Collaboration Using OSSmole: a repository of FLOSS data and analyses

Megan Conklin
Elon University
Department of Computing Sciences
Elon, NC 27244
1(336)229-4362
mconklin@elon.edu

James Howison
Syracuse University
School of Information Studies
Syracuse, NY 13210
1(315)395-4056
jhowison@syr.edu

Kevin Crowston
Syracuse University
School of Information Studies
Syracuse, NY 13210
1(315)380-3923
crowston@syr.edu

ABSTRACT
This paper introduces a collaborative project *OSSmole* which collects, shares, and stores comparable data and analyses of free, libre and open source software (FLOSS) development for research purposes. The project is a clearinghouse for data from the ongoing collection and analysis efforts of many disparate research groups. A collaborative data repository reduces duplication and promote compatibility both across sources of FLOSS data and across research groups and analyses. The primary objective of OSSmole is to mine FLOSS source code repositories and provide the resulting data and summary analyses as open source products. However, the OSSmole data model additionally supports donated raw and summary data from a variety of open source researchers and other software repositories. The paper first outlines current difficulties with the typical quantitative FLOSS research process and uses these to develop requirements for such a collaborative data repository. Finally, the design of the OSSmole system is presented, as well as examples of current research and analyses using OSSmole.

Categories and Subject Descriptors
D.2.8 [**Software Engineering**]: Metrics – *complexity measures, process metrics, product metrics.*

General Terms
Measurement, Human Factors.

Keywords
Open source software, free software, libre software, data mining, data analysis, data repository, source control, defect tracking, project metrics.

1. INTRODUCTION
OSSmole is a collaborative project designed to gather, share and store comparable data and analyses of free and open source software development for academic research. The project draws on the ongoing collection and analysis efforts of many research groups, reducing duplication, and promoting compatibility both across sources of online FLOSS data and across research groups and analyses.

International Conference on Software Engineering Workshop on Mining Software Repositories '05, May 17, 2005, St. Louis, Missouri, USA.

Creating a collaborative repository for FLOSS data is important because research should be as reproducible, extensible, and comparable as possible. Research with these characteristics creates the opportunity to employ meta-analyses ("analyses of analyses") which exploit the diversity of existing research by comparing and contrasting existing results to expand knowledge. Unfortunately, the typical FLOSS research project usually proceeds in a way that does not necessarily achieve these goals. Reproducing, extending, and comparing research project results requires detailed communal knowledge of the many choices made throughout a given research project. Traditional publication methods prioritize results but mask or discard much of the information needed to understand and exploit the differences in the data collection and analysis methodologies of different research groups. OSSmole is designed to provide resources and support to academics seeking to prepare the next generation of FLOSS research.

2. BACKGROUND AND METHOD
Obtaining data on FLOSS projects is both easy and difficult. It is easy because FLOSS development utilizes computer-mediated communications heavily for both development team interactions and for storing artifacts such as code and documentation. As many authors have pointed out, this process leaves a freely available and, in theory at least, highly accessible trail of data upon which many academics have built interesting analyses. Yet, despite this presumed plethora of data, researchers often face significant practical challenges in using this data in a deliberative research discourse.

2.1. Data Selection
The first step in collecting online FLOSS data is selecting which projects and which attributes to study. Two techniques often used in estimation and selection are census and sampling. (Case studies are also used but these will not be discussed in this paper.)

Conducting a census means to examine all cases of a phenomena, taking the measures of interest to build up an entire accurate picture. Taking a census is difficult in FLOSS for a number of reasons. First, it is hard to know how many FLOSS projects there are 'out there' and hard to know which projects are actually in or out. For example, are corporate-sponsored projects part of the phenomenon or not? Do single person projects count? What about school projects?

Second, projects, and the records they leave, are scattered across a surprisingly large number of locations. It is true that many are located in the major general repositories, such as Sourceforge and GNU Savannah. It is also true, however, that there are a quickly growing number of other repositories of varying sizes and focuses (e.g. CodeHaus, GridForge, CPAN (the perl

repository) ...) and that many projects, including the well-known and well-studied Apache and Linux projects, prefer to "roll their own" tools. This locational diversity obscures many FLOSS projects from attempts at census. Even if a full listing of projects and their locations could be collated, there is also the practical difficulty of dealing with the huge amount of data—sometimes years and years of email conversations, source control data, and defect tracking data—required to conduct comprehensive analyses.

