

# Challenges in Software Evolution: the Libre Software Perspective\*

Jesus M. Gonzalez-Barahona, Gregorio Robles and Israel Herraiz  
GSyC, Universidad Rey Juan Carlos (Madrid, Spain)  
{jgb, grex, herraiiz}@gsyc.esct.urjc.es

## Abstract

*Libre (free, open source) software is providing huge quantities of data suitable to be used in studies of software evolution. Many different aspects of its development process can be studied from data available in public repositories, ranging from the source code in release archives to mailing lists, bug report systems and version control systems. There are already several software evolution studies using that information, leading to a better understanding of libre software projects. However, those projects also pose some new challenges which have to be addressed in the near future, and which could in some cases be also relevant for proprietary software environments. In the software evolution field some of those challenges are: the manipulation and analysis of the high quantity of data available for thousands of projects, which requires new methods and tools; the jump from studying sources of information in isolation (which is currently the state of the art) to the integration of data from various sources; and the importance of considering characteristics of developers as a parameter of software evolution.*

**Keywords:** software evolution, libre software engineering, mining software repositories

## 1. Introduction

Libre software<sup>1</sup> constitutes a good benchmark for comparing software analysis tools and techniques, and an excellent framework for studying software evolution. Projects usually maintain large quantities of information available in public repositories, including version control systems, software package repositories, bug tracking databases, web-based discussion forums, mailing list archives, etc. Most of this information (but not all) can be retrieved and analyzed automatically from an historical perspective, making evolutionary analyses possible [3, 14]. German et al. and our research group have presented during the latest years some ways to exploit those data from a number of corners, in an almost fully automatic way [5, 17].

However, beyond these studies there is a wide field that should be explored, in order to better understand how software evolves, and to learn lessons that could be applied both to libre and proprietary

---

\*This work has been funded in part by the European Commission, under the CALIBRE CA, IST program, contract number 004337, by the Universidad Rey Juan Carlos under project PPR-2004-42 and by the CICYT under project TIN2004-07296.

<sup>1</sup>Through this paper we will use the term “libre software” to refer to code that conforms to either the definition of “free software” (according to the Free Software Foundation) or of “open source software” (according to the Open Source Initiative).

software projects. This position paper presents some challenges related to libre software that, in the opinion of the authors, should be addressed by the software evolution community in the near future.

## **2. Handling and analysis of huge quantities of data**

The problem of getting the “global picture” of libre software development is still an open issue. Cases of detailed studies of large collections of projects (in the hundreds), crossing information from several different kinds of repositories, or analysis of historic patterns in different projects over long periods of time, are still rare.

Up to now, studies tend to focus on a small number of projects, or on certain aspects of the information about them. In the libre software world, the most studied ones are Mozilla [1, 2, 13, 15], Linux [6, 15, 22] and Apache [13]. Some of these studies have already started to consider data in time, doing simple evolutionary analysis.

Unfortunately, getting the global picture poses difficult problems, due to the scale: managing huge quantities of data with inconsistencies, missing information, inaccuracies, and different representations.

Such an amount of data can not be analyzed just with classical statistical techniques such as linear regression. Many of the correlations are clearly non-linear; and in many cases the distribution of events in the gathered data is still unknown. Therefore, other processing techniques, such as soft computing (support vector machines, neural networks or fuzzy additive models) or data mining are needed.

Our research group has been strongly focused on the data mining area, designing and implementing a set of tools that provide several views of the software development process, as will be described in more detail in section 5. Other good candidates for holistic analyses in the future are broad studies that investigate code reuse or the existence of technical dependencies for thousands of projects, and how it evolves over time.

Regarding soft computing tools, we are exploring them to build predictor models which can describe the behavior of large libre software projects. We have applied these techniques to a case study [11], checking whether classical linear statistics may help in extracting some conclusions from these data. One of our goals is to use these techniques to obtain a characterization of software systems based on several criteria that go beyond (but include) software evolution.

## **3. Integration of data from various sources**

Libre software provides the possibility to study the many traces left behind during the development process. But unfortunately, the information is not structured in such a way that makes it easy to interconnect traces related to the same artifact but coming from different sources. Therefore, methods for the integration of the information gathered from different places have to be found. In this respect, we are exploring several possibilities, which could also be used in union:

- Integration through artifacts. We define a granularity level (project, directory, file, class, method or even line) at which we identify all the actions related to every artifact.
- Integration by identification of traces from other sources. For instance we find bug report identifiers in version repository logs. Some other research groups already have started to walk in this direction [1, 2].

- Integration at the developer level. Actors that take part in the development process can be identified, and their activity tracked in the various sources of information, even when different identities are used (several e-mail address, logins, and even different spelling of real names). We have introduced some ideas about this while preserving developer anonymity in [16].

