Automating the Measurement of Open Source Projects

Daniel German
Department of Computer Science
University of Victoria
dmgerman@uvic.ca

Audris Mockus
Avaya Labs
Department of Software Technology Research
233 Mt Airy Rd., Basking Ridge, NJ 07920
audris@mockus.org

Abstract

The proliferation of open source projects raises a number of vital economic, social, and software engineering questions that are subject of intense research. Based on experience analyzing numerous open source and commercial projects we propose a set of tools to support extraction and validation of software project data. Such tools would streamline empirical investigation of open source projects and make it possible to test existing and new theories about the nature of open source projects. Our software includes tools to extract and summarize information from mailing lists, CVS logs, ChangeLog files, and defect tracing databases. More importantly, it cross-links records from various data sources and identifies all contributors for a software change. We illustrate some of the capabilities by analyzing data from Ximian Evolution project.

1. Introduction

The proliferation of open source projects raises a number of vital economic, social, and software engineering questions that are subject of intense research. Based on experience analyzing open source [3] and traditional projects [6] we propose a set of tools to support extraction and validation of open source software project data.

Any software project leaves traces of participant actions and decisions in mailing list archives, in version control and problem tracking databases. Previous work, e.g., [1, 4], has identified version control and problem tracking databases as a promising repository of information about a software project. It describes methods and tools to retrieve, process, and model such data at the fine level of Modification Requests (MRs or logical changes to software) in order to understand the relationships among process/product factors and key outcomes, such as, quality, effort, and interval.

We focus on a system to support the task of measuring and analyzing open source software data. Such tools would streamline empirical investigation of open source projects and make it possible to test existing and new theories about the nature of open source projects.

Our system retrieves, summarize, and validate data from mailing lists, CVS logs, ChangeLog files, and defect tracking systems based on Bugzilla: systems and sources that are commonly used in open source projects. Because open source projects usually involve individuals who do not meet face to face, almost all of the activity in the project leaves detectable traces in change logs, mailing lists, or problem reporting system. Such rich data provides excellent basis to study and compare open source software projects. Unfortunately, extracting, cleaning and validating, and drawing conclusions from such data poses formidable challenges because the data sources are not designed as measurement tools, and the tools as well as practices of using the tools vary from project to project. As part of system validation methodology we used a *cleanroom* approach where two authors has created a totally independent implementation of the basic set of tools. An verified their correctness by comparing the resulting output. This has allows us better to understand the pitfalls when creating a repeatable process of extracting valid information from these diverse data sources. SoftChange can be found at http://sourceforge.net/projects/sourcechange.

We start by briefly describing a common open source project environment exemplified by Ximian Evolution in Section 2, then we outline the architecture of SoftChange system in Section 3, and conclude by illustrating some of the validation, cross-linking, and measurement techniques in Section 4.

2. Background

Ximian Evolution evolved from the GNOME project as its groupware suite based around a mail client. The project started in the beginning of 1998, and it is currently composed of approximately 185k lines of code. Evolution recently received the “2003 LinuxWorld Open Source Prod-

uct Excellence Award" in the category of "Best Front Office Solution". One of the objectives of Evolution is to provide a free software product with functionality similar to Microsoft Outlook or Lotus Notes[7].

The development environment includes CVS (used for version control) Bugzilla (used to track defects), one mailing list for developers and one for users, and a collection of documentation pages hosted in the Ximian and GNOME developers web sites.

Core developers have write access to CVS whereby they can submit modifications via the CVS commit command. The commit command is usually preceded by update command, that merges any changes committed by other developers since the last update. CVS prevents committing files without an update if such modifications exist.

When a developer commits a group of files to the repository, he or she often modifies the relevant ChangeLog file with a description of the current change, and this description is usually repeated as the log message, which is recorded by CVS, along with the data corresponding to the commit. The commit triggers an email message to the GNOME cvs-commits mailing list, which includes all the details of the transaction: who made it, when, files modified, and the log message.

Each CVS commit often corresponds to a logical change which we refer to as MR and contains one or more deltas (revisions to individual files). A description of good CVS practices [8] provides more details on the process of using CVS, but it is not clear if all of the practices described there are wide-spread.

