Chapter II
FLOSSmole:
A Collaborative Repository for FLOSS Research Data and Analyses

James Howison
Syracuse University, USA

Megan Conklin
Elon University, USA

Kevin Crowston
Syracuse University, USA

ABSTRACT

This paper introduces and expands on previous work on a collaborative project, called FLOSSmole (formerly OSSmole), designed to gather, share and store comparable data and analyses of free, libre, and open source software (FLOSS) 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 FLOSS data and across research groups and analyses. The paper outlines current difficulties with the current typical quantitative FLOSS research process and uses these to develop requirements and presents the design of the system.

INTRODUCTION

This article introduces a collaborative project called FLOSSmole, 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. Our intent in developing FLOSSmole is to reduce duplication, and to promote compatibility
both across sources of FLOSS data and across research groups and analyses.

Creating a collaborative data and analysis repository for research on FLOSS is important because research should be as reproducible, extendable, and comparable as possible. Research with these characteristics creates the opportunity to employ meta-analyses, exploiting the diversity of existing research by comparing and contrasting results to expand our knowledge. Unfortunately, the current typical FLOSS research project proceeds in a way that does not necessarily achieve these goals. These goals require detailed communal knowledge of the many choices made throughout a research project. Traditional publication prioritizes results, but masks or discards much of the information needed to understand and exploit the differences in our data collection and analysis methodologies. FLOSSmole was originally designed to provide resources and support to academics seeking to prepare the next generation of FLOSS research. Since its inception, FLOSSmole has also been a valuable resource for nonacademics who are also seeking good data about development practices in the open source software industry.

BACKGROUND OF PROBLEM

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. This way of developing software leaves a freely available and, in theory at least, highly accessible trail of data upon which many academics have built interesting analyses about optimal organization of development teams, economics of building software in the commons, and the like. Yet, despite this presumed plethora of data, researchers often face significant practical challenges in using this data to construct a collaborative and deliberative research discourse. In Figure 1, we outline the research process we believe is followed in much of the quantitative literature on FLOSS.

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 article.) 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 it is hard to know which projects should actually be included. For example, are corporate-sponsored projects part of the phenomenon or not? Do single-person projects count? What about school projects?

Second, the projects themselves, 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 number of other repositories of varying sizes and focuses (e.g., CodeHaus, CPAN), and that many projects, including the well-known and much-studied Apache and Linux projects, prefer to use their own repositories and their own tools. This diversity of location effectively hides significant portions of the FLOSS world 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 e-mails, CVS, and bug tracker conversations — required to conduct certain comprehensive analyses.

Do the difficulties with census-taking mean that sampling would be more effective? By saying sampling we mean taking a random selection of a small (and thus more manageable) subgroup of projects that can, through careful selection,
represent the group as a whole. While this will go some way toward solving the manageability problem, sampling FLOSS projects is difficult for the same reason as census-taking: 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 (Conklin, 2004; Xu, Yongqin, Christley, & Madey, 2004). Selecting, even at random, from highly skewed distributions does not, in general, produce a representative sample. The difficulty of sampling is demonstrated in the tendency of FLOSS studies to first limit their enquiries 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 Krishnamurthy, 2002), neither of which is a satisfactory general technique. Selection of projects to study is further complicated by the fact that the public repositories contain a large number of projects that are dormant, relocated, or dead.

**BACKGROUND: DATA COLLECTION DIFFICULTIES**

Once the projects of interest have been identified and 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 (Howison & Crowston, 2004). Because the FLOSS repositories are usually maintained using a database back-end and a Web front-end, the data model appears straightforward to repro-

---

*Figure 1. The typical quantitative FLOSS research process (notice its noncyclical and noncollaborative nature)*
duce. 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-intensive 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. (Some, such as Freshmeat, provide a nightly build containing several text files with the majority of their information included.) And understandably so because it is not a costless process to make data-dumps available. Dumps can contain personally identifiable and financial information (as with the Sourceforge linked donation system) and so the data must be anonymized or otherwise modified to protect this information. 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 entirely. 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 may not make those dumps easily available. Other projects, such as the Sourceforge dump available from Notre Dame⁶, only provide the dumps to qualified academic researchers with editorial restrictions. It is unclear what effect this limitation will have on research efforts in the open source community, however, since research efforts are certainly not limited to academics.

Even when dumps are available it is necessary to interpret their database schema. 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, and there are inconsistencies between old and recently added data. The interpretation and transformation of this data into information semantically interesting to researchers is not a trivial process, and there is no reason to think that researchers will do this transformation in a consistent fashion.

