

1. ACTIVIDAD INVESTIGADORA

1.A.
CALIDAD Y DIFUSIÓN DE
RESULTADOS DE LA ACTIVIDAD
INVESTIGADORA

1.A.1.
**PUBLICACIONES CIENTÍFICAS
INDEXADAS DE ACUERDO CON UN
ÍNDICE DE CALIDAD RELATIVO**

Libre Software for Computer Science Classes

In 1993, the authors began using 386BSD to teach computer science classes at Madrid's Carlos III University, in Spain. Seven years later, NetBSD and GNU/Linux are the operating systems of choice for several of the University's computer science teaching laboratories.

Jesús M. González-Barahona, Pedro de-las-Heras-Quirós, José Centeno-González, Vicente Matellán-Olivera, and Francisco J. Ballesteros, *King Juan Carlos University, Spain*

During 1991 and 1992, the landscape of libre software, and of software development in general, was ready for a change. In two different communities, two very exciting events were taking place: 386BSD, a libre derivative of the BSD code, was born, and Linus Torvalds distributed the first versions of the Linux kernel. Soon after, the software community had two libre operating systems at its disposal (for some basic references to online information on libre products and distribution, see the “Online Resources” and “Terminology” sidebars).

In late 1992 at Carlos III University, we were introducing new computer science studies and planning a distributed systems course. We were looking for a software platform suitable for practical lectures on distributed systems (our practical lectures take place in a computer lab and usually involve a programming exercise). Unfortunately, we couldn't afford workstations—we could only share a PC laboratory running MS-DOS, used to teach other computer science subjects. Therefore, we were pleased to discover that 386BSD could run on those PCs and already had the complete environment we needed for the practical lectures. We decided to give the new, exciting libre operating systems a chance—

both GNU/Linux and 386BSD were reasonably stable platforms.

History of the Unix Lab

After a testing period, we established a stable environment that several students helped us maintain and improve. When the semester started, a small group of around 20 students used 386BSD to learn to build client-server applications. The experiment was a success. The students, using the environment, were productive and fulfilled the goal we had set: to build simple RPC-based applications. 386BSD exposed them to a system similar to Unix workstations but at a fraction of the cost and with many new features—such as access to source code.

There are also institutional barriers to libre software. Many departments within universities are reluctant to use or directly oppose libre software, not on technical grounds, but based on their unfamiliarity with the software. Though less common, we also experienced problems regarding a lack of support for certain hardware devices.

Another problem is that finding technical staff well trained in the installation and administration of libre software is still not easy.

Many of these disadvantages are not directly related to libre software characteristics, but rather to its current situation and perception. Should this situation change in the future, many of these problems will disappear or else turn into advantages.

In October 1999, our team moved to the King Juan Carlos University, and we have already started implementing the lessons learned at the Carlos III University. In addition, we are exploring the suitability of libre software for teleteaching and for implementing virtual campus facilities. We are also holding discussions in which we explain to

students the benefits of a lab based on libre software.

Both in Carlos III University and King Juan Carlos University, we have helped create Linux users groups, which are a good source of informal support for both students and teachers using libre software. In those groups, we can also find well-trained technical staff to help maintain the lab.

We are very interested in the use of libre software for teaching computer science and welcome any comments you have about your experiences in this area. ☺

Acknowledgments

This article, and the experiences it describes, has only been possible thanks to the outstanding work of hundreds of libre software developers. We are very grateful to all of them.

References

1. G.R. Wright and W.R. Stevens, *TCP/IP Illustrated*, Addison-Wesley, Reading, Mass., 1995.
2. M.K. McKusick et al., *The Design and Implementation of the 4.4 BSD Operating System*, Addison-Wesley, Reading, Mass., 1996.

IEEE COMPUTER SOCIETY

Career Service Center

- Certification
- Educational Activities
- Career Information
- Career Resources
- Student Activities
- Activities Board

<http://computer.org>

Advance your career
Search for jobs
Post a resume
List a job opportunity
Post your company's profile
Link to career services

<http://computer.org/careers/>

Beyond source code: The importance of other artifacts in software development (a case study)

Gregorio Robles^{a,*}, Jesus M. Gonzalez-Barahona^a, Juan Julian Merelo^b

^a *Grupo de Sistemas y Comunicaciones, Departamento de Ingeniería Telemática y Tecnología Electrónica, Universidad Rey Juan Carlos, Tulipan s/n, 28933 Mostoles, Madrid, Spain*

^b *Grupo Geneura, Universidad de Granada, Campus Aynadamar, Daniel Saucedo Aranda s/n, 18071 Granada, Spain*

Received 24 February 2006; accepted 25 February 2006

Available online 21 April 2006

Abstract

Current software systems contain increasingly more elements that have not usually been considered in software engineering research and studies. Source artifacts, understood as the source components needed to obtain a binary, ready to use version of a program, comprise in many systems more than just the elements written in a programming language (source code). Especially when we move apart from systems-programming and enter the realm of end-user applications, we find files for documentation, interface specifications, internationalization and localization modules and multimedia data. All of them are source artifacts in the sense that developers work directly with them, and that applications are built automatically using them as input. This paper discusses the differences and relationships between source code (usually written in a programming language) and these other files, by analyzing the KDE software versioning repository (with about 6,800,000 commits and 450,000 files). A comprehensive study of those files, and their evolution in time, is performed, looking for patterns and trying to infer from them the related behaviors of developers with different profiles, from where we conclude that studying those ‘other’ source artifacts can provide a great deal of insight on a software system.

© 2006 Elsevier Inc. All rights reserved.

Keywords: Mining software repositories; Source code analysis; Source code management systems

1. Introduction

Software systems have evolved during the last decades from command-line programs to huge end-user applications full of graphics and multimedia elements. Besides, a piece of software has nowadays to be adapted to different cultural environments (language and notational conventions) if it aims to become mainstream. All this has caused that software development is an endeavor that is no longer carried out only by software developers. In many cases it has become an activity that requires the coordinated work of several groups, with different backgrounds and that perform different tasks such as internationalization and local-

ization (from now on *i18n*, short for *internationalization* and *l10n*, short for *localization*¹), graphic design, user interface design, writing of technical and end-user documentation and creation of multimedia elements.

During the software construction process these diverse elements are handled together, conforming an integral body that has to be developed, managed and maintained. Despite this new environment, ‘classical’ source code analysis is still focused on the output of the work performed by software developers: source code written in a programming language. The rest of the elements mentioned above are

¹ Internationalization is the process of designing applications so that they can be adapted to various languages and regions without engineering changes. Localization is the process of adapting software for a specific region or language by adding locale-specific components and translating text.

* Corresponding author. Tel.: +34 91 488 81 06; fax: +34 91 664 74 94.
E-mail address: grex@gsyc.escet.urjc.es (G. Robles).
URL: <http://gsyc.escet.urjc.es/~grex> (G. Robles).

- Massey, B., 2005. Longitudinal analysis of long-timescale Open Source repository data. In: Proceedings of the International Workshop on Predictor Models in Software Engineering (PROMISE 2005), St. Louis, MI, USA.
- Mockus, A., Votta L.G., 2000. Identifying reasons for software changes using historic databases. In: Proceedings of the International Conference on Software Maintenance, San Jose, CA, USA, pp. 120–130.
- Mockus, A., Fielding, R.T., Herbsleb, J.D., 2002. Two case studies of Open Source software development: Apache and Mozilla. *ACM Transactions on Software Engineering and Methodology* 11 (3), 309–346.
- Parnas, D.L., 1994. Software aging. In: Proceedings of the International Conference on Software Engineering, Sorrento, Italy, pp. 279–287.
- Purushothaman, R., Perry, D.E., 2005. Toward understanding the rhetoric of small source code changes. *IEEE Transactions on Software Engineering* 31 (6), 511–526.
- Robles, G., Koch, S., Gonzalez-Barahona, J.M., 2004. 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, pp. 51–55.
- Robles, G., Amor, J.J., Gonzalez-Barahona, J.M., Herraiz, I., 2005. Evolution and growth in large libre software projects. In: Proceedings of the International Workshop on Principles in Software Evolution, Lisbon, Portugal, pp. 165–174.
- Robles, G., Gonzalez-Barahona, J.M., Herraiz, I., 2005. An empirical approach to software archaeology. In: Poster Proceedings of the International Conference on Software Maintenance, Budapest, Hungary, pp. 47–50.
- Turski, W.M., 1996. Reference model for smooth growth of software systems. *IEEE Transactions on Software Engineering* 22 (8), 599–600.
- Zimmermann, T., Weigerber, P., 2004. Processing CVS data for fine-grained analysis. In: Proceedings of the International Workshop on Mining Software Repositories, Edinburg, Scotland, UK, pp. 2–6.
- Zimmermann, T., Weigerber, P., Diehl, S., Zeller, A., 2005. Mining version histories to guide software changes. *IEEE Transactions on Software Engineering* 31 (6), 429–445.