Usually, the ChangeLog and the CVS log messages indicate the nature of the change, lists PR numbers from Bugzilla, and often include a URL pointing to the change and/or to the PR.

3. Architecture

SoftChange is a collection of scripts intended to retrieve, process, correlate and validate the change information from a typical open source project. By functionality the SoftChange can be broken into retrieval of raw data, validation, and summarization parts. Each data processing stage produces successively better quality data on software software changes, however links to raw data are retained to allow automatic and manual validation.

Retrieval of raw data Software changes are obtained from sources listed below and are represented as a table of delta including login, timestamp, comment, file, and other attributes available for each particular source of data.

Parsing CVS mailing lists. This information is relatively easy to download and process. The main drawback is that CVS could be configured in different ways to produce email messages, and, often these messages do not contain full information from CVS logs.

Getting logs from CVS systems. This is a bit more time and network-bandwidth consuming but it obtains all the detail about changes.

Getting code changes from CVS system. This is the most involved operation, but it allows fully to reconstruct the code evolution in each file and is necessary to obtain code related measures: does the change involve comments or code, what function was changed, etc.

Extracting changes from ChangeLog files. This is least precise, but may be the only option in systems that to not use version control.

Problem reports (PRs) are obtained by retrieving the web pages for each PR and extracting the available attributes of each PR, including its ID, description, status history (identity of individuals, dates, and status changes).

Augmenting and validating of raw data

Removing system generated artifacts. This mostly concerns eliminating branch delta from CVS logs where the code is not modified. Another common problem is copying of CVS repository files. There we eliminate duplicate delta from two or more files that had the same origin or are included in several modules. In the gnome CVS repository this was a frequent problem with 13 percent of all delta involving such files. Six files belong to 212 different modules!

Other important tasks include matching delta obtained from various sources, matching delta to PR numbers, determining contributors for each delta, and converting deltas into MRs.

Finally, change measures (see, e.g, [5] and summary benchmarks (see, e.g., [3]) are produced.

Some of the infrastructure to process the semi-structured text in software changes involves detection/parsing of dates, bug numbers, and contributors and link between various spellings of names and email.

4. The Evolution of Evolution

We have chosen to use Evolution to illustrate some of the capabilities of SoftChange. We focus on the data provided by CVS logs and the CVS commit mailing list. Our data includes changes to the CVS repository from April 1998 to January 2003.

4.1. CVS MRs

In CVS a delta is a change to a single file. It is produced by the commit command executed by a developer. A single commit usually includes several files that correspond to a particular logical operation (we call this operation an MR).

It is worth noting that commits may involve several unrelated MRs, however each such MR would typically have different comments. A MR provides insight on how files interrelate, the way developers work, and the extend of a given change. Unfortunately CVS does not keep track of MRs. As a consequence, SoftChange analyzes the deltas and gathers them into MRs. All deltas from a single developer with the same comment and committed within short (two minute) interval are grouped into a single MR.

Figure 4.1 shows the number of MRs per month for Evolution. The plot also shows the different releases in the project. There are several interesting observations from this graph. First, the number of MRs dramatically increased when Ximian is created. Second, there was a surge in the number of minor releases and MRs in the six months that preceded the release of Version 1.0, and a relatively flat development afterward. Our hypothesis is that after version 1.0, developers have been spending more time fixing bugs. We have also seen that number of added lines to the project goes down, indicating activity related to defect fixing, not introduction of new functionality. We need to correlate Bugzilla data with CVS data to get further evidence.

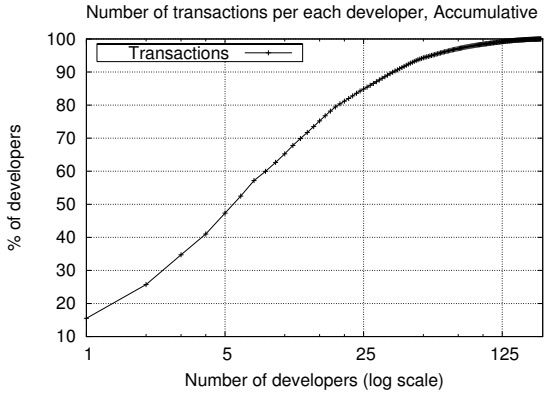

Figure 2. From a total of 196 developers, 5 account for 47% of the MRs, while 20 account for 81% of the MRs, and 55 have done 95% of them.