## 4. Developers and cost estimation

In traditional software development environments, software developers have tight schedules and a fixed number of hours per week to devote to a given set of tasks (programming/maintaining software being just one of them). But this is not the case in libre software, where the availability and behavior of human resources is another research dimension in itself, specially in the case of voluntary contributions. Because of the growing importance of libre software systems in traditional environments, this becomes an important matter of study. In other words, in libre software projects the evolution of software depends a lot on the evolution of developers.

A good starting point for these studies is the tracking of the activity of actors in several public information sources. With those data, we can walk a path that could lead us to the calculation and estimation of productivity parameters, and software production costs, which should be part of the research agenda for the near future.

Some other issues related to developers are also important, such as their integration process into the project, how they gain software comprehension, or how do they organize themselves. Some studies have already shown that some developer territoriality may exist [4], and how software is usually read [23], although more general studies should be performed in the future.

Studying development at the team (often referred as community) level may also provide interesting results. We have started to introduce some methods in this respect, for instance by having indexes that may give some insight about the communities that surround libre software projects, and how do they evolve over time. Just as an example, our research group has started to quantify the half-life [20] of the *core* team for some libre software projects (defined as the time required for a certain group of contributors to fall to half of its initial population).

## 5. Our research lines

Our research group at the Universidad Rey Juan Carlos has been analyzing libre software from an empirical point of view for several years now. Due to the large amount of available data and the huge amount of projects we plan to study, we depend on automating most processes. Therefore, we have built some tools that allow us to obtain some preliminary results, and extract conclusions. These tools may serve as a starting point for addressing the challenges presented in this position paper. All of them have in common that they are libre software themselves, so that independent research groups have the possibility to use, and enhance them, comparing the results. Below we offer a list of the tools that we are using, and some of the results we have obtained up to the moment:

- *CVSAnalY*<sup>2</sup> [21] is a CVS and Subversion repository analyzer that extracts information from a repository. *CVSAnalY* has a web interface<sup>3</sup> that allows to browse through results and figures. We

---

<sup>2</sup><http://cvsanaly.tigris.org/>

<sup>3</sup><http://libresoft.urjc.es/cvsanal/>

have used this to investigate the evolution of *core* groups in time in libre software projects [9] as well as the (social and technical) structure of some libre software projects in time, as it is the case of Apache [7], GNOME and KDE [12].

- *GlueTheos*<sup>4</sup> [18] is a tool that retrieves periodical snapshots from versioning systems, calls several external tools (including some mentioned in this list) to perform several quantitative analysis, stores the results into a database and proceeds with some higher level analysis.
- *DrJones* can be used to perform empirical analysis of software archaeology for software archived in a versioning repository. With it we have started to study how old the current code base of some very well-known libre software applications [19].
- *CODD*<sup>5</sup> [17] (and its newer version, *pyCODD*) is an authorship extraction tool that looks for copyright statements, e-mail addresses or RCS login traces in source code files.
- *SLOCCount*<sup>6</sup> [8] is a tool that performs advanced counting of physical source lines of source code. It uses several heuristics to determine the programming language, to filter out comments, etc. SLOCCount has been used to analyze the evolution of the Debian GNU/Linux distribution over the last 5 years [10]. Recent analyses on Debian include the permanence of packages and developers in time [20].

## 6. Conclusions

In this position paper we have shown some of the challenges that should be addressed by the research community during the next years in a very specific field: the evolution of software in libre software projects. Those challenges are: the manipulation and analysis of large amounts of data, the integration of information obtained from various sources that refer to the same artifacts, and the evolution of the developers who participate in the creation of software. Later, we have presented our research lines, with special attention to the tools that we have built, and the results that we have been obtained so far. We expect that those tools will help in fulfilling the research agenda we have briefly presented.

## References

- [1] G. Antoniol, M. D. Penta, H. Gall, and M. Pinzger. Towards the integration of cvs repositories, bug reporting and source code meta-models. In *2nd Workshop on Software Evolution through Transformations: Model-based vs. Implementation-level Solutions*, Rome, Italy, 2004. Electronic Notes in Theoretical Computer Science, Elsevier.
- [2] M. Fischer, M. Pinzger, and H. Gall. Populating a release history database from version control and bug tracking systems. In *Proceedings of the International Conference on Software Maintenance*, pages 23–32, Amsterdam, The Netherlands, September 2003.
- [3] H. Gall, M. Jazayeri, R. Klösch, and G. Trausmuth. Software evolution observations based on product release history. In *Proc Intl Conf Software Maintenance*, pages 160–170. IEEE Computer Society, 1997.