Web Images More... jgbarah@gmail.co



Scholar



Jesus M. Gonzalez-Barahona

Beyond source code: the importance of other artifacts in software development (a case study) PDF

Authors: Gregorio Robles, Jesus M Gonzalez-Barahona, Juan Julian Merelo

Publication date: 2006/9/30

Journal: Journal of Systems and Software

Volume: 79

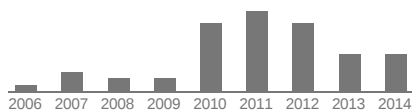
Issue: 9

Pages: 1233-1248

Publisher: Elsevier

Description: Current software systems contain increasingly more elements that have not usually been considered in software engineering research and studies. Source artifacts, understood as the source components needed to obtain a binary, ready to use version of a program, comprise in many systems more than just the elements written in a programming language (source code). Especially when we move apart from systems-programming and enter the realm of end-user applications, we find files for documentation, interface specifications, ...

Total citations: [Cited by 56](#)



Scholar articles: [Beyond source code: the importance of other artifacts in software development \(a case study\)](#)
 G Robles, JM Gonzalez-Barahona, JJ Merelo - Journal of Systems and Software, 2006
[Cited by 56](#) - [Related articles](#) - [All 8 versions](#)

Dates and citation counts are estimated and are determined automatically by a computer program.

[My Citations](#)
 [Help](#)
 [About Google Scholar](#)
 [Privacy & Terms](#)



Contents lists available at ScienceDirect

Information Economics and Policy

journal homepage: www.elsevier.com/locate/ie

Geographic origin of libre software developers [☆]

Jesus M. Gonzalez-Barahona ^{a,*}, Gregorio Robles ^a, Roberto Andradas-Izquierdo ^a,
Rishab Aiyer Ghosh ^b

^a GSyC/LibreSoft, Departamento de Sistemas Telemáticos y Computación, Universidad Rey Juan Carlos, C/Tulipan s/n, 28903 Mostoles, Spain

^b Collaborative Creativity Group, United Nations University (UNU-MERIT), Keizer Karelplein 19, 6211TC Maastricht, The Netherlands

ARTICLE INFO

Article history:

Available online 29 July 2008

JEL classification:

C81

L17

Keywords:

Geographical location

Data mining

Libre software

Free software

Open source software

ABSTRACT

This paper examines the claim that libre (free, open source) software involves global development. The anecdotal evidence is that developers usually work in teams including individuals residing in many different geographical areas, time zones and even continents and that, as a whole, the libre software community is also diverse in terms of national origin. However, its exact composition is difficult to capture, since there are few records of the geographical location of developers. Past studies have been based on surveying a limited (and sometimes biased) sample and extrapolating that sample to the global distribution of developers. In this paper we present an alternate approach in which databases are analyzed to create traces of information from which the geographical origin of developers can be inferred. Applying this technique to the SourceForge users database and the mailing lists archives from several large projects, we have estimated the geographical origin of more than one million individuals who are closely related to the libre software development process. The paper concludes that the result is a good proxy for the actual distribution of libre software developers working on global projects.

© 2008 Elsevier B.V. All rights reserved.

1. Introduction

One of the most well known claims about libre (free, open source) software ¹ is that it is based on an internationally distributed pool of developers. Most of these projects are open to participation by any interested individual with Internet access who has sufficient knowledge and skills to make a contribution. There are apparently few barriers to participation that are due to the geographical location of a developer. However, the distribution of those developers in the different regions of the globe is extremely uneven, showing that barriers do exist, even when they are not built by the projects themselves. Obtaining a clear picture of the actual geographical distribution of libre software developers is a precondition to analyzing the nature and source of barriers and their effect on the participation of specific populations in this global phenomenon. The

[☆] This work has been funded in part by the European Commission, under the FLOSSMETRICS and FLOSSWorld projects (IST program, Contract Numbers 015722 and 033982). This work is based in part on the SourceForge database provided by University of Notre Dame, see details at <http://www.nd.edu/oss/Data/data.html>. This work is based in part in the contribution “Geographic Location of Developers at SourceForge”, by Gregorio Robles and Jesus M. Gonzalez-Barahona, presented in the Mining Software Repositories Workshop, Shanghai, May 2006.

* Corresponding author.

E-mail addresses: jgb@gsyc.es (J.M. Gonzalez-Barahona), grex@gsyc.es (G. Robles), randradas@gsyc.es (R. Andradas-Izquierdo), ghosh@merit.unu.edu (R.A. Ghosh).

¹ Through this paper we will use the term libre software to refer both to free software (according to the Free Software Foundation) and to open source software (according to the Open Source Initiative).

5. Conclusions and further research

In this paper we have shown a more complete and detailed landscape of the geographical distribution of libre software developers around the world than previous studies. The combined analysis of SourceForge users data and mailing list archives also offers a more complete perspective on participation.

It is important to emphasize that this research is focused on global development (that is, projects not targeted in particular to a specific geographical or cultural community), and for that reason, developers participating only in regional or local projects are not considered. These developer groups can form sizable communities in some areas of the world.⁷ In addition, and despite the large size and representativeness of our samples, some bias might remain. For instance, projects that are not present in SourceForge may have a different geographical distribution. However, since SourceForge hosted projects and the other projects we studied account for a sizable fraction of all libre software available, we believe this bias is inconsequential.

All of the results presented are the result of heuristics and (educated) assumptions, and are therefore inexact. We have worked with sources having rather different error margins, and we have used heuristics that are sound, but they are subject to a certain error rate in identifying locations. To assess the validity of the methodology for estimating the national origin, it would be desirable to check (probably by contacting developers themselves) a large fraction of SourceForge users. The results could then be compared with those of our study. However, the validations we have performed seem to indicate that the results are statistically sound, and that the figures shown are good estimators of the reality.

Our methodology is not focused on identifying the geographical location of single developers (although in many cases that is done), but on finding the aggregate numbers of developers of a certain national origin. In many cases, therefore, we use algorithmically driven estimations to infer the proportion of nationals of a certain country in a population of users having certain characteristics. This is certainly a limitation of the proposed approach, especially if we were interested in (individual) developer identification methods as proposed in other works (e.g., Robles and Gonzalez-Barahona, 2005).

Acknowledgements

We thank the SourceForge team, and Greg Madey from the University of Notre Dame, for providing access to the SourceForge data. Thanks to Martin Michlmayr for his help with the Debian mail archives. Also, a big thank you goes to our colleagues from GSyC/LibreSoft for their help in verifying the validity of the data. Last, but not least, thanks to the anonymous reviewers; their comments have helped to improve this paper.

References

- David, P.A., Waterman, A., Arora, S., 2003. FLOSS-US. The free/libre/open source software survey for 2003. Technical Report, Stanford Institute for Economic and Policy Research, Stanford, CA, USA.
- Dempsey, B.J., Weiss, D., Jones, P., Greenberg, J., 2002. Who is an open source software developer? *Communications of the ACM* 45 (2), 67–72.
- Ghosh, R.A., Glott, R., Krieger, B., Robles, G., 2002. Survey of developers (free/libre and open source software: survey and study). Technical Report, International Institute of Infonomics, University of Maastricht, The Netherlands.
- Hertel, G., Niedner, S., Herrmann, S., 2003. Motivation of software developers in open source projects: an internet-based survey of contributors to the linux kernel. *Research Policy* (32), 1159–1177.
- Lancashire, D., 2001. Code, culture and cash: the fading altruism of open source development. *First Monday* 6 (12).
- Robles, G., Gonzalez-Barahona, J.M., 2005. Developer identification methods for integrated data from various sources. In: *Proceedings of the International Workshop on Mining Software Repositories*, St. Louis, Missouri, USA, pp. 106–110.
- Robles, G., Gonzalez-Barahona, J.M., 2006. Geographic location of developers at SourceForge. In: *Proceedings of the Mining Software Repositories Workshop*, Shanghai, China.
- Robles, G., Scheider, H., Tretkowski, I., Weber, N., 2001. Who is doing it? A research on libre software developers. Technical Report, Technische Universitt Berlin, Berlin, Germany.
- Tuomi, I., 2004. Evolution of the Linux credits file: methodological challenges and reference data for open source research. *First Monday* 9 (6).

⁷ A separate research in which the authors are also working shows that important regional communities, with little relationship with the global community, exist in regions such as East Asia and Brazil.

Web Images More... jgbarah@gmail.com

Google

Scholar ← Edit 🗑 Export ▾



Jesus M. Gonzalez-Barahona

Geographic origin of libre software developers

Authors: Jesus M Gonzalez-Barahona, Gregorio Robles, Roberto Andradas-Izquierdo, Rishab Aiyer Ghosh

Publication date: 2008/12/31

Journal: Information Economics and Policy

Volume: 20

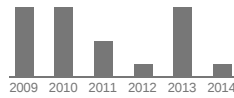
Issue: 4

Pages: 356-363

Publisher: North-Holland

Description: This paper examines the claim that libre (free, open source) software involves global development. The anecdotal evidence is that developers usually work in teams including individuals residing in many different geographical areas, time zones and even continents and that, as a whole, the libre software community is also diverse in terms of national origin. However, its exact composition is difficult to capture, since there are few records of the geographical location of developers. Past studies have been based on surveying a limited ...

