion feature is displayed.

which is found in figure 2.

4. DOCUMENTATION

The Monthly data dumps come to us undocumented except for what the attribute and table names may indicate. To facilitate querying the data, a wiki (using the mediawiki engine) is provided which hosts researcher and administrator-contributed content. Foreign key constraints are not present in the database, but are implied between certain tables (for example the group_id row appears in many tables, which is an unique project identifier). Every table has its own entry in the wiki along with the database table description from the most recent schema in which that table appears. There are around 80 tables that appear in every schema. Additionally, a brief description of the data contained in that table and information about data accuracy may be available depending on the table.

Another documentation resource provided is the schema browser. Descriptive output is produced dynamically by querying the database with either a schema name (returns the list of tables in that schema) or a schema name and a table name (returns the PostgreSQL table description). Users can query from the top level links to each schema or use the query field to enter a query of one of the following formats: *schema.table, schema, table*. If just a table name is entered, that table's description will be retrieved from the most recent schema. This capability is placed on the right side of the query page to ease table lookup. The schema browser can be seen in figure 5.

5. ADDITIONAL DATA

In addition to the descriptive and quantitative statistics available from the SourceForge data dumps, there are two often requested kinds of data missing, specifically CVS/SVN versioning data and mailing list data. In a working paper [15], we describe the acquisition of CVS (Concurrent Versions System) and Subversion metadata from all SourceForge projects that use those resources and allow public access to them. That data has been loaded alongside the SourceForge data dumps. This metadata is important to OSS researchers. Versioning metadata consists of who made a change (a commit), when the commit was made, the files that were changed, a user-supplied comment, and in the case of CVS, the number of line changes made to the file. Subversion is relatively new, but its adoption is widespread throughout the OSS community and many projects migrated from CVS to SVN (potentially creating difficulty for those studying such projects). This data is now available for querying along with the SourceForge data dumps. More details of the CVS/SVN metadata and the database it is stored in can be found in the aforementioned working paper [15]. Entity-relation diagrams for the versioning metadata are available in figures 3 and 4.

5.1 CVS and SVN Research

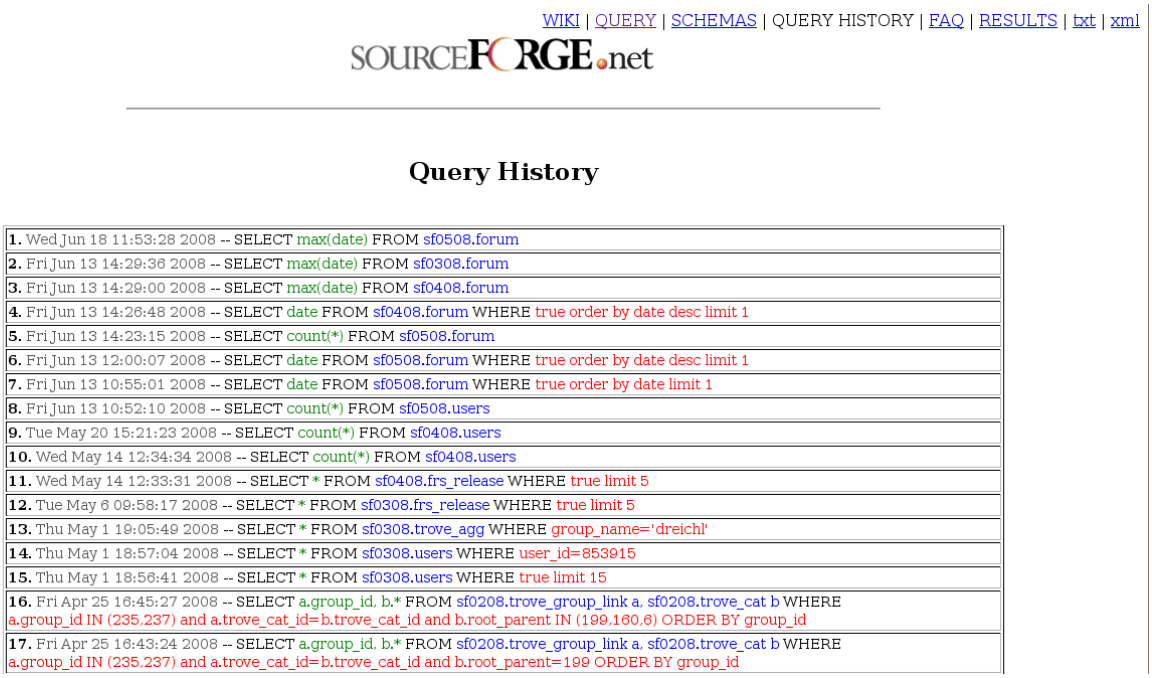

Figure 2: The query history page. Queries are color-coded by section of the SQL statement.