These difficulties suggest sampling, or the random selection of a small, and thus manageable, sub-group of projects which is then analyzed to represent the whole. While sampling could solve the manageability problem presented in census-taking, there is still another difficulty with both processes: the total population from which to take the sample selection is not well-defined. Perhaps more importantly, sampling open source projects is methodologically difficult because everything FLOSS research has shown so far points to massively skewed distributions across almost all points of research interest [1] [8]. Selecting at random from these highly skewed datasets will yield samples which will be heavily weighted to single-developer projects, or projects which are still in listings but which are stillborn, dormant, or dead. These are often not the most interesting research subjects.

The large skew also makes reporting distributions of results at best difficult and at worst misleading because averages and medians are not descriptive of the distribution. The difficulty of sampling is demonstrated in the tendency of FLOSS studies to firstly limit their inquiries to projects using one repository (usually Sourceforge), and often to draw on samples created for entirely different purposes (such as top-100 lists as in [6]), neither of which is a satisfactory general technique.

2.2. Data Collection

Once the projects of interest have been located, the actual project data must be collected. There are two techniques that prevail in the FLOSS literature for collecting data: web spidering and obtaining database dumps.

Spidering data is fraught with practical complexities [5]. Because the FLOSS repositories are usually maintained using a database back-end and a web front-end, the data model appears straightforward to reproduce. The central limitation of spidering, however, is that the researcher is continually in a state of discovery. The data model is always open to being changed by whoever is controlling the repository and there is usually no way that the researcher will know of changes in advance. Spidering is a time- and resource-consuming process, and one that is being unnecessarily replicated throughout the world of FLOSS research.

Getting direct access to the database is clearly preferable, but not all repositories make their dumps available. And understandably so: it is not a costless process to make data-dumps available. Dumps can contain personally identifiable and/or financial information (as with the Sourceforge linked donation system) and so must be anonymized or otherwise treated. Repositories are facing an increasing number of requests for database snapshots from academics and are either seeking a scalable way to do releases or declining to release the data[1]. It is often unclear whether database dumps obtained by one research project can be shared with other academics, so rather than possibly breach confidentiality or annoy their subjects by asking for signed releases, it is understandable that academics who do get a database dump do not make those dumps easily available.

Even when a dump is made available, it is necessary to interpret the database schema and identify missing data elements. This is not always as straightforward as one would expect. After all, the databases were designed to be used to build Web pages quickly, not to conduct academic analyses. Furthermore, they have been built over time and face the complexity that any schema faces when stretched and scaled beyond its original intended use: labels are obscured, extra tables are used, there are inconsistencies between old and recently-added data. The interpretation and transformation of this data into information that is interesting to researchers is not a trivial process, and there is no reason to think that researchers will make these transformations in a consistent fashion. It is also possible that some repositories do not themselves store the type of historical information about projects that would be interesting for academic research. For example, while a snapshot of a repository might show the current list of developers each project, it could be missing important historical information about which developers have worked on which projects in the past.

Even pristine and well-labeled data from repositories is not sufficient because different repositories store different data elements. Different forges can have projects with the same names; different developers can have the same name across multiple forges; the same developer can go by multiple names in multiple forges. In addition, forges have different terminology for things like developer roles, project topics, and even programming languages. The differences are compounded by fields which are named the same but which represent different data. This is especially true of calculated fields, such as activity or downloads, for which there is incomplete publicly-available information how these fields are calculated.