Total citations: [Cited by 23](#)



Scholar articles: [Geographic origin of libre software developers](#)
 JM Gonzalez-Barahona, G Robles... - Information Economics and Policy, 2008
[Cited by 23](#) - [Related articles](#) - [All 5 versions](#)

Dates and citation counts are estimated and are determined automatically by a computer program.

[Help](#) [Privacy](#) [Terms](#) [Provide feedback](#) [My Citations](#)

Macro-level software evolution: a case study of a large software compilation

Jesus M. Gonzalez-Barahona · Gregorio Robles ·
Martin Michlmayr · Juan José Amor ·
Daniel M. German

Published online: 29 November 2008

© The Author(s) 2008. This article is published with open access at Springerlink.com

Editors: Ahmed Hassan, Stephan Diehl and Harald Gall

Abstract Software evolution studies have traditionally focused on individual products. In this study we scale up the idea of software evolution by considering software compilations composed of a large quantity of independently developed products, engineered to work together. With the success of libre (free, open source) software, these compilations have become common in the form of ‘software distributions’, which group hundreds or thousands of software applications and libraries into an integrated system. We have performed an exploratory case study on one of them, Debian GNU/Linux, finding some significant results. First, Debian has been doubling in size every 2 years, totalling about 300 million lines of code as of 2007. Second, the mean size of packages has remained stable over time. Third, the number of dependencies between packages has been growing quickly. Finally, while C is still by far the most commonly used programming language for applications, use of the C++, Java, and Python languages have all significantly increased. The study helps not

J. M. Gonzalez-Barahona (✉) · G. Robles · J. J. Amor
Universidad Rey Juan Carlos, Madrid, Spain
e-mail: jgb@gsync.es

G. Robles
e-mail: grex@gsync.es

J. J. Amor
e-mail: jjamor@gsync.es

M. Michlmayr
Open Source Program Office, HP, Innsbruck, Austria
e-mail: martin@michlmayr.org

D. M. German
University of Victoria, Victoria, Canada
e-mail: dmgerman@uvic.ca




Juan José Amor has a M.Sc. in Computer Science from the Universidad Politécnica de Madrid and he is currently pursuing a Ph.D. at the Universidad Rey Juan Carlos, where he is also a project manager. His research interests are related to libre software engineering, mainly effort and schedule estimates in libre software projects. Since 1995 he has collaborated in several libre software organizations; he is also co-founder of LuCAS, the best known libre software documentation portal in Spanish, and Hispalinux, the biggest spanish Linux user group. He also collaborates with Barrapunto.com and Linux+.



Daniel M. German is associate professor of computer science at the University of Victoria, Canada. His main areas of interest are software evolution, open source software engineering and intellectual property.

Web Images More... jgbarah@gmail.com



Scholar ← Edit 🗑️ Export ▾



Jesus M. Gonzalez-Barahona

Macro-level software evolution: a case study of a large software compilation [\[HTML\]](#) from [springer.com](#)

Authors: Jesus M Gonzalez-Barahona, Gregorio Robles, Martin Michlmayr, Juan José Amor, Daniel M German
 Publication date: 2009/6/1
 Journal: Empirical Software Engineering
 Volume: 14
 Issue: 3
 Pages: 262-285
 Publisher: Springer US

Description: Abstract Software evolution studies have traditionally focused on individual products. In this study we scale up the idea of software evolution by considering software compilations composed of a large quantity of independently developed products, engineered to work together. With the success of libre (free, open source) software, these compilations have become common in the form of 'software distributions', which group hundreds or thousands of software applications and libraries into an integrated system. We have performed an ...

Total citations [Cited by 62](#)



Scholar articles [Macro-level software evolution: a case study of a large software compilation](#)
 JM Gonzalez-Barahona, G Robles, M Michlmayr... - Empirical Software Engineering, 2009
[Cited by 62](#) - [Related articles](#) - [All 18 versions](#)

Dates and citation counts are estimated and are determined automatically by a computer program.

[Help](#) [Privacy](#) [Terms](#) [Provide feedback](#) [My Citations](#)

On the reproducibility of empirical software engineering studies based on data retrieved from development repositories

Jesús M. González-Barahona · Gregorio Robles

Published online: 18 October 2011

© The Author(s) 2011. This article is published with open access at Springerlink.com

Editors: Martin Shepperd and Tim Menzies

Abstract Among empirical software engineering studies, those based on data retrieved from development repositories (such as those of source code management, issue tracking or communication systems) are specially suitable for reproduction. However their reproducibility status can vary a lot, from easy to almost impossible to reproduce. This paper explores which elements can be considered to characterize the reproducibility of a study in this area, and how they can be analyzed to better understand the type of reproduction studies they enable or obstruct. One of the main results of this exploration is the need of a systematic approach to assess the reproducibility of a study, due to the complexity of the processes usually involved, and the many details to be taken into account. To address this need, a methodology for assessing the reproducibility of studies is also presented and discussed, as a tool to help to raise awareness about research reproducibility in this field. The application of the methodology in practice has shown how, even for papers aimed to be reproducible, a systematic analysis raises important aspects that render reproduction difficult or impossible. We also show how, by identifying elements and attributes related to reproducibility, it can be better understood which kind of reproduction can be done for a specific study, given the description of datasets, methodologies and parameters it uses.

Keywords Repeatable results · Mining software repositories · Reproducibility


J. M. González-Barahona
Universidad Rey Juan Carlos, Mostoles, Spain

G. Robles (✉)
Universidad Rey Juan Carlos, Fuenlabrada, Spain
e-mail: grex@gsyc.urjc.es



Gregorio Robles is Associate Professor at the Universidad Rey Juan Carlos, which has several campi distributed in the region of Madrid (Spain). He earned his PhD in 2006 and currently has mainly teaching duties in the field of computer networks, although he teaches in a master on libre software as well introductory computer courses to journalism students. Regarding research, he works in two fields: a) technology enhanced learning, which means that he tries to use technology to improve the learning processes, and b) software engineering research on Free/Libre/Open Source Software systems with special focus on empirical issues.

Web Images More... jgbarah@gmail.com



Scholar ← Edit 🗑️ Export ▾



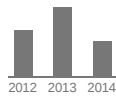
Jesus M. Gonzalez-Barahona

On the reproducibility of empirical software engineering studies based on data retrieved from development repositories [\[HTML\] from springer.com](#)

Authors: Jesús M González-Barahona, Gregorio Robles
 Publication date: 2012/2/1
 Journal: Empirical Software Engineering
 Volume: 17
 Issue: 1-2
 Pages: 75-89
 Publisher: Springer US

Description: Abstract Among empirical software engineering studies, those based on data retrieved from development repositories (such as those of source code management, issue tracking or communication systems) are specially suitable for reproduction. However their reproducibility status can vary a lot, from easy to almost impossible to reproduce. This paper explores which elements can be considered to characterize the reproducibility of a study in this area, and how they can be analyzed to better understand the type of reproduction ...

Total citations **Cited by 27**



Scholar articles [On the reproducibility of empirical software engineering studies based on data retrieved from development repositories](#)
 JM González-Barahona, G Robles - Empirical Software Engineering, 2012
[Cited by 27](#) - [Related articles](#) - [All 7 versions](#)

Dates and citation counts are estimated and are determined automatically by a computer program.

[Help](#) [Privacy](#) [Terms](#) [Provide feedback](#) [My Citations](#)

The Evolution of the Laws of Software Evolution: A Discussion Based on a Systematic Literature Review

ISRAEL HERRAIZ, Technical University of Madrid, Spain
DANIEL RODRIGUEZ, University of Alcalá, Madrid, Spain
GREGORIO ROBLES and JESUS M. GONZALEZ-BARAHONA, GSyC/Libresoft,
University Rey Juan Carlos, Madrid, Spain

After more than 40 years of life, software evolution should be considered as a mature field. However, despite such a long history, many research questions still remain open, and controversial studies about the validity of the laws of software evolution are common. During the first part of these 40 years, the laws themselves evolved to adapt to changes in both the research and the software industry environments. This process of adaption to new paradigms, standards, and practices stopped about 15 years ago, when the laws were revised for the last time. However, most controversial studies have been raised during this latter period. Based on a systematic and comprehensive literature review, in this article, we describe how and when the laws, and the software evolution field, evolved. We also address the current state of affairs about the validity of the laws, how they are perceived by the research community, and the developments and challenges that are likely to occur in the coming years.

Categories and Subject Descriptors: D.2.7 [Software Engineering]: Distribution, Maintenance and Enhancement

General Terms: Management

Additional Key Words and Phrases: Laws of software evolution, software evolution

ACM Reference Format:

Herraiz, I., Rodriguez, D., Robles, G., and Gonzalez-Barahona, J. M. 2013. The evolution of the laws of software evolution: A discussion based on a systematic literature review. *ACM Comput. Surv.* 46, 2, Article 28 (November 2013), 28 pages.

DOI: <http://dx.doi.org/10.1145/2543581.2543595>

1. INTRODUCTION

In 1969, Meir M. Lehman did an empirical study (originally confidential, later published [Lehman 1985b]) within IBM, with the idea of improving the company's programming effectiveness. The study received little attention in the company and had no impact on its development practices. This study, however, started a new and prolific field of research: *software evolution*.