---

<sup>4</sup><http://libresoft.urjc.es/index.php?menu=Tools&Tools=GlueTheos>

<sup>5</sup><http://codd.berlios.de>

<sup>6</sup>We use an enhanced version of the one authored by David A. Wheeler: <http://www.dwheeler.com/sloccount/>

- [4] D. German. An empirical study of fine-grained software modifications. In *Proceedings of the International Conference in Software Maintenance*, Chicago, IL, USA, 2004.
- [5] D. German and A. Mockus. Automating the measurement of open source projects. In *Proceedings of the 3rd Workshop on Open Source Software Engineering*, Portland, Oregon, 2003.
- [6] M. W. Godfrey and Q. Tu. Evolution in Open Source software: A case study. In *Proceedings of the International Conference on Software Maintenance*, pages 131–142, San Jose, California, 2000.
- [7] J. M. Gonzalez-Barahona, L. Lopez-Fernandez, and G. Robles. Community structure of modules in the apache project. In *Proceedings of the 4th Workshop on Open Source Software Engineering*, Edinburg, Scotland, UK, 2004.
- [8] J. M. Gonzalez-Barahona, M. A. Ortuño Perez, P. de las Heras Quiros, J. Centeno Gonzalez, and V. Matellan Olivera. Counting potatoes: the size of Debian 2.2. *Upgrade Magazine*, II(6):60–66, Dec. 2001.
- [9] J. M. Gonzalez-Barahona and G. Robles. Unmounting the "code gods" assumption. Technical report, Universidad Rey Juan Carlos, 2003.
- [10] J. M. González-Barahona, G. Robles, M. Ortuño Pérez, L. Rodero-Merino, J. Centeno González, V. Matellan-Olivera, E. Castro-Barbero, and P. de-las Heras-Quirós. Analyzing the anatomy of GNU/Linux distributions: methodology and case studies (Red Hat and Debian). In S. Koch, editor, *Free/Open Source Software Development*, pages 27–58. Idea Group Publishing, Hershey, PA, USA, 2004.
- [11] I. Herraiz, G. Robles, and J. M. Gonzalez-Barahona. Towards predictor models for large libre software projects, 2005. Not published yet. Available from the authors at request.
- [12] L. Lopez, J. M. Gonzalez-Barahona, and G. Robles. Applying social network analysis to the information in CVS repositories. In *Proc of the Intl Workshop on Mining Software Repositories*, pages 101–105, Edinburg, UK, 2004.
- [13] A. Mockus, R. T. Fielding, and J. D. Herbsleb. Two case studies of Open Source software development: Apache and Mozilla. *ACM Transactions on Software Engineering and Methodology*, 11(3):309–346, 2002.
- [14] A. Mockus and L. G. Votta. Identifying reasons for software changes using historic databases. In *Proceedings of the International Conference on Software Maintenance*, pages 120–130, October 2000.
- [15] J. W. Paulson, G. Succi, and A. Eberlein. An empirical study of open-source and closed-source software products. *Transactions on Software Engineering*, 30(4), April 2004.
- [16] G. Robles and J. M. Gonzalez-Barahona. Developer identification methods for integrated data from various sources, 2005. Submitted to the 2nd International Workshop on Mining Software Repositories. Pending acceptance. Not published yet. Available from the authors at request.
- [17] G. Robles, J. M. Gonzalez-Barahona, J. Centeno-Gonzalez, V. Matellan-Olivera, and L. Rodero-Merino. Studying the evolution of libre software projects using publicly available data. In *Proceedings of the 3rd Workshop on Open Source Software Engineering*, pages 111–115, Portland, Oregon, 2003.
- [18] G. Robles, J. M. Gonzalez-Barahona, and R. A. Ghosh. Gluetheos: Automating the retrieval and analysis of data from publicly available software repositories. In *Proceedings of the International Workshop on Mining Software Repositories*, pages 28–31, Edinburg, Scotland, UK, 2004.
- [19] G. Robles, J. M. Gonzalez-Barahona, and I. Herraiz. An empirical approach to Software Archaeology, 2005. Not published yet. Available from the authors at request.
- [20] G. Robles, J. M. Gonzalez-Barahona, and M. Michlmayr. Evolution of volunteer participation in libre software projects: evidence from Debian. In *Proceedings of the 1st International Conference on Open Source Systems*, Genova, Italy, July 2005. To appear.
- [21] G. Robles, S. Koch, and J. M. Gonzalez-Barahona. Remote analysis and measurement of libre software systems by means of the CVSanaly tool. In *Proceedings of the 2nd ICSE Workshop on Remote Analysis and Measurement of Software Systems (RAMSS)*, Edinburg, Scotland, UK, 2004.
- [22] S. R. Schach, B. Jin, D. R. Wright, G. Z. Heller, and A. J. Offutt. Maintainability of the Linux kernel. *IEE Proceedings–Software*, 149:18–23, 2002.
- [23] D. Spinellis. *Code Reading: The Open Source Perspective*. Addison Wesley Professional, 2003.