Versioning metadata is an often-used and valuable OSS project resource. In [8], CVS metadata is visually depicted from both a file and author perspective. In both [7] and [9] the authors graph data chronologically and examine the number of modification requests (MRs) in a particular timespan. A modification request is equivalent to a CVS or SVN code commit (check-in). Grouping file changes into one atomic commit is important because it represents one user committing changes to possibly more than one file. This can be useful for determining patterns, such as which files are closely related. While SVN logs are inherently grouped in this manner, CVS logs are not and require careful inspection to determine which files, if any, were committed simultaneously. In [9], they also display social network statistics (which can be determined from the CVS or SVN users data) at different points in time. In [1], the researchers use a tool called `cvs2cl` (http://www.red-bean.com/cvs2cl/) to group CVS logs into change logs, which can allow a better grasp of code changes. Number of CVS commits has been used as a metric for project activity [10] [11]. In [2], CVS activity and many other metrics were used to cluster contributors. In [5], CVS data was studied in various ways in conjunction with data from the SRDA.

5.2 User and Project Data and Connections

CVS and SVN user-project networks can be created for any point in the history of the data. Since it may be useful for a researcher to look at how this network changes over time, it is essential we provide this capability and it can be calculated fairly easily. In the pre-processing phase, first and last commit time are recorded for each user for each project. This data is stored in the user_group tables. With this data and a supplied time, we can determine the user-project network at that time with minimal overhead for small networks

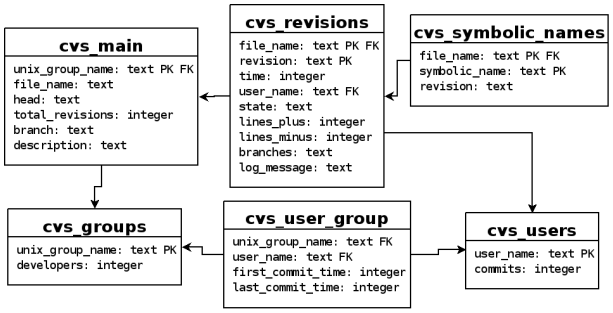

Figure 3: Entity-relation diagram for new CVS data.

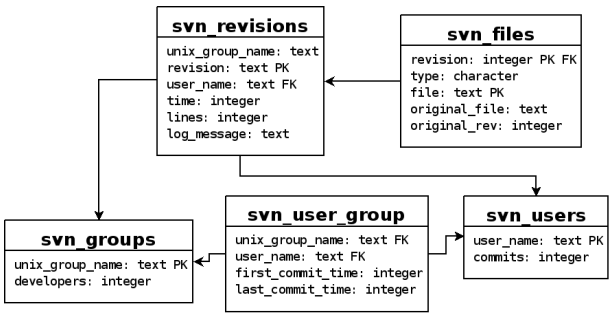

Figure 4: Entity-relation diagram for new SVN data.