Software evolution deals with the process by which programs are modified and adapted to their changing environment. The aim of Lehman's research was to formulate a scientific theory of software evolution. As any sound theory, it was meant to be based on empirical results and aimed at finding invariant properties to be observed on entire classes of software development projects. As a result of his research, some invariants were found, which were first described in Lehman [1974] as the *laws of software evolution*.

Author's address: israel.herraiz@upm.es.

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies show this notice on the first page or initial screen of a display along with the full citation. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, to republish, to post on servers, to redistribute to lists, or to use any component of this work in other works requires prior specific permission and/or a fee. Permissions may be requested from Publications Dept., ACM, Inc., 2 Penn Plaza, Suite 701, New York, NY 10121-0701 USA, fax +1 (212) 869-0481, or permissions@acm.org.

© 2013 ACM 0360-0300/2013/11-ART28 \$15.00

DOI: <http://dx.doi.org/10.1145/2543581.2543595>

- SHULL, F., CARVER, J., VEGAS, S., AND JURISTO, N. 2008. The role of replications in empirical software engineering. *Empirical Software Eng.* 13, 2, 211–218.
- SIEBEL, N. T., COOK, S., SATPATHY, M., AND RODRIGUEZ, D. 2003. Latitudinal and longitudinal process diversity. *J. Software Maintenance Evol. Res. Pract.* 15, 1, 9–25.
- SJØBERG, D. I. K., DYBÅ, T., ANDA, B. C. D., AND HANNAY, J. E. 2008. Building theories in software engineering. In *Guide to Advanced Empirical Software Engineering*, F. Shull, J. Singer, and D. I. K. Sjøberg, Eds., Springer, London, 312–336.
- SMITH, N., CAPILUPPI, A., AND RAMIL, J. F. 2005. A study of open source software evolution data using qualitative simulation. *Software Process: Improve. Pract.* 10, 3, 287–300.
- TEMPERO, E., ANSLOW, C., DIETRICH, J., HAN, T., LI, J., LUMPE, M., MELTON, H., AND NOBLE, J. 2010. Qualitas corpus: A curated collection of Java code for empirical studies. In *Proceedings of the Asia Pacific Software Engineering Conference*.
- TURSKI, W. M. 1996. Reference model for smooth growth of software systems. *IEEE Trans. Software Eng.* 22, 8, 599–600.
- TURSKI, W. M. 2002. The reference model for smooth growth of software systems revisited. *IEEE Trans. Software Eng.* 28, 8, 814–815.
- VASA, R. 2010. *Growth and Change Dynamics in Open Source Software Systems*. Ph.D. thesis, Swinburne University of Technology, Melbourne, Australia.
- WERNICK, P. AND LEHMAN, M. M. 1999. Software process white box modelling for FEAST/1. *J. Syst. Software* 46, 2–3, 193–201.
- WOODSIDE, C. M. 1980. A mathematical model for the evolution of software. *J. Syst. Software* 1, 4, 337–345.
- WOODSIDE, C. M. 1985. A mathematical model for the evolution of software. In *Program Evolution. Processes of Software Change*, M. M. Lehman and L. A. Belady, Eds., Academic Press, San Diego, CA, 339–354.

Received April 2012; revised October 2012; accepted June 2013

Understanding How Companies Interact with Free Software Communities

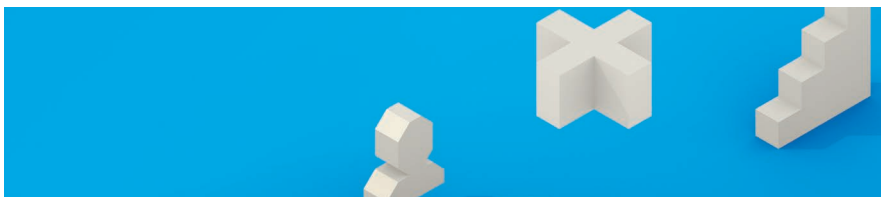
Jesus M. Gonzalez-Barahona, Universidad Rey Juan Carlos

Daniel Izquierdo-Cortazar, Bitergia

Stefano Maffulli, OpenStack

Gregorio Robles, Universidad Rey Juan Carlos

// When free, open source software development communities work with companies that use their output, it's especially important for both parties to understand how this collaboration is performing. The use of data analytics techniques on software development repositories can improve factual knowledge about performance metrics. //



FREE, LIBRE, OPEN source software (FLOSS) communities can be very complex and difficult to understand, so quantitative, neutral information about them becomes valuable,

especially when they're large and involve competing companies.¹ Fortunately, many FLOSS projects operate in open, public development repositories, which has facilitated the

progress of new analytics techniques to study them. In turn, these techniques have started to produce useful results for both practitioners and other stakeholders.²

In particular, studies of company participation in large projects have started to raise industrial interest: FLOSS foundations want to learn about company participation in their projects, and the companies want to know more about corporate activity in the projects on which they rely. Here, we present two studies in this area—by LibreSoft, a research group specializing in the quantitative analysis of software development, and Bitergia, a LibreSoft spin-off company focused on software analytics services—into company activity in OpenStack and the fairness of WebKit's review process.

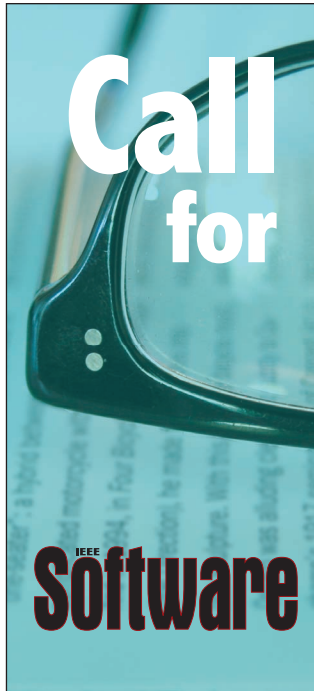
Characterizing Company Participation

Although corporate contributions have always played a role in FLOSS projects, the emergence of “communities of companies” is driving an interest in analyzing company behavior. Although this new kind of community admits individual contributions, it clearly prioritizes corporate interests, and participating companies, which can be commercial competitors, employ most of the developers.³ Researchers can study several aspects of this participation:

- activity, or how companies contribute to the project in code changes, bug fixes, or participation in discussions;
- neutrality, or how neutral the project stays with respect to accepting contributions from or fixing bugs reported by companies; and
- collaboration, or how companies work in the same areas, collaborate to fix bugs, or take joint decisions.

linuxfoundation.org/sites/main/files/publications/whowriteslinux.pdf.

5. H. Kagdi, M.L. Collard, and J.I. Maletic, "A Survey and Taxonomy of Approaches for Mining Software Repositories in the Context of Software Evolution," *J. Software Maintenance and Evolution: Research and Practice*, vol. 19, no. 2, 2007, pp. 77–131.
6. E. Kouters et al., "Who's Who in Gnome: Using Isa to Merge Software Repository Identities," *Proc. 28th IEEE Int'l Conf. Software Maintenance (ICSM 2012 ERA)*, IEEE, 2012, pp. 592–595.
7. C. Bird and N. Nagappan, "Who? Where? What? Examining Distributed Development in Two Large Open Source Projects," *Proc. 9th IEEE Working Conf. Mining Software Repositories (MSR 12)*, IEEE, 2012, pp. 237–246.
8. P. Rigby et al., "Contemporary Peer Review in Action: Lessons from Open Source Development," *IEEE Software*, vol. 29, no. 6, 2012, pp. 56–61.
9. J.M. Gonzalez-Barahona and G. Robles, "On the Reproducibility of Empirical Software Engineering Studies Based on Data Retrieved from Development Repositories," *Empirical Software Eng.*, vol. 17, nos. 1–2, 2012, pp. 75–89.



Articles

IEEE Software seeks practical, readable articles that will appeal to experts and nonexperts alike. The magazine aims to deliver reliable information to software developers and managers to help them stay on top of rapid technology change. Submissions must be original and no more than 4,700 words, including 200 words for each table and figure.

Author guidelines:
www.computer.org/software/author.htm
Further details: software@computer.org
www.computer.org/software

IEEE computer society NEWSLETTERS Stay Informed on Hot Topics

COMPUTING NOW
TRAINING SPOTLIGHT
TRANSACTIONS CONNECTION
WHAT'S NEW BUILD YOUR CAREER COMPUTING ORGANIZATION DIGITAL LIBRARY NEWS FLASH
IN COMPUTER CAREER COMPUTING ORGANIZATION
MEMBER CONNECTION NEWS FLASH
DIGITAL LIBRARY NEWS FLASH
CONFERENCE CONNECTION
CONNECTION
TRANSACTIONS CONNECTION
NEW IN COMPUTER
BUILD YOUR CAREER
MEMBER CONNECTION
COMPUTING NOW
TRAINING SPOTLIGHT
MEMBER CONNECTION



computer.org/newsletters

Studying the laws of software evolution in a long-lived FLOSS project

Jesus M. Gonzalez-Barahona^{1,*†}, Gregorio Robles¹, Israel Herraiz² and Felipe Ortega¹

¹*GSyC/LibreSoft, Universidad Rey Juan Carlos, Fuenlabrada, Spain*

²*Universidad Politécnica de Madrid, Madrid, Spain*

SUMMARY

Some free, open-source software projects have been around for quite a long time, the longest living ones dating from the early 1980s. For some of them, detailed information about their evolution is available in source code management systems tracking all their code changes for periods of more than 15 years. This paper examines in detail the evolution of one of such projects, *glibc*, with the main aim of understanding how it evolved and how it matched Lehman's laws of software evolution. As a result, we have developed a methodology for studying the evolution of such long-lived projects based on the information in their source code management repository, described in detail several aspects of the history of *glibc*, including some activity and size metrics, and found how some of the laws of software evolution may not hold in this case. © 2013 The Authors. *Journal of Software: Evolution and Process* published by John Wiley & Sons Ltd.