We have identified a total of approximately 18000 MRs. As figure 3 shows, 80% of them include 4 files or less, with a minimum of 1 file and a maximum of 650. The largest MR corresponds to the change of the license to the GPL version 2 on 2001/10/27 (in that day a single developer modified a total of 1257 files, the largest number of files modified in a single day by a single developer). Similarly, 2001/06/23 was the second busiest day (in terms of files modified), when 688 files were altered to reflect that Helixcode had changed its name to Ximian.

There is a relatively large number of MRs involving only

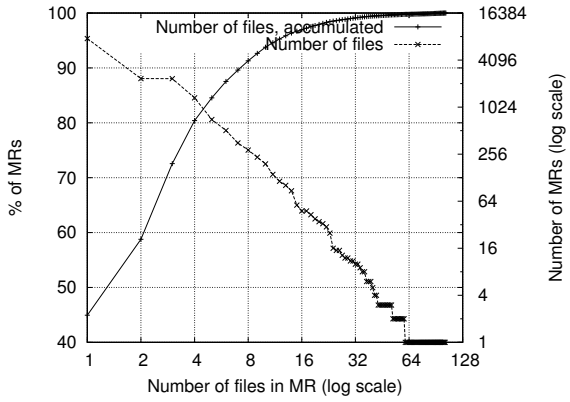

Figure 3. 80% of MRs include 4 or less files.

one file (22%). A preliminary analysis shows that most of these changes are of two types: a) files which were overlooked in a previous MR, and committed minutes later; and b) minor corrections, such as fixing spelling mistakes. Further analysis is needed to corroborate this hypothesis.

Figure 2 shows that 5 developers (from a total of 196) account for 47% of the MRs, while 20 have done 81% of them. Here we used CVS logins to identify developers. Since some of the code is provided by developers without CVS commit access, the reported numbers underestimates the number of contributors. While SoftChange has capabilities to identify contributors, due to time constraints we do not report the data here. In table 1 we list the top 11 developers. The top 10 appear to be Ximian employees or consultants (see http://primates.ximian.com/). This fact corroborates our hypothesis that private companies (such as RedHat, Ximian, and Eazel) have had a very important effect in the development of the GNOME project [2]. In that respect it is similar to Mozilla project where top developers work for Netscape (see, e.g., [3]).

4.2. ChangeLogs

ChangeLog files are an important source of information about the development and evolution of a project. As the GNU ChangeLog standards indicate, the ChangeLog explains how earlier versions of software were different from the current version. The Evolution developers are fairly consistent in their modifications to the ChangeLog files. From all MRs involving 2 or more files, 93% include a modification of a ChangeLog.

Table 2 shows the 10 most modified files, 8 of them are ChangeLogs. ChangeLogs (and CVS logs) provide insight on patches submitted by developers without a CVS account, as developers are expected to be careful to give credit to the patch submitter in the corresponding ChangeLog entry.