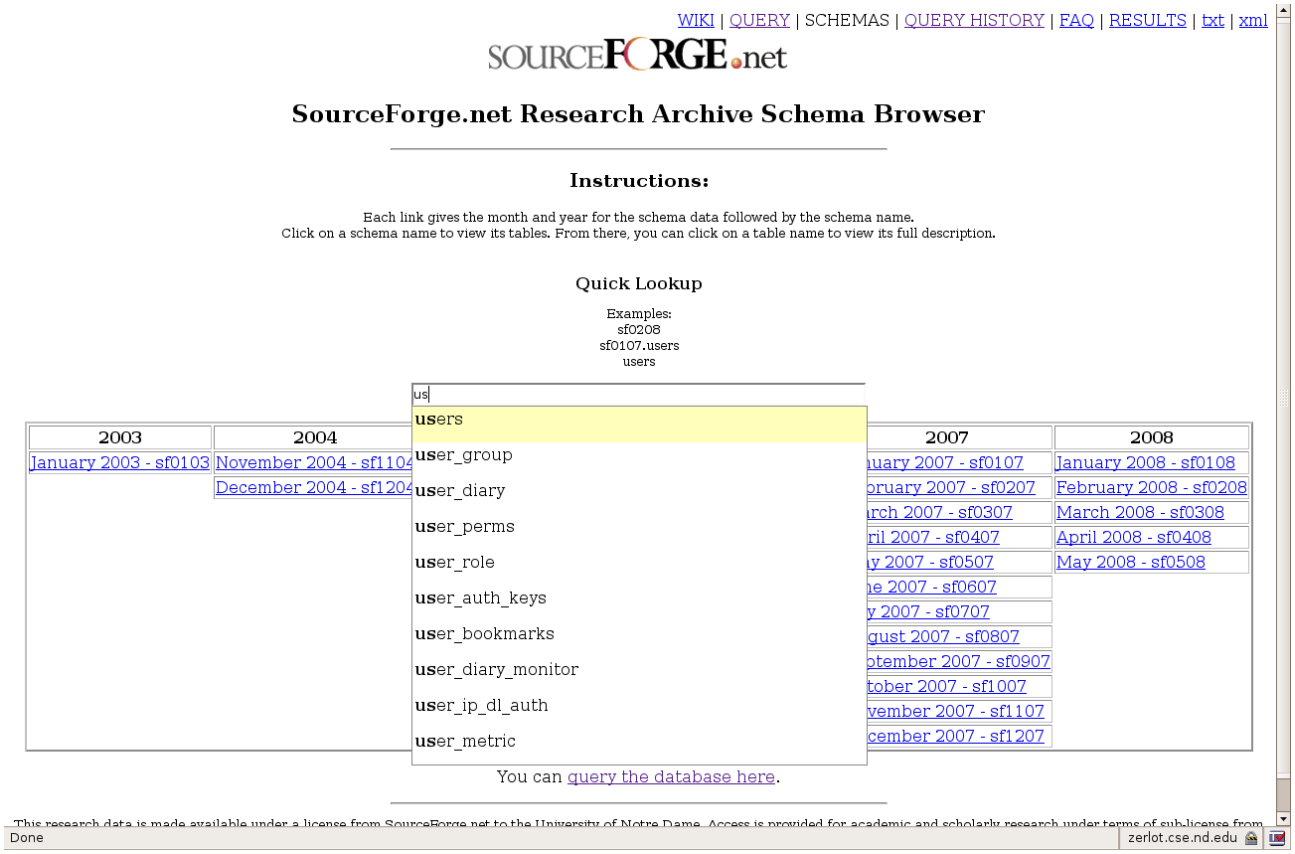

Figure 5: The schema browser. Autocompletion is also shown here.

or cliques by checking if the supplied time is within the first and last commit window. This serves another purpose in that we can easily compare versioning data to *timeline* data which is tied to a particular moment in time.

5.3 Other Open Source Hosting Sites

While SourceForge is the largest OSS development website, BerliOS Developer and GNU Savannah are some other popular OSS hosting websites. BerliOS is a German website with about 1500 users and nearly 3200 projects. GNU Savannah is another hosting site that started when the SourceForge project itself was relicensed as proprietary software. While SourceForge has over 100,000 projects, these small and potentially tight-knit hosting sites offer another rich set of research data. GNU Savannah has an enormous number of revisions on extremely mature and widely used OSS projects dating back decades. We obtained CVS and SVN metadata from all projects on GNU Savannah and BerliOS that use these services. Additionally, java.net hosts software projects and provides CVS repositories. Apache.org is another hosting site for Apache Software Foundation projects, many of which use the Subversion repositories. If possible, we hope to obtain versioning metadata from these sites in the future.

6. ADMINISTRATIVE CONCERNS

For each month when we receive a new data dump, there are hundreds of wiki pages that need to be updated. The reason for regularly updating wiki pages with static data is to show which schemas each table appears in and to display the most recent instance of each table since occasionally minor changes are made to them by SourceForge. Most of these pages need to be updated in a similar way in similar places. To ease this process and improve robustness, updates to many pages are automated. Using a mediawiki versioning tool called *mvs* (http://search.cpan.org/ markj/WWW-Mediawiki-Client/bin/mvs), which provides a command line client, all necessary wiki pages are retrieved and updated using perl scripts. Updates made to other parts of the wiki pages by users remain intact after this update process.

The registration process consists of a scholarly researcher completing a form (which can be found through the wiki for those interested in access requirements and data restrictions) and faxing it to the Principal Investigator (Greg Madey). Then a user is created through the wiki and finally, results directories and symbolic links are created. An authentication module is used by the server to authenticate users through the wiki database when accessing the query page. This has the added benefit of using the wiki infrastructure for password management. Obviously, the same username and password are used when making changes to the wiki.

7. FEATURES IN DEVELOPMENT

A development server was recently deployed, which should help speed development of new and current features. In this

environment, major changes can be tested without worry of interrupting researchers who may be scripting thousands of queries or investigating table information. Features in some phase of development are detailed in this section along with the benefit they will provide.