Received 17 March 2012; Revised 3 July 2013; Accepted 12 July 2013

KEY WORDS: free software; open source software; software evolution; source code management system; mining software repositories

1. INTRODUCTION

Large, long-lived software projects are difficult to study. Until some years ago, having access to one of them was very difficult and time-consuming. In addition, reliable information about them has to be available for all the period under study. If available, it still has to be found, extracted, and analyzed to produce meaningful time series and other data artifacts that can be studied to gain knowledge about its evolution.

This situation changed during the last years with the public availability of mature free, open-source software (FLOSS) projects, some of which now feature more than 15 or even 20 years of history. Many have been supported by source code management (SCM) systems over a large fraction of their life, and the corresponding repositories have been maintained carefully enough to still have much of the historical information available, with enough reliability and detail to allow for the derivation of significant conclusions.

In this paper, we study one of those long-lived FLOSS projects, *glibc*, with over 20 years of history in its SCM repository. Evolution in this kind of large, long-lived projects has many edges: all of them should be considered to understand the whole story. Our purpose is to analyze this wealth of information using a repeatable methodology that allows for the analysis of several aspects of the evolution of the project. In the process, we have also identified several problems and shortcomings of the available data that have to be taken into account for such analysis. More in particular, the main goal of the study can be summarized as follows:

*Correspondence to: Jesus M. Gonzalez-Barahona, GSyC/LibreSoft, Universidad Rey Juan Carlos, Fuenlabrada, Spain.

†E-mail: jgb@gsyc.es

This is an open access article under the terms of the Creative Commons Attribution License, which permits use, distribution and reproduction in any medium, provided the original work is properly cited.

31. Madhavji NH, Fernandez-Ramil J, Perry DE (eds.). *Software Evolution and Feedback. Theory and Practice*. Wiley, Chichester, UK 2006.
32. Lehman MM, Perry DE, Ramil JF. Implications of evolution metrics on software maintenance. *Proceedings of International Conference on Software Maintenance*. IEEE Computer Society, 1998; 208–217.
33. Herraiz I, Gonzalez-Barahona JM, Robles G. Towards a theoretical model for software growth. *International Workshop on Mining Software Repositories*, IEEE Computer Society: Minneapolis, MN, USA, 2007; 21–30.
34. Gonzalez-Barahona JM, Robles G. On the reproducibility of empirical software engineering studies based on data retrieved from development repositories. *Empirical Software Engineering* 2012; **17**(1-2):75–89.

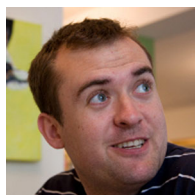
AUTHORS' BIOGRAPHIES



Jesus M. Gonzalez-Barahona teaches and researches at the Universidad Rey Juan Carlos and collaborates with Bitergia, a software development analytics company. He is interested in understanding free/open source software development, in finding ways to improve its efficiency, and in sharing this knowledge. He is a member of the Computer Society of the Institute of Electrical and Electronics Engineers and of the Asociacion de Técnicos en Informática.



Gregorio Robles is an associate professor at the Universidad Rey Juan Carlos. His research interests involve software engineering in free/libre open source software, mining software repositories, and technology enhanced learning. He is a member of the Institute of Electrical and Electronics Engineers.



Israel Herraiz is an assistant professor at the Technical University of Madrid. He holds a PhD degree with European Mention on Computer Science from the Universidad Rey Juan Carlos. He has been a visiting scholar to the University of California at Davis, the Queen's University and University of Victoria (Canada), and the Open University and Oxford University (UK).



Felipe Ortega is a researcher in the Department of Statistics and Operations Research at the University Rey Juan Carlos. He got a PhD degree in Computer Science from the same institution. He studies open online communities and open collaboration from a quantitative perspective, applying data science methods and technologies. He has been visiting scholar at Télécom Bretagne (France) and has been an invited speaker at the Georgia Institute of Technology, Xerox PARC and Instituto Cervantes.

1.A.2.

**PUBLICACIONES CIENTÍFICAS NO
INDEXADAS DE ACUERDO CON UN
ÍNDICE DE CALIDAD RELATIVO**

UPGRADE is the European Online Magazine for the Information Technology Professional, published bimonthly at <http://www.upgrade-cepis.org/>

Publisher

UPGRADE is published on behalf of CEPIS (Council of European Professional Informatics Societies, <http://www.cepis.org/>) by Novática (<http://www.ati.es/novatica/>) and Informatik/Informatique (<http://www.svifsi.ch/revue/>)

Chief Editors

François Louis Nicolet, Zurich <nicolet@acm.org>
 Rafael Fernández Calvo, Madrid <rfoalvo@ati.es>

Editorial Board

Peter Morrogh, CEPIS President
 Prof. Wolfried Stucky, CEPIS President-Elect
 Fernando Sanjuán de la Rocha and Rafael Fernández Calvo, ATI
 Prof. Carl August Zehnder and François Louis Nicolet, SVI/FSI

English Editors: Mike Andersson, Richard Butchart, David Cash, Arthur Cook, Tracey Darch, Laura Davies, Nick Dunn, Rodney Fennemore, Hilary Green Roger Harris, Michael Hird, Jim Holder, Alasdair MacLeod, Pat Moody, Adam David Moss, Phil Parkin, Brian Robson

Cover page designed by Antonio Crespo Foix, © ATI 2001

Layout: Pascale Schürmann

E-mail addresses for editorial correspondence: <nicolet@acm.org> and <rfoalvo@ati.es>

E-mail address for advertising correspondence: <novatica@ati.es>

Copyright

© Novática and Informatik/Informatique. All rights reserved. Abstracting is permitted with credit to the source. For copying, reprint, or republication permission, write to the editors.

The opinions expressed by the authors are their exclusive responsibility.

Open Source / Free Software: Towards Maturity

Guest Editors: Joe Ammann, Jesús M. González-Barahona, Pedro de las Heras Quirós

Joint issue with NOVÁTICA and INFORMATIK/INFORMATIQUE

- 2 Presentation – *Joe Ammann, Jesús M. González-Barahona, Pedro de las Heras Quirós, Guest Editors*
- 4 Free Software Today
 – *Pedro de las Heras Quirós and Jesús M. González-Barahona*
The position of many major companies with regard to Free Software is changing. New companies are becoming giants. It is vital for the data on which we base this idea to be right up to date. Any impression based on data from a few months ago will very possibly be wrong.
- 12 Should Business Adopt Free Software?
 – *Gilbert Robert and Frédéric Schütz*
We explain what Free Software is, and what its advantages are for users, and provide an overview of its status in business, in particular by looking at the obstacles which still stand in the way of its use.
- 20 Harm from The Hague – *Richard Stallman*
The proposed Hague Treaty threatens to subject software developers in Europe to U.S. software patents. The consequence is that you could be sued about information you distributed under the laws of any country, and the judgement would be enforced by your country.
- 23 Software Patentability with Compensatory Regulation: a Cost Evaluation – *Jean Paul Smets and Hartmut Pilch*
The European Patent Office has proposed to remove limitations on patentability, such as the exclusion of computer programs. The French Academy of Technologies suggests additional regulation measures in order to reduce potential abuses of software patents.
- 33 Open Source in a Major Swiss Bank
 – *Klaus Bucka-Lassen and Jan Sorensen*
This article highlights which advantages and disadvantages of Open Source Software are of significance for a financial services provider. It describes the problems that arose, and what convinced management to use Struts for Web application developments.
- 36 European Initiatives Concerning the Use of Free Software in the Public Sector
 – *Juan Jesús Muñoz Esteban*
The European Commission is beginning to make use of Free Software for some of their strategic initiatives. A study of the use of Free Software in several administrations of different countries analyses the reasons for adopting it.
- 41 GNU Enterprise Application Software – *Neil Tiffin and Reinhard Müller*
GNUe is a set of integrated business applications and tools to support accounting, supply chain, human resources, sales, manufacturing, and other business processes. We describe the project, the idea and motivation for developers and users behind it.
- 45 The Debian GNU/Linux Project – *Javier Fernández-Sanguino Peña*
The Debian GNU/Linux project is one of the most ambitious Free Software projects, involving a large number of developers creating a totally free operating system.
- 50 Journal File Systems in Linux – *Ricardo Galli*
Linux buffer/cache is really impressive and affected, positively, all the figures of my compilations, copies and random reads and writes.
- 57 The Crisis of Free Scientific Software – *David Santo Orcero*
The scientific world was among the pioneers in creating Free Software. In the 1990s Free Software started to spread into other areas. In certain fields this reached a point where there are either no free tools available, or no more free tools are being actively developed.
- 60 Counting Potatoes: the Size of Debian 2.2
 – *Jesús M. González-Barahona, Miguel A. Ortuño Pérez, Pedro de las Heras Quirós, José Centeno González and Vicente Matellán Olivera*
Debian is the largest Free Software distribution, with more than 4,000 source packages in the release currently in preparation. We show that the Debian development model is at least as capable as other development methods to manage distributions of this size.