2.3. Data Validation

Once projects have been selected and the available data harvested, researchers must be confident that the data adequately represents the activities of a project. For example, projects may use the given repository tools to differing degrees: many projects are listed on Sourceforge, and use the mailing lists and web hosting provided there. But some of these same projects will shun the notoriously quirky *Tracker* bug-tracking system at Sourceforge, preferring to set up their own systems. Other projects host their activities outside Sourceforge but maintain a ‘placeholder’ registration there. These projects will often have very out-of-date registration information, followed by a link to an external Web site. It is very difficult, without doing detailed manual examination of each project, to know exactly how each project is using its repository tools. It is thus difficult to be confident that the data collected is a reasonable depiction of the project’s activities.

Complete accuracy is, of course, not always required because in large scale data analysis some 'dirty' data is acceptably handled through statistical techniques. At a minimum, though, researchers contemplating the accuracy of their data must have some reason to believe that there are no systematic reasons that the data collected in the name of the group would be unrepresentative. Unfortunately, given the idiosyncrasies of FLOSS projects, confidence on this point appears to require project-by-project

[1] It is understood that an NSF funded project on which the Sourceforge project manager is a co-PI is planning to make Sourceforge dumps generally available, but the details of this project are, at the time of writing, not available. See http://www.nd.edu/~oss/People/people.html

verification, a time-consuming process for individual researchers and projects, and one which is presumably repeated by every researcher going through this information-gathering exercise.

The upshot of this issue is that each step of the typical FLOSS research process introduces variability into the data. This variability then underlies any quantitative analysis of FLOSS development. Decisions about project selection, collection, and cleaning are compounded throughout the cycle of research. FLOSS researchers have not, so far, investigated the extent to which this variability affects their findings and conclusions. The demands of traditional publication also mean that the decisions are not usually fully and reproducibly reported.

Our critique is not against the existence of differences in research methods or even datasets. There is, rightly, more than one way to conduct research, and indeed it is this richness that is at the heart of knowledge discovery. Rather, our critique is that the research community is currently unable to begin a meta-analysis phase because the current process of FLOSS research is hampered by variability, inconsistency, and redundant, wasted effort in data collection and analysis. It is time to learn from the free and open source approaches we are studying and develop an open, collaborative solution.

3. PROPOSED SOLUTION

3.1. Goals of OSSmole

The above problem description allows us to identify requirements for building a system to support research into FLOSS projects. We call the system we have built OSSmole. The OSSmole system is a central repository of data and analyses about FLOSS projects which have been collected and prepared in a decentralized, collaborative manner. Data repositories have been useful in other fields, forming datasets and interchange formats (cf ARFF) around which research communities focus their efforts. For example, the TREC datasets have supported a community of information retrieval specialists facilitating performance and accuracy comparisons[2]. The GenBank is the NIH database of all publicly-available gene sequences.[3] The PROMISE software engineering repository is a collection of data for building predictive models of the software engineering process.[4] The goal of the OSSmole project is to provide a high-quality, widely-used database of FLOSS project information, and to share standard analyses for replication and extension of this data.

A data and analysis clearinghouse for FLOSS data should be:

Collaborative—The system should leverage the collective effort of FLOSS researchers. It should reduce redundancies in data collection and free a researcher's time to pursue novel analyses. Thus, in a manner akin to the BSD rather than the GPL licensing model, OSSmole expects but does not require that those that use data contribute additional data and the analysis scripts that they obtain or use.

Available—The system should make the data and analysis scripts available without complicated usage agreements, where possible through direct unmonitored download or database queries. This ease the startup requirements for new researchers who wish to implement novel techniques but face high data collection costs. This will also lower the barriers to collegial replication and critique.

Comprehensive and compatible—Given the multiplicity of FLOSS project forges identified above, the system should cover more than just one repository. The system should also be able to pull historical snapshots for purposes of replication or extension of earlier analyses. Compatibility requires that the system should translate across repositories, allowing researchers to conduct both comprehensive and comparative analyses. There is also the potential to develop a data interchange format for FLOSS project collateral. FLOSS project leaders, fearing data and tool lock-in, might find this format useful as they experiment with new tools or and repositories.

Designed for academic research—The data model and access control features should be designed for convenience for academic researchers. This means a logical and systematic data model which is properly documented with well-labeled fields. The source of each data element should be known and transparent. Researchers should be able to trace the source of each data element so that they can make decisions about whether to include a particular record or attribute in their analyses.