SoftChange produces the list of delta from CVS logs,

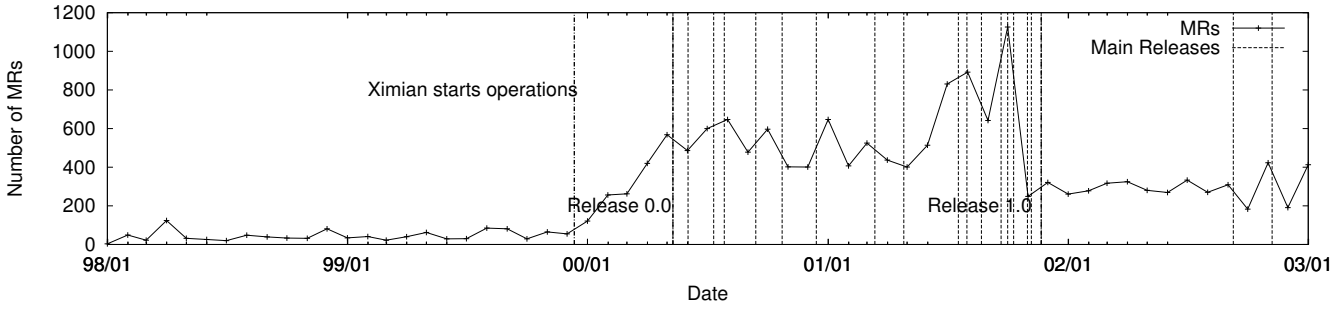

Figure 1. Number of MRs per month. There is a significant increase in activity after Ximian starts operations (originally known as Helixcode, and later renamed to Ximian), and the largest activity coincides with the release of version 1.0.

Percent.	Accum.	Userid
15.52%	15.52%	fejj
10.19%	25.72%	ettore
9.00%	34.72%	danw
6.31%	41.02%	clahey
6.30%	47.32%	zucchi
5.15%	52.47%	jpr
4.70%	67.17%	toshok
2.79%	59.96%	federico
2.70%	62.66%	peterw
2.55%	65.25%	unammx
2.59%	67.80%	iain
32.20%	100.00%	*rest of developers*

Table 1. % of MRs committed to the project by the top 11 most active developers. Only iain does not seem to be a Ximian employee.

Deltas	Percent.	Accum.	File
2725	3.72%	3.72%	mail/ChangeLog
1853	2.53%	6.25%	camel/ChangeLog
1768	2.41%	8.66%	calendar/ChangeLog
1404	1.92%	10.57%	addressbook/ChangeLog
1268	1.73%	12.30%	ChangeLog
1239	1.69%	13.99%	shell/ChangeLog
1166	1.59%	15.58%	po/ChangeLog
638	0.87%	16.45%	configure.in
517	0.71%	17.16%	composer/ChangeLog
460	0.63%	17.79%	mail/mail-callbacks.c

Table 2. Top 10 most modified files. ChangeLogs clearly take the lead. As its name implies, mail-callbacks.c contains the callbacks of the mail client, hence the frequency at which it is modified.

CVS email, and ChangeLogs, to allow assessment of quality for each data source. Different sources contain different amounts of information and comparison is non trivial. For example, CVS email for evolution module in 2002 has 11875 deltas, and CVS logs for the same period contain 12111 deltas. While there are a total 12711 CVS log delta, we exclude 599 deltas that do not modify files (branch delta). Such selection is not possible with email archives since they do not keep information about the extent of modification. The remaining of the deltas are missing from files that that were not part of Evolution module initially, so CVS mail archives indicate a different module for these deltas.

4.3. Files

Figure 4 shows the distribution of files according to their types. It includes files which have been erased but remain in the repository (a place known as the attic) in case they are needed later. The source files are mainly C files (.c and .h). Table 3 shows the number of deltas per file extension. As it is expected, most of the changes are made to source code files (there are 698 .c and 617 .c files plus 498 .c and 453 .h files in the attic); .h files tend to be modified 1/3 the number of times .c files are modified). ChangeLog files are followed by makefiles (.am), and then translations (.po). .ics files correspond to information about time zones needed by Evolution at run-time (they are 394 of them, none in the attic). Documentation is typically written in SGML (there are 80 different .sgml files plus 148 in the attic).

Most MRs tend to concentrate on few hot spots. After ChangeLog files, .c files are the most modified. The module with the largest number of modifications is `mail`. This is not surprising, since the mail client represents the core functionality of Evolution. For Figure 5, we only took into account .c and .h files; the figure shows that 12 files (approx. 0.5% of the total) account for 10% of the deltas.