Coming issue:
“Knowledge Management”

Counting Potatoes: the Size of Debian 2.2

Jesús M. González-Barahona, Miguel A. Ortuño Pérez, Pedro de las Heras Quirós, José Centeno González and Vicente Matellán Olivera

Debian is the largest Free Software distribution, with well over 2,800 source packages in the latest stable release (Debian 2.2) and more than 4,000 source packages in the release currently in preparation. But, how large is “the largest”? In this paper, we use David Wheeler’s sloccount system to determine the number of physical source lines of code (SLOC) of Debian 2.2 (aka Potato). We show that Debian 2.2 includes over 56,000,000 physical SLOC (almost twice than Red Hat 7.1, released about 8 months later), showing that the Debian development model (based on the work of a large group of voluntary developers spread around the world) is at least as capable as other development methods (like the more centralized one, based on the work of employees, used by Red Hat or Microsoft) to manage distributions of this size.

Keywords: Debian, Free Software, Libre Software, SLOC, Lines of Code, Linux

1 Introduction

On August 14th of 2000 the Debian Project announced Debian GNU/Linux 2.2, the “Joel ‘Espy’ Klecker” release [Debian22Ann] [Debian22Rel]. Code named “Potato”, it is the latest (to date) release of the Debian GNU/Linux Operating System. In this paper, we have counted this distribution, showing its size, and comparing it to other distributions.

Debian is not only the largest GNU/Linux distribution, it is also one of the more reliable, and enjoys several awards based on users preferences. Although its user base is difficult to estimate, since the Debian Project does not sell CDs or other media with the software, it is for sure important within the Linux market. It also takes special care to benefit from one of the freedoms that Free Software provides to users: the availability of source code. Therefore, source packages are carefully crafted for easy compilation and reconstruction of original (upstream) sources. This makes it also convenient to measure and in general, to get statistics about it.

The idea of this paper came after reading David Wheeler’s fine paper [Wheeler 01]. We encourage the reader to at least browse it, and compare the data it offers for Red Hat Linux with the ones found here.

The structure of this paper is as follows. Next section provides some background about the Debian Project and the Debian 2.2 GNU/Linux distribution. After that, we discuss the method we have used for collecting the data shown in this paper. Later on, we offer the results of counting Debian 2.2 (including total counts, counts by language, counts for the largest packages, etc.). The following section offers some comments on the numbers and how they should be understood, and some comparisons with Red Hat Linux and other operating systems. To finish, we include some conclusions and references.

2 Some Background about Debian

The Debian 2.2 GNU/Linux distribution is put together and maintained by the Debian project. In this section, we offer some background data about Debian as a project, and about the Debian 2.2 release.

Jesús M. González Barahona is a lecturer at the *Universidad Rey Juan Carlos*, and collaborator of *BarraPunto.Com*. He began working on the promotion of Free Software in 1991, in the PD-SOFT group (later the *Sobre* group). Since then he has been involved in many activities in this field, such as the organisation of seminars, giving courses and taking part in Free Software working groups. He is currently collaborating on several Free Software projects (Debian and The Espiral among others), he takes part in the Working Group on Free software promoted by the DG-INFO of the European Commission, he collaborates with associations like Hispalinux and EuroLinux, he writes for several publications about Free Software, and he advises companies in their strategies regarding this subject. Co-editor of the Free Software section of *Novática*. <jgb@gsysc.esctet.urjc.es>

Miguel A. Ortuño Pérez is engineer in informatics, professor at the *Universidad Rey Juan Carlos* in Madrid where his domains of interests are mobile computation and Free Software. Previously he worked at the University of Oviedo in various projects related to remote teaching.

José Centeno González is professor at the *Universidad Rey Juan Carlos* in Madrid. He joined the Informatics department of the *Universidad Carlos III* in Madrid in 1993 where he worked until 1999. His research interests include distributed systems programming, communications protocols and mobility. He is also interested in the impact of Free Software in the domain of software engineering and industry. He holds a degree in telecommunication engineering from the *Universidad Politécnica de Madrid* where he expects to obtain the doctor’s degree this year.

Vicente Matellán Oliveras is professor at the *Universidad Rey Juan Carlos* in Madrid. He has been active in the fields of Free Software, among others in the creation of *OpenResources.com* and *BarraPunto.com*.

For instance, the Linux kernel amounts for 1,780,000 SLOC (release 2.2.19) in Debian 2.2, while the same package it amounts for 2,437,000 SLOC (release 2.4.2) in Red Hat 7.1, or XFree includes 1,270,000 SLOC (release 3.3.6) in Debian 2.2, while the release included in Red Hat 7.1 amounts for 1,838,000 (XFree 4.0.3). This differences in releases make it difficult to directly compare the figures for Red Hat and Debian.

The reader should also note that there is a methodological difference between the study on Red Hat and ours on Debian. The former extracts all the source code, and uses MD5 checksums to avoid duplicates across the whole distribution source code. In the case of Debian, we have extracted the packages one by one, only checking for duplicates within packages. However, the total count should not be very affected for this difference.

6 Conclusions and Related Work

It is important to notice that these counts may represent roughly the whole collection of stable Free Software packages available on GNU/Linux at the time of the Debian 2.2 release (August 2000). Of course, there is Free Software not included in Debian, but when we come to popular, stable and usable packages, most of them have been packaged by a Debian developer and included in the Debian distribution. Therefore, with some care, it could be said that this kind of software amounted for about 60,000,000 SLOC around the summer of 2001. Using the COCOMO model, this implies a cost (using traditional, proprietary software development models) close to 2,000 million USD and effort of more than 170,000 person-months.

We can also compare this count to that of other Linux-based distributions, notably Red Hat. Roughly speaking, Debian 2.2 is about twice the size of Red Hat 7.1, which was released about eight months later. It is also larger than the latest Microsoft operating systems (although, as is discussed in the corresponding section, this comparisons could be misleading).

When coming to the details, some interesting data can be shown. For instance, the most popular language in the distribution is C (more than 70%), followed by C++ (close to 10%), LISP and Shell (about 5%), and Perl and FORTRAN (about 2%). The largest packages in Debian 2.2 are Mozilla (about 2,000,000 SLOC), the Linux kernel (about 1,800,000 SLOC), XFree86 (1,250,000), and PM3 (more than 1,100,000).

There are not many detailed studies of the size of modern, complete operating systems. Of them, the work by David Wheeler, counting the size of Red Hat 6.2 and Red Hat 7.1 is the most comparable. Other interesting paper, with some intersection with this paper is [Godfrey/Tu 00], an study on the evolution over time of the Linux kernel. Some other papers,

already referenced, provide total counts of some Sun and Microsoft operating systems, but they are not detailed enough, except for providing estimations for the whole of the system.

Finally, we find it important to repeat once more that we are offering only estimations, not actual figures. They depend too much on the selection of the software to measure, and on some other factors which were already discussed. But we believe they are accurate enough to draw some conclusions, and to compare with other systems.

Acknowledgements

This paper is obviously inspired by "More Than a Gigabuck: Estimating GNU/Linux's Size;", by David Wheeler [Wheeler 01]. We have also used his tool *sloccount*. Without his work, this paper would have been completely unthinkable.

We would also like to thank the comments and suggestions of many Debian developers, which have helped to improve this paper.

Bibliography

- [Boehm 81]
Barry W. Boehm, 1981, Software Engineering Economics, Prentice Hall.
- [ComWorld 00]
Computer World, Salary Survey 2000,
<http://www.computerworld.com/cwi/careers/surveysandreports>.
- [Debian22Ann]
Debian Project, Debian GNU/Linux 2.2, the "Joel 'Espy' Klecker" release, is officially released,
<http://www.debian.org/News/2000/20000815>.
- [DebianPol]
Debian Project, Debian Policy Manual,
<http://www.debian.org/doc/debian-policy/>.
- [Debian22Rel]
Debian Project, Debian GNU/Linux 2.2 release information,
<http://www.debian.org/releases/2.2/>.
- [DFSG]
Debian Project, Debian Free Software Guidelines,
http://www.debian.org/social_contract#guidelines.
- [Godfrey/Tu 00]
Michael W. Godfrey, Qiang Tu, Software Architecture Group (SWAG), Department of Computer Science, University of Waterloo, August 3–4, 2000, Evolution in Open Source Software: A Case Study, 2000 Intl Conference on Software Maintenance
<http://plg.uwaterloo.ca/~migod/papers/icsm00.pdf>.
- [Lucovsky 00]
Mark Lucovsky, August 3–4, 2000, From NT OS/2 to Windows 2000 and Beyond – A Software-Engineering Odyssey, 4th USENIX Windows Systems Symposium, http://www.usenix.org/events/usenix-win2000/invitedtalks/lucovsky_html/.
- [Schneier 00]
Bruce Schneier, March 15, 2000, Software Complexity and Security, Crypto-Gram Newsletter,
<http://www.counterpane.com/crypto-gram-0003.html>.
- [Wheeler 01]
David A. Wheeler, More Than a Gigabuck: Estimating GNU/Linux's Size, <http://www.dwheeler.com/sloc>.