Of high quality—Researchers should be confident that the data in the system is of high quality. The origins and collection techniques for individual data elements must be traceable so that errors can be identified and not repeated. Data validation performed routinely by researchers can also be shared (for example, scripts that sanity-check fields or distributions) and analyses can be validated against earlier ones. This is a large advantage over individual research projects which may be working with single, non-validated datasets. It reflects the "many-eyes" approach to quality assurance, familiar from FLOSS development practices.

Support reproducible and comparable analyses—The system should specify a standard application programming interface (API) for inserting and accessing data via programmed scripts. That allows analyses to specify, using the API, exactly the data used. It is also desirable that data extracted from the database for transformation be exported with verbose comments detailing its origin and how to repeat the extraction. The best way to ensure reproducible and comparable analyses is to have as much of the process as possible be script-driven. Ideally, these scripts could available for analysis by the research community.

A system that meets these requirements, we believe, will promote the discovery of knowledge about FLOSS development by facilitating the next phase of extension through replication, apposite critique, and well-grounded comparisons.

3.2. OSSmole Data Model

The OSSmole data model is designed to support data collection, storage and analysis from multiple open source forges in a way that meets the above requirements. OSSmole is able to take both spidered data and data inserted from a direct database dump. The raw data is timestamped and stored in the database, without overwriting any data previously collected about the same project. Finally, periodic raw and summary reports are generated and made publicly-available on the project web site.

The type of data that is currently collected from the various open source forges includes: the full HTML source of the forge data page for the project, project name, programming language(s), natural language(s), platform(s), open source license type,

[2] http://trec.nist.gov

[3] http://www.ncbi.nlm.nih.gov/GenBank

[4] http://promise.site.uottawa.ca/SERepository

operating system(s), intended audience(s), and the main project topic(s). Developer-oriented information includes: number of developers, developer information (name, username, email), and the developer's role on the project. We have also collected issue-tracking data (mainly bugs) such as date opened, status, date closed, priority and so on. Data has been collected from Sourceforge, GNU Savannah, the Apache foundation's Bugzilla and Freshmeat. We are currently creating mappings between fields from each of these repositories and assessing how comparable the fields are. The forge-mapping task is extensive and time-consuming, but the goal is to build a dataset that is more complete and is not specific to only one particular forge.

Because OSSmole is constantly growing and changing as new forges are added, and because data from multiple collectors is both expected and encouraged, it is important that the database also store information about where each data record originally came from (i.e. script name, version, command-line options used, name and contact information of person donating the data, and date of collection and donation). This process ensures accountability for problematic data, yet encourages collaboration between data collectors. The information is stored inside the database to ensure that it does not get decoupled from the data. Donated raw data files are also stored in their original formats, in case of problems with the database imports or unforseen mapping problems between projects.

Likewise, it is a general rule that data is not overwritten when project details change; rather, one of the goals of the OSSmole project is that a full historical record of the project be kept in the database. This will enable researchers to analyze project and developer changes over time and enable access to data that is difficult or impossible to access once it has slipped from the repositories front ends.

Access to the OSSmole project is two-pronged: both data and scripts are continually made available to the public under an open source license. Anyone can download the OSSmole raw and summary data for use in their own research projects or just to get information about "the state of the industry" in open source development. The raw data is provided as multiple text files; these files are simply tab-delimited data dumps from the OSSmole database. Summary files are compiled periodically, and show basic statistics. Examples of summary statistics that are commonly published would be: the count of projects using a particular open source license type, or the count of new projects in a particular forge by month and year, or the number of projects that are written using each programming language. It is our hope that more sophisticated analyses will be contributed by researchers and that the system will provide dynamic and up-to-date results rather than the static "snapshots" that traditional publication unfortunately leaves us.