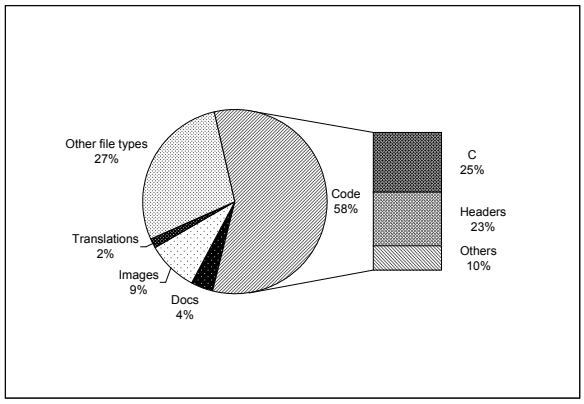

Figure 4. Number of files per type. Includes those in the Attic.

Deltas	Percent.	Accum.	Extension
29420	40.13%	40.13%	.c
15360	20.95%	61.09%	ChangeLog
9897	13.50%	74.59%	.h
3611	4.93%	79.52%	.am
2754	3.76%	83.27%	.po
1899	2.59%	85.86%	.ics
1839	2.51%	88.37%	.sgml
1413	1.93%	90.30%	.in
860	1.17%	91.47%	.png
783	1.07%	92.54%	.xml
5467	7.46%	100.00%	*other types*

Table 3. Deltas per file extension. C files (.h and .c) and ChangeLog modifications account for almost 75% of modifications.

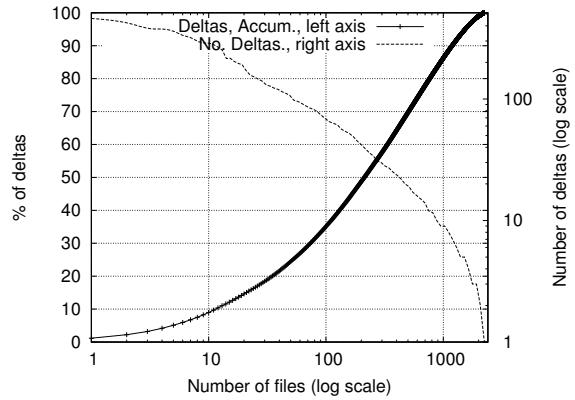

Figure 5. 2240 .c and .h files have been modified in 39317 deltas. 12 files account for 10% of the deltas; and less than 100 files have been involved in more than 100 deltas.

4.4. Summary and Further Work

We are still in the early stages of developing SoftChange. The project itself is created in an open source fashion to facilitate usage and contributions. Numerous topics including definition of automating production of change and project measures, data quality summaries code measurement and discussion list analysis tools are on the agenda. Our goal is to provide a framework that supports the comparison of open source projects (such as Mozilla, Apache, gcc, GNOME, KDE, etc) that rely upon CVS, Bugzilla, and mailing lists as their main repository of information about the project. We have illustrated some of the capabilities using Ximian Evolution as an example.

Acknowledgments

This work has been partially funded by ASI and NSERC.

References

[1] D. Atkins, T. Ball, T. Graves, and A. Mockus. Using version control data to evaluate the impact of software tools: A case study of the version editor. *IEEE Transactions on Software Engineering*, 28(7):625–637, July 2002.

[2] D. M. German. The evolution of the GNOME Project. In *Proceedings of the 2nd Workshop on Open Source Software Engineering*, 2002.

[3] A. Mockus, R. T. Fielding, and J. Herbsleb. Two case studies of open source software development: Apache and mozilla. *ACM Transactions on Software Engineering and Methodology*, 11(3):1–38, July 2002.

[4] A. Mockus and L. G. Votta. Identifying reasons for software change using historic databases. In *International Conference on Software Maintenance*, pages 120–130, San Jose, California, October 11-14 2000.

[5] A. Mockus and D. M. Weiss. Predicting risk of software changes. *Bell Labs Technical Journal*, 5(2):169–180, April–June 2000.

[6] A. Mockus and D. M. Weiss. Globalization by chunking: a quantitative approach. *IEEE Software*, 18(2):30–37, March 2001.

[7] E. Perazzoli. Ximian Evolution: The GNOME Groupware Suite. http://developer.ximian.com/articles/whitepapers/evolution/, 2001.

[8] V. Venugopalan. Software configuration management for open source projects. http://ibiblio.org/gferg/ldp/SCM-OpenSource/.