Web Images More... jgbarah@gmail.com

Google

Scholar

[←](#) [Edit](#) [🗑](#) [Export ▾](#)



Jesus M. Gonzalez-Barahona

Counting potatoes: the size of Debian 2.2

[\[PDF\] from gsync.es](#)

Authors: Jesús M González-Barahona, MA Ortuno Perez, Pedro de las Heras Quirós, José Centeno González, Vicente Matellán Olivera

Publication date: 2001/12

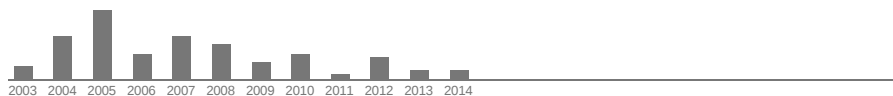
Journal: Upgrade Magazine

Volume: 2

Issue: 6

Pages: 60-66

Total citations: [Cited by 77](#)



Scholar articles: [Counting potatoes: the size of Debian 2.2](#)
JM González-Barahona, MAO Perez... - Upgrade Magazine, 2001
[Cited by 77](#) - [Related articles](#) - [All 2 versions](#)

Dates and citation counts are estimated and are determined automatically by a computer program.

[Help](#) [Privacy](#) [Terms](#) [Provide feedback](#) [My Citations](#)

UPGRADE is the European Journal for the Informatics Professional, published bimonthly at <http://www.upgrade-cepis.org/>

Publisher

UPGRADE is published on behalf of CEPIIS (Council of European Professional Informatics Societies, <http://www.cepis.org/>) by NOVÁTICA <http://www.ati.es/novatica/>, journal of the Spanish CEPIIS society ATI (Asociación de Técnicos de Informática <http://www.ati.es/>).

UPGRADE is also published in Spanish (full issue printed, some articles online) by NOVÁTICA, and in Italian (abstracts and some articles online) by the Italian CEPIIS society ALSI <http://www.alsi.it> and the Italian IT portal Tecnoteca <http://www.tecnoteca.it/>.

UPGRADE was created in October 2000 by CEPIIS and was first published by NOVÁTICA and INFORMATIK/INFORMATIQUE, bimonthly journal of SVI/FSI (Swiss Federation of Professional Informatics Societies, <http://www.svifsi.ch/>).

Editorial Team

Chief Editor: Rafael Fernández Calvo, Spain rfoalvo@ati.es
Associate Editors:

- François Louis Nicolet, Switzerland, nicolet@acm.org
- Roberto Carniel, Italy, carniel@dgt.uniud.it

Editorial Board

Prof. Wolffried Stucky, CEPIIS President
Fernando Piera Gómez and
Rafael Fernández Calvo, ATI (Spain)
François Louis Nicolet, SI (Switzerland)
Roberto Carniel, ALSI – Tecnoteca (Italy)

English Editors: Mike Andersson, Richard Butchart, David Cash, Arthur Cook, Tracey Darch, Laura Davies, Nick Dunn, Rodney Fennemore, Hilary Green, Roger Harris, Michael Hird, Jim Holder, Alasdair MacLeod, Pat Moody, Adam David Moss, Phil Parkin, Brian Robson.

Cover page designed by Antonio Crespo Foix, © ATI 2003

Layout: Pascale Schürmann

E-mail addresses for editorial correspondence:
nicolet@acm.org and rfoalvo@ati.es

E-mail address for advertising correspondence:
novatica@ati.es

Upgrade Newsletter available at

<http://www.upgrade-cepis.org/pages/editinfo.html#newsletter>

Copyright

© NOVÁTICA 2003. All rights reserved. Abstracting is permitted with credit to the source. For copying, reprint, or republication permission, write to the editors.

The opinions expressed by the authors are their exclusive responsibility.

ISSN 1684-5285

Next issue (Oct. 2003):
“e-Learning – Borderless Education”

Software Engineering – State of an Art

Guest Editor: Luis Fernández-Sanz

Joint issue with NOVÁTICA

- 2 Presentation: Software Engineering. A Dream Coming True?
– Luis Fernández-Sanz

The guest editor presents the issue, that focuses on a really broad field like Software Engineering (SE) which has been driving the evolution of software development since the late sixties of the past century. The papers cover different areas of interest related to the application of engineering principles to software development and maintenance. As usual, a list of useful references is also included for those interested in knowing more about this subject.

- 5 Software Project Management. Adding Stakeholder Metrics to Agile Projects
– Tom Gilb

In this article the author offers an analysis of the implications of the new agile methods in the field of software development.

- 10 Model-Driven Development and UML 2.0. The End of Programming as We Know It? – Morgan Björkander

This paper focuses on the idea of a truly model-driven software and analyses the influence that the new version of UML (Unified Modeling Language) is having on this process.

- 15 Component-Based Software Engineering – Alejandra Cechich and Mario Piattini-Velthuis

This paper studies the important role that components play in the field of Software Engineering.

- 21 An Overview of Software Quality – Margaret Ross

The author reviews important issues concerning quality in software development and also deals with the issues of users with disabilities and the influence of legislation regulating this aspect.

- 26 Lessons Learned in Software Process Improvement – José-Antonio Calvo-Manzano Villalón, Gonzalo Cuevas-Agustín, Tomás San Feliu-Gilabert, Antonio de Amescua-Seco, M^a Magdalena Arcilla-Cobián, and José-Antonio Cerrada-Somolinos

This paper describes the lessons learned by SOMEPRO, a Software Engineering R & D group in the Universidad Politécnica de Madrid, in more than ten software process improvement projects.

- 30 A New Method for Simultaneous Application of ISO/IEC 15504 and ISO 9001:2000 in Software SME's – Antònia Mas-Pichaco and Esperança Amengual-Alcover

The authors offer their view, based on practical experiences, of the always thorny problem of applying best software development practices to organizations where resources are especially limited.

- 37 Empirically-based Software Engineering – Martin Shepperd

This paper presents an overview of empirical Software Engineering and its implications for practitioners and researchers in four areas (object-orientation, inspections, formal specification and project failure factors.)

- 42 Software Engineering Professionalism – Luis Fernández-Sanz and María-José García-García

The aim of this paper is to provide a brief overview of what goes into making up our true perception of software engineers as specialised professionals within the field of Information Technologies.

- 47 Searching for the Holy Grail of Software Engineering – Robert L. Glass

In this article, the author defends eclecticism in development methods and the contribution that Software Engineering should make in this respect whenever the nature of a project demands flexible methods in order to be successful.

- 49 Free Software Engineering: A Field to Explore – Jesús M. González-Barahona and Gregorio Robles

This article analyses the existing points of contact between Software Engineering and the development of free software, and puts forward a few future lines of research in this respect.

Free Software Engineering: A Field to Explore

Jesús M. González-Barahona and Gregorio Robles

The challenge of free software is not that of a new competitor who, under the same rules, produces software faster, cheaper and of a better quality. Free software differs from ‘traditional’ software in more fundamental aspects, starting with philosophical reasons and motivations, continuing with new economic and market rules and ending up with a different way of producing software. Software Engineering cannot ignore this phenomenon, and the last five years or so has seen ever more research into all these issues. This article takes a look at the most significant studies in this field and the results they are producing, with a view to providing the reader with a vision of the state of the art and the future prospects of what we have come to call free Software Engineering.

Keywords: Software Engineering, free Software Engineering, free software.

1 Introduction

Although free software¹ has been developed for several decades, only for the last few years have we begun to pay attention to its development models and processes from the point of view of Software Engineering. Just as there is no single development model for proprietary software, neither is there only one in the free software world [11], but, that said, some interesting features are shared by a large number of the projects we have looked at, features which may be at the root of free software.

In 1997 Eric S. Raymond published his first, widely read article “The cathedral and the bazaar” [18], in which he describes some characteristics of free software development models, laying great stress on what differentiates these models from proprietary development models. Since then that article has become one of the best known (and most criticised) in the free software world, and to a certain extent, was the starting pistol for the development of free Software Engineering.

2 The Cathedral and the Bazaar

Raymond makes an analogy between the way mediaeval cathedrals were built and the classic way of producing software. He argues that in both cases there is a clear distribution of tasks and roles, with the designer on top of everything, controlling the process from beginning to end. Planning is strictly controlled in both cases, giving rise to clearly defined processes in which, ideally, everyone taking part in the activity has a very limited and specific role to play.