The scripts that populate the OSSmole database are also available for download under an open source license. These scripts are given for two reasons: first, so that interested researchers can duplicate and validate our findings, and second, so that anyone can expand on our work, for example by modifying a script to collect data from a new forge. Indeed this process has begun with the recent publication of a working paper comparing and critiquing our spidering and summaries and beginning collaboration [7]. OSSmole expects and encourages contributions of additional forge data. (Each set of donated data is given a unique number so that the different "data sources" can be included or excluded for a given analysis. This allows us to accept donated data, along with a description of where the data came from. This transparency gives researchers the ability to include or exclude the donation from their analyses.) Researchers interested in donating or using OSSmole data should see the OSSmole project page at http://ossmole.sf.net and join the mailing list for information on how to contribute.

4. RESULTS

Because it is a regularly-updated, publicly-available data repository, OSSmole data has been used both for constructing basic summary reports about the state of open source, as well as for more complex social network analyses of open source development teams. For example, summary reports posted as part of the OSSmole project regularly report the number of open source projects, the number of projects per programming language, the number of developers per project, etc. This sort of descriptive data is useful for constructing "state of the industry" reports, or for compiling general statistical information about open source projects. The OSSmole collection methods are transparent and able to be reproduced, so OSSmole can serve as a reliable resource for these metrics. Having a stable and consistently-updated source of this information will also allow metrics to be compared over time. One of the problems with existing analyses of open source project data is that researchers will run a collection and analyze it once, publish the findings, and then never run the analysis again. The OSSmole data model and collection methodology was designed to support historical comparisons of this kind.

OSSmole data was used in a number of large-scale social network analyses of FLOSS project development. Crowston and Howison [3] reports the results of a SNA centralization analysis in which the data suggests that, contrary to the rhetoric of FLOSS practicioner-advocates, there is no reason to assume that FLOSS projects share social structures. Further OSSmole data was used in the preparation of [2] which, in an effort to avoid the ambiguities of relying on ratings or downloads, develops a range of quantitative measures of FLOSS project success including the half-life of bugs. OSSmole makes available the full data and analysis scripts which make these analyses fully reproducible and, we hope, extendable.

Another project using OSSmole data [1] explored whether open source development teams have characteristics typical of a self-organized, complex network. This research investigated whether FLOSS development networks will evolve according to "rich get richer" or "winner take all" models, like other self-organized complex networks do. Are new links (developers) in this network attracted to the largest, oldest, or fittest existing nodes (project teams)? The OSSmole data was used to determine that there are indeed many characteristics of a complex network present in FLOSS software development, but that there may also be a mutual selection process between developers and teams that actually stops FLOSS projects from matching the "winner take all" model seen in many other complex networks.

Recently, another researcher, Dawid Weiss, collected data by spidering Sourceforge [7]. Weiss then compared the data and collection methodology to the OSSmole data collection techniques and results. He chose to focus mostly on the changes between when his results were gathered, and when the first OSSmole results were gathered a few months prior. There are two main differences noted in this technical report. First, he discovered that the Sourceforge management team made changes to the data in between the two gathering processes (specifically,

they relabeled all the target operating systems and recategorized them). Second, there are differences in how data is gathered and cleaned between research projects (specifically, the OSSmole team cleaned out any inaccessible project for which we could gather no information other than a name, but he did not do this cleaning). These two observations about the data collection and analysis effort are precisely why OSSmole desires to be a collaborative, "many eyes" approach.

The most interesting thing about the intersection of the Weiss research with OSSmole is that he found the OSSmole dataset without our assistance, conducted numerous analyses, then contacted our team to share his results. This experience illustrates the convenience and necessity of having a publicly-available dataset of this information. Because OSSmole is designed with collaboration in mind, these sorts of comparative results can be easily integrated into the OSSmole database, and then used in tandem with native OSSmole data or alone. As such, we have now fully integrated the Weiss data into the OSSmole database.

5. LIMITATIONS AND FUTURE WORK

There are, of course, limitations in the OSSmole project and our approach. Firstly, it is limited to data available online as a result of documented project activities. Certainly, these are not the only interactions FLOSS team members have. Thus while textual data like mailing lists, source control system history and comments, forums, and IRC chat logs could be included, OSSMole does not capture unlogged instant messaging or IRC, voice-over-IP or face-to-face interactions of FLOSS developers. Nor do we intend to store interviews or transcripts conducted by researchers which would be restricted by policies governing research on human subjects. We are also following the discussion about the ethical concerns of using data about open source projects closely [4].