Even pristine and labeled data from repositories is not sufficient because different repositories store different data. Different forges can have projects with the same names, different developers can have the same name across multiple forges, and the same developer can go by multiple names. Forges have different terminology for things like developer roles, project topics, and even programming languages. They often have fields which are named the same in multiple forges but which represent different data. Another problematic area is calculated fields, such as activity or downloads, for which there is incomplete publicly available information on their formula or correctness.

**BACKGROUND: DATA CLEANING DIFFICULTIES**

Once projects have been selected and the available data harvested, researchers must be confident that data adequately represent the activities of a project. For example, projects use repository tools to differing degrees. For example, 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 tracking systems using, perhaps, Bugzilla or RT software. Other projects host their activities outside Sourceforge but maintain a “placeholder” registration with little used mailing lists and out of date release information. It is very difficult, short of detailed
examination of each project, to know whether a project is fully using a tool. Thus, it is difficult to state with confidence that the data collected about that tool is a reasonable depiction of the project’s activities.

Complete accuracy is, of course, not 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 verification, a time-consuming process for individual researchers and projects, and one that is too frequently repeated by other researchers.

The conclusion we draw from this analysis is that each step of the typical FLOSS research process introduces variability into the data that 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. In addition, 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 difference in datasets. There is, rightly, more than one way to conduct research, and indeed this richness drives discovery. Rather, our critique is that the research community is currently unable to begin a meta-analysis phase or a reflective phase because the current process of FLOSS research introduces variability that is difficult to trace. The research process is also hampered by 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 to this problem.

**PROPOSING A SOLUTION: FLOSSMOLE**

The previous problem description motivates our attempt to build a system to support research into FLOSS projects. FLOSSmole (formerly OSSmole — the name was changed to reflect our inclusion of Free and Libre software in addition to open source software) is a central repository of data and analyses about FLOSS projects which have been collected and prepared in a decentralized manner. Data repositories have been useful in other fields; the presence of trusted datasets allows research communities to focus their efforts. For example, the TREC datasets have supported a community of information retrieval specialists facilitating performance and accuracy comparisons; the UMI machine learning repositories have been widely used in the development of new machine learning algorithms. There are numerous examples from biology and physics as well. The intention of FLOSSmole is to provide high-quality and widely used datasets, and to share standard analyses for validation, replication, and extension.

**REQUIREMENTS OF THE FLOSSMOLE SYSTEM**

Below we list some of our initial requirements for an optimal data and analysis clearinghouse for FLOSS data, and we note to what extent FLOSSmole has met each of these requirements. The next section expands on additional specific design attributes of the FLOSSmole system.

An optimal data and analysis repository for FLOSS data should be:

**Collaborative.** The system should leverage the collective effort of FLOSS researchers to reduce redundancies and to free researchers’ time to pursue novel analyses. Thus, in a manner akin to the BSD rather than the GPL licensing model, FLOSSmole 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 through interactive database queries. This should end the problem of data lockup, and will ease entry of new researchers with novel techniques. Freely available data also lowers the barriers to collegial replication and critique. FLOSSmole scripts and data are open-sourced and available to anyone via the FLOSSmole Sourceforge project page.

**Comprehensive and compatible.** Given the fragmentation of FLOSS project storage identified previously, the system should cover more than just one repository. The system should 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. (Currently FLOSSmole contains data from three repositories.) There exists the potential to develop an “interchange” format for FLOSSmole project collateral which projects themselves, which fear data and tool lock-in, might find convenient and useful as they experiment with new tools and repositories.

**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-points 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 validated against earlier analyses. By implementing these requirements, FLOSSmole is potentially a large advantage over individual research projects working with nonvalidated single datasets because it implements the “many eyeballs” FLOSS methodology for quality assurance.

**Able to support reproducible and comparable analyses.** It is desirable that data extracted from the database for transformation be exported with verbose comments detailing its origins 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, and in this goal, FLOSSmole excels. Optimally, the system should specify a standard application programming interface (API) for inserting and accessing data via programmed scripts. That would allow analyses to specify, using the API, exactly the data used.

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 comparison.

**ADDITIONAL DESIGN DETAILS**

The FLOSSmole data model is designed to support data collection, storage, and analysis from multiple free and open source forges in a way that meets the previously stated requirements. This section lists some additional design details we have made in implementing our FLOSSmole system.

FLOSSmole is able to take both spidered data and data inserted from a direct database dump. The raw data is time stamped and stored in the database, without overwriting any data previously collected, including data from the same project and from the same forge. 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, database environments, programming languages, natural languages, platforms, open source license type, operating systems, intended audiences, and the main project topics. Developer-oriented information includes number of developers, developer information (name, username, e-mail), and the developer’s role on the project. We have also collected issue-tracking data (mainly bugs), such as date opened, status, date closed, and priority. 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.

FLOSSmole is constantly growing and changing as new forges are added. And because data from multiple collectors are 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.