7.1 Stored Queries

Although the wiki provides a Web 2.0 aspect, we can take things further. Users must query the database using SQL, although many of our users are newcomers to the declarative language. Potentially, there is overlap between the queries researchers are performing. Stored queries would allow users to access their own or other user's previously executed queries that they thought others may find helpful. For example, here is the query to obtain operating system-related information on project with group_id 235:

SELECT b.*
FROM sf0208.trove_group_link a, sf0208.trove_cat b
WHERE a.group_id = 235 and a.trove_cat_id = b.trove_cat_id and b.root_parent=199

This query is difficult to remember and may be difficult to distinguish from similar looking queries that retrieve programming language data, for example. When the stored queries feature is complete, a user will be able to store a query, give it a descriptive name, and identify the variable part (in this query, it is the group_id 235). Then a researcher can access a list of these queries, load one, enter the variable data specific to the project or user they wish to retrieve information from, and execute the query. By default, queries will be public, with the option to hide your queries from other users. Additionally, users will be able to comment on the reliability of a query and view the popularity of a query. This feature will lower the learning curve for new researchers and encourage others not to repeat work that has already been done (typing in a complex query).

7.2 Automated Graph Production

In [7], numerous graphs are provided which visually depict the development history or current state of an OSS project. Researchers will likely be graphing data obtained from the database. Automating this process will save them time and allow them to see interesting trends earlier on in the research process. Versioning metadata plotted over time is one such interesting category that is easily automated given our database structure. In figures 6 and 7, cumulative line changes over time are graphed for the gcc compiler and the emacs editor. Cumulative changes are shown instead of number of changes because when plotted over a large range of time, it is difficult to see smaller changes. A cumulative graph shows sustained development more clearly. Both the gcc and emacs projects span many decades of development.

7.3 Web Service Improvement

The SOAP interface is useful from a programmatic standpoint, but currently limited. The web service actually just executes a query. The user must retrieve the results (which can also be scripted). We also do not provide a WSDL file for users, but rather a wiki page with some sample client code they can use. Result format is also restricted to text with a user-specified field separator. Result sets in XML and programmatic schema browser access are being developed to provide a more complete, cleaner, and more robust

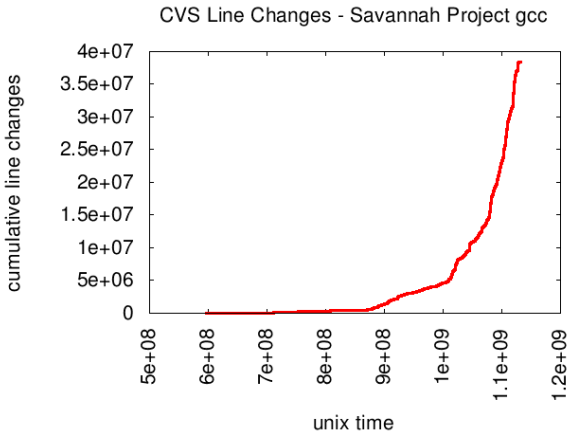

Figure 6: Cumulative line changes over time for the GNU C Compiler. The first tick is Nov 1985 and each tick is approximately 3 years and 2 months. 1.2e9 is Jan 2008.

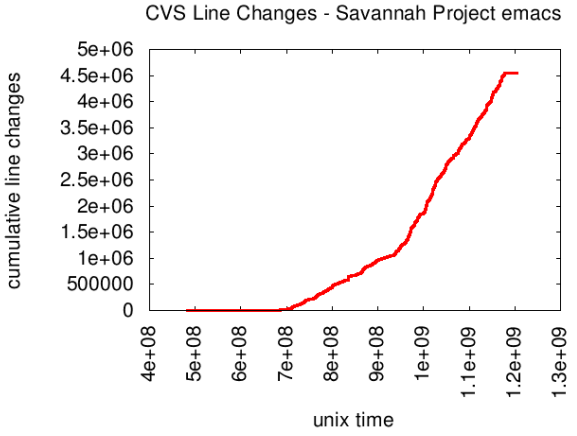

Figure 7: Cumulative line changes over time for the emacs editor. The first tick is Sep 1982 and each tick is approximately 3 years and 2 months.

web service interface to the database and the new versioning metadata tables.