There are also dangers in this approach which should be acknowledged. The standardization implied in this kind of repository, while desirable in many ways, runs the risk of reducing the valuable diversity that has characterized academic FLOSS research. We hope to provide a solid and traceable dataset and basic analyses which will support, not inhibit, interpretative and theoretical diversity. This diversity also means that research is not rendered directly comparable simply because analyses are based on OSSMole data or scripts. We are hopeful that OSSMole, by acting as a scaffold, will give researchers more time for such interesting work.

We will not be surprised to find parallel proposals or projects being prepared or implemented by others in the academic research community, although we are not aware of any detailed proposals or existing code at the time of writing.

It is quite likely that a functional hierarchy could develop between cooperating projects, something akin to the relationship between FLOSS authors and distributions, such as Debian or Red Hat and their package management systems (*apt* and *rpm*). For example, such an arrangement would allow groups to specialize in collecting and cleaning particular sources of data and others to concentrate on their compatibility. Certainly the existing communities of academics interested in FLOSS, such as http://opensource.mit.edu, are encouraged to be a source of data and support. Similarly, we would like to extend to people who donate data the ability to specify a license for that data.

One of the practical problems with spidering projects, like OSSmole, is keeping abreast of changes to the web site (or data source) being spidered. This is a known challenge with any spidering project, and was one of the main motivators for starting this project in the first place: if one research team can worry about spidering, saving, and aggregating the data, then that frees other teams to do other interesting analyses with the data, or to collect new data.

6. CONCLUSION

Researchers study FLOSS projects in order to better understand collaborative human behavior during the process of building software. Yet it is not clear that current researchers have many common frames of reference when they write and speak about the open source phenomenon. As we study open software development we learn the value of openness and accessibility of code and communications; OSSmole is a step towards applying that to academic research on FLOSS. It is our hope that by providing a repository of traceable and comparable data and analyses on FLOSS projects, OSSmole begins to address these difficulties and supports the development of a productive ecosystem of FLOSS research.

7. ACKNOWLEDGMENTS

Our thanks to the members of the ossmole-discuss mailing list, especially Gregorio Robles, Dawid Weiss, and Niti Jain.

8. REFERENCES

[1] Conklin, M. Do the rich get richer? The impact of power laws on open source development projects. Open Source Convention. (*OSCON '04)* (Portland, Oregon, USA, July 25-30, 2004). At http://www.elon.edu/facstaff/mconklin/pubs/oscon_revised.pdf.

[2] Crowston, K., Annabi, H., Howison, J., and Masano, C. Towards a portfolio of FLOSS project success metrics. In *Proceedings of the Open Source Workshop of the International Conference on Software Engineering (ICSE '04).*

[3] Crowston, K. and Howison, J. The social structure of free and open source software development. *First Monday 10,* 2 (February, 2005).

[4] El-Emam, K. Ethics and Open Source. In *Empirical Software Engineering 6*, 4 (Dec. 2001), 291-292.

[5] Howison, J. and Crowston, K. The perils and pitfalls of mining Sourceforge. In *Proceedings of the Workshop on Mining Software Repositories at the International Conference on Software Engineering (ICSE '04).*

[6] Krishnamurthy, S. Cave or community? An empirical examination of 100 mature open source projects. *First Monday 7*, 6 (June, 2004).

[7] Weiss, D. A large crawl and quantitative analysis of open source projects hosted on sourceforge. Research Report ra-001/05, Institute of Computing Science, Pozna University of Technology, Poland, 2005. At http://www.cs.put.poznan.pl/dweiss/xml/publications/index.xml

[8] Xu, J, Gao, Y., Christley, S. and Madey, G. A topological analysis of the open source software development community. In *Proceedings of 38th Hawaii International Conference on System Sciences (HICSS 05)* (Hawaii, USA, January 4-7, 2005).