Likewise, it is a general rule that data are not overwritten when project details change; rather, one of the goals of the FLOSSmole 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 are difficult or impossible to access once they are no longer viewable from the repository’s front-end interface.

Access to the FLOSSmole project is two-pronged: both data and scripts are continually made available to the public under an open source license. Anyone can download the FLOSSmole raw and summary data for use in their own research projects or just to get information about the state of the art in open source development. The raw data are provided as multiple text file “data dumps” from the FLOSSmole 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 continually be contributed by researchers, and that the system will provide dynamic and up-to-date results rather than the static pictures that traditional publication unfortunately leaves us.

The scripts that populate the FLOSSmole 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 conference paper comparing and commenting on our spidering and summaries and beginning collaboration (Weiss, 2005). FLOSSmole expects and encourages contributions of additional forge data, and interested researchers should see the FLOSSmole project page and join the mailing list for information on how to contribute.

RESULTS

Because it is a regularly updated, publicly available data repository, FLOSSmole data have 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 FLOSSmole project regularly report the number of open source projects, the number of projects per programming language, the number of developers per project, and so forth. These sorts of descriptive data are useful for constructing “state of the industry” reports, or for compiling general statistical information about open source projects. The FLOSSmole collection methods are transparent and easily reproduced, so FLOSSmole 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 FLOSSmole data model and collection methodology was designed to support historical comparisons of this kind.

FLOSSmole data were used in a number of large-scale social network analyses of FLOSS project development. Crowston and Howison (2004) report the results of a SNA centralization analysis in which the data suggest that, contrary to the rhetoric of FLOSS practitioner-advocates, there is no reason to assume that FLOSS projects share social structures. Further FLOSSmole data were used in the preparation of Crowston, Howison, and Annabi (2006) 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. FLOSSmole makes available the full data and analysis scripts, which make these analyses fully reproducible, and, we hope, extendable.

FLOSSmole data were also used in a recent exploration of whether open source development teams have characteristics typical of a complex network (Conklin, Howison, & Crowston, 2004). This research investigated whether FLOSS development networks will evolve according to “rich get richer” or “winner take all” models, as 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 FLOSSmole data were 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.

Projects of a nonacademic nature are making use of FLOSSmole data as well. The Swik project from SourceLabs is a wiki-driven system for managing facts about other open source software projects. Swik uses FLOSSmole data to populate its initial list of projects. Working independently, the Swik team was able to download FLOSSmole data and put them to use immediately to save time and effort during their development process. By using a dataset that was freely available and for which the provenance of all data was known and validated, Swik was able to accelerate their development cycle.

LIMITATIONS AND FUTURE WORK

There are, of course, limitations in the FLOSSmole project and in our approach. First, we are limited to collecting data available online, and we are limited to collecting data gathered as a direct result of documented project activities. Of course, electronically documented project activities are not the only interactions FLOSS team members have, and even these activities are not always available for perusal by outside parties. Thus while textual data like mailing lists, CVS
comments, Forums, and IRC chat logs could be included. FLOSSmole does not aim to capture unlogged instant messaging, IRC, Voice-over-IP, or face-to-face interactions of FLOSS developers. Nor do we intend to store interviews or transcripts conducted by researchers that would be restricted by research ethics policies.

There are also dangers in this approach that should be acknowledged. The standardization implied in an academic repository, while valuable, 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 that 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 FLOSSmole data or scripts; the hard intellectual work remains and hopefully FLOSSmole, by supporting baseline activities, leaves us more time for such work.

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 (i.e., 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, we expect that the existing communities of academics interested in FLOSS, such as opensource.mit.edu, will be a source of data and support.

Finally, we must also consider privacy issues. There is some discussion in the research community about breaching developer privacy in a large system of aggregated data like ours (Robles, 2005), specifically in terms of uniquely identifying developers and analyzing their work products. FLOSSmole should have the ability to hash the unique keys indicating a developer’s identity. This effort will have to be researched, implemented, and documented for the benefit of our community.

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; FLOSSmole 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, FLOSSmole begins to address these difficulties and supports the development of a productive ecosystem of FLOSS research.

ACKNOWLEDGMENT

This research was partially supported by NSF Grants 03-41475, 04-14468, and 05-27457. Any opinions, findings, conclusions, or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation. The authors thank the FLOSS developers who contributed to the research.

REFERENCES


Missouri.


**ENDNOTES**

1. FLOSSmole: http://ossmole.sf.net
2. Sourceforge: http://www.sf.net
7. FLOSSmole: http://ossmole.sf.net
8. Swik: http://swik.net/

*This work was previously published in Int. Journal of Information Technology and Web Engineering, Vol 1, Issue 3, edited by E. Damiani and G. Succi, pp. 17-26, copyright 2006 by IGI Publishing (an imprint of IGI Global).*