8. PLANNED FEATURES

Result sets can be very large, often around 20MB of text. For such sets, local examination (and probably postprocessing) is necessary. Attempting to view such sets can hang a browser. Result pagination or smart inspection is a planned feature to ameliorate this shortcoming. Such a feature would also allow for visual postprocessing and reordering. Another shortcoming of the archive is that results simply clobber the most recent results. That is, all queries output to the same two files (one for text results, one for XML results). This was a naive default decision to deal with space constraints in an easy to implement manner. However, this means queries are potentially being repeated if a user forgets to download results before executing the next query. A helpful feature would be to store results in a database and allow users to name result sets or provide short descriptions for later retrieval. This would help researchers from a logistical standpoint.

Another improvement to be made is to the autocompletion feature. Currently, it is naive and will offer completions that make sense syntactically but not logically with regards to the database (e.g. select a field name from a table that does not have that particular field). A smarter solution is to parse text entries asynchronously and only return relevant and valid options for the autocompletion feature. Additionally, we are in the process of obtaining mailing list data. Conceivably, the only other widespread data related to SourceForge to be obtained after this is source code itself, which is another long-term goal of the SRDA.

9. CONCLUSIONS

In this paper, we described the state of the SourceForge Research Data Archive, a valuable resource for Open Source Software research. Notably, new versioning metadata for over 100,000 projects has been added and features were implemented to ease the research process for our users. Features and resources under development were described as well as plans for the future. Current work should yield helpful user contributions in the form of stored queries. We hope the archive continues to help researchers around the world and that our users find our additions helpful.

10. ACKNOWLEDGEMENTS

The material presented in this paper was supported in part by grants from the National Science Foundation's CISE IIS-Digital Society & Technology program under Grant ISS-0222829 and by the National Science Foundation's CISE Computing Research Infrastructure program under Grant CNS-0751120. Y. Gao and S. Christley contributed to early versions of the SourceForge Research Data Archive (SRDA).

11. REFERENCES

[1] D. Beyer and A. Noack. Mining co-change clusters from version repositories, 2005.

[2] S. Christley and G. Madey. Global and temporal analysis of social positions at sourceforge.net. In *The Third International Conference on Open Source Systems (OSS 2007)*, Limerick, Ireland, June 2007.

[3] C. Daffara and J. Gonzalez-Barahona. Flossmetrics project, 2007.

[4] A. de Groot, S. Kügler, P. J. Adams, and G. Gousios. Call for quality: Open source software quality observation. In E. Damiani, B. Fitzgerald, W. Scacchi, M. Scotto, and G. Succi, editors, *OSS*, volume 203 of *IFIP*, pages 57–62. Springer, 2006.

[5] D. P. Delorey, C. D. Knutson, and A. MacLean. Studying production phase sourceforge projects: An exploratory analysis using cvs2mysql and sfra+. In *2nd International Workshop on Public Data about Software Development (WoPDaSD), co-located with The Third International Conference on Open Source Systems*, Limerick, Ireland, June 2007.

[6] Y. Gao, M. VanAntwerp, S. Christley, and G. Madey. A research collaboratory for open source software research. In *FLOSS '07: Proceedings of the First International Workshop on Emerging Trends in FLOSS Research and Development*, page 4, Washington, DC, USA, 2007. IEEE Computer Society.

[7] D. German and A. Mockus. Automating the measurement of open source projects. In *Proceedings of ICSE 03 Workshop on Open Source Software Engineering*, Portland, Oregon, 2003.

[8] E. Gilbert and K. Karahalios. Lifesource: two cvs visualizations. In *CHI '06: CHI '06 extended abstracts on Human factors in computing systems*, pages 791–796, New York, NY, USA, 2006. ACM.

[9] L. L.-F. Gregorio. Applying social network analysis to the information in cvs repositories.

[10] K. Healy and A. Schussman. The ecology of open-source software development, 2003.

[11] D. Hinds. *Social Network Structure as a Critical Success Condition for Open Source Software Project Communities*. PhD thesis, Florida International University, 2008.

[12] J. Howison, M. Conklin, and K. Crowston. Flossmole: A collaborative repository for floss research data and analyses. In *International Journal of Information Technology and Web Engineering, 1(3)*, pages 17–26, 2006.

[13] G. Madey and S. Christley. F/oss research repositories & research infrastructures, February 2008.

[14] G. Robles, S. Koch, and J. M. González-Barahona. Remote analysis and measurement of libre software systems by means of the cvsanaly tool. In *In Proceedings of the 2nd ICSE Workshop on Remote Analysis and Measurement of Software Systems (RAMSS), 26th International Conference on Software Engineering*, Edinburgh, Scotland, 2004.

[15] M. VanAntwerp and G. Madey. Warehousing, mining, and querying open source versioning metadata, 2008.