1. **Note from the Editor of Upgrade:** Our editorial policy is to continue to use, in English, the term ‘free software’, though we are aware that the term ‘open source software’, or simply ‘open source’, appears to be winning the battle; ‘libre software’ is also gaining popularity because it avoids the ambiguity of the English word ‘free’.

Included in what Raymond views as the cathedral building model are not only the heavyweight processes of the software industry (the classic cascade model, the different aspects of the Rational Unified Process, etc.), but also some free software projects, as is the case of GNU, <<http://www.gnu.org/>>, and NetBSD, <<http://www.NetBSD.org/>>. According to Raymond, these projects are highly centralized, since only a few people are responsible for the design and implementation of the software. The tasks these people perform and the roles they play are perfectly defined, and anyone wanting to join the development team would need to be assigned a task and a role according to the needs of the project. Another feature is that releases of this type of programme tend to be spaced out over time, following a fairly strict schedule. This leads to having few releases of the software with lengthy intervals between each


Jesús M. González Barahona is a professor at the Universidad Rey Juan Carlos, Madrid, Spain. He researches in the field of distributed systems and large scale peer to peer computing. He is also interested in free / open source Software Engineering. He began working on the promotion of free software in 1991. He is currently collaborating on several free software projects (including Debian), and he collaborates with associations such as Hispalinux and EuroLinux, writes in several media on free software related matters, and advises companies on their strategies related to these issues. He is a member of ATI, coordinator of the Free Software section of Novática, and has also been a guest editor of several monographs in Novática and Upgrade.
<jgb@gsync.escet.urjc.es>

Gregorio Robles is a professor at the Universidad Rey Juan Carlos, Madrid, Spain. His research work is centred on the study of free software development from an engineering point of view and especially with regard to quantitative issues. He has developed or collaborated on the design of programmes to automate the analysis of free software and the tools used to produce them. He was also involved in the FLOSS study of free software financed by the European Commission IST programme.
<greg@gsync.escet.urjc.es>

References

- [1] Antoniadis Ioannis, Samoladas Ioannis, Stamelos Ioannis, and G. L. Bleris. Dynamical simulation models of the Open Source Development process, 2003, pending publication in *Free/Open Source Software Development*, published by Stefan Koch, Idea Inc, Vienna.
- [2] Nikolai Bezroukov. A Second Look at the Cathedral and the Bazaar, December 1998. <http://www.firstmonday.dk/issues/issue4_12/bezroukov/index.html>.
- [3] Barry W. Boehm, 1981. *Software Engineering Economics*, Prentice Hall.
- [4] Frederick P. Brooks Jr., 1975. *The Mythical Man-Month: Essays on Software Engineering*, Addison-Wesley.
- [5] Justin R. Ehrenkrantz. Release Management Within Open Source Projects, May 2003. <<http://opensource.ucc.ie/icse2003/3rd-WS-on-OSS-Engineering.pdf>>.
- [6] Jesús M. González Barahona, Miguel A. Ortuño Pérez, Pedro de las Heras Quirós, José Centeno González, and Vicente Matellán Olivera. Counting potatoes. The size of Debian 2.2, Upgrade, vol. 2, issue 6, December 2001. <<http://upgrade-cepis.org/issues/2001/6/up2-6Gonzalez.pdf>>. Also available at <<http://people.debian.org/~jgb/debian-counting/>>.
- [7] Jesús M. González Barahona and Gregorio Robles. Unmounting the code god assumption, Mayo 2003, Proceedings of the Fourth International Conference on eXtreme Programming and Agile Processes in Software Engineering (Genoa, Italia). <<http://libresoft.dat.escet.urjc.es/html/downloads/xp2003-barahona-robles.pdf>>.
- [8] Jesús M. González Barahona, Gregorio Robles, Miguel A. Ortuño Pérez, Luis Roderó Merino, José Centeno González, Vicente Matellán Olivera, Eva M. Castro Barbero, and Pedro de las Heras Quirós. Anatomy of two GNU/Linux distributions, 2003”, pending publication in “Free/Open Source Software Development, published by Stefan Koch, Idea Inc, Vienna.
- [9] Daniel Germán and Audris Mockus. Automating the Measurement of Open Source Projects, May 2003. <<http://opensource.ucc.ie/icse2003/3rd-WS-on-OSS-Engineering.pdf>>.
- [10] Rishab Aiyer Ghosh and Vipul Ved Prakash, The Orbiten Free Software Survey, May 2000. <http://www.firstmonday.dk/issues/issue5_7/ghosh/index.html>.
- [11] Kieran Healy and Alan Schussman, The Ecology of Open Source Software Development, January 2003. <<http://opensource.mit.edu/papers/healyschussman.pdf>>.
- [12] Paul Jones. Brooks’ Law and open source: The more the merrier?, May 2000, <<http://www-106.ibm.com/developerworks/opensource/library/os-merrier.html?dwzone=opensource>>.
- [13] Stefan Koch and Georg Schneider. Results from Software Engineering Research into Open Source Development Projects Using Public Data, 2000. <<http://www.wai.wu-wien.ac.at/~koch/forschung/sw-eng/wp22.pdf>>.
- [14] Sandeep Krishnamurthy. Cave or Community? An Empirical Examination of 100 Mature Open Source Projects, May 2002. <<http://opensource.mit.edu/papers/krishnamurthy.pdf>>.
- [15] Jesús M. González Barahona and Gregorio Robles Martínez. Libre Software Engineering. <<http://libresoft.dat.escet.urjc.es/>>.
- [16] Audris Mockus, Roy T. Fielding, and James D. Herbsleb. Two Case Studies of Open Source Software Development: Apache and Mozilla, Junio 2000. <<http://www.research.avayalabs.com/techreport/ALR-2002-003-paper.pdf>>.
- [17] Alessandro Narduzzo and Alessandro Rossi. Modularity in Action: GNU/Linux and Free/Open Source Software Development Model Unleashed, May 2003. <<http://opensource.mit.edu/papers/narduzzorossi.pdf>>.
- [18] Eric S. Raymond. The Cathedral and the Bazaar, Musings on Linux and Open Source by an Accidental Revolutionary, May 1997. <<http://catb.org/~esr/writings/cathedral-bazaar/>>.
- [19] Christian Robottom Reis and Renata Pontin de Mattos Fortes. An Overview of the Software Engineering Process and Tools in the Mozilla Project, February 2002. <<http://opensource.mit.edu/papers/reismozilla.pdf>>.
- [20] Gregorio Robles, Jesús González Barahona, José Centeno González, Vicente Matellán Olivera, and Luis Roderó Merino. Studying the evolution of libre software projects using publicly available data, May 2003, Proceedings of the 3rd Workshop on Open Source Software Engineering at the 25th International Conference on Software Engineering. <<http://opensource.ucc.ie/icse2003/3rd-WS-on-OSS-Engineering.pdf>>.
- [21] Ilkka Tuomi. Internet, Innovation, and Open Source: Actors in the Network, 2001. <http://www.firstmonday.dk/issues/issue6_1/tuomi/>.
- [22] Paul Vixie. *Software Engineering*, 1999. <<http://www.oreilly.com/catalog/opensources/book/vixie.html>>.
- [23] David A. Wheeler. Estimating Linux’s Size, July 2000. <<http://www.dwheeler.com/sloc>>.
- [24] David A. Wheeler. More Than a Gigabuck: Estimating GNU/Linux’s Size, June 2001. <<http://www.dwheeler.com/sloc>>.

Web Images More... jgbarah@gmail.com



Scholar ← Edit 🗑️ Export ▾



Jesus M. Gonzalez-Barahona

Free software engineering: A field to explore

[\[PDF\]](#) from ifipwg213.org

Authors: Jesús M González-Barahona, Gregorio Robles
 Publication date: 2003/8
 Journal: Upgrade
 Volume: 4
 Issue: 4
 Pages: 49-54

Description: The challenge of free software is not that of a new competitor who, under the same rules, produces software faster, cheaper and of a better quality. Free software differs from 'traditional' software in more fundamental aspects, starting with philosophical reasons and motivations, continuing with new economic and market rules and ending up with a different way of producing software. Software Engineering cannot ignore this phenomenon, and the last five years or so has seen ever more research into all these issues. This article takes a ...

Total citations: [Cited by 23](#)



Scholar articles: [Free software engineering: A field to explore](#)
 JM González-Barahona, G Robles - Upgrade, 2003
[Cited by 23](#) - [Related articles](#) - [All 12 versions](#)

Dates and citation counts are estimated and are determined automatically by a computer program.

[Help](#) [Privacy](#) [Terms](#) [Provide feedback](#) [My Citations](#)

UPGRADE is the European Journal for the Informatics Professional, published bimonthly at <http://www.upgrade-cepis.org/>

UPGRADE is the anchor point for UPENET (UPGRADE European Network), the network of CEPIS member societies' publications, that currently includes the following ones:

- **Mondo Digitale**, digital journal from the Italian CEPIS society AICA
- **Novática**, journal from the Spanish CEPIS society ATI
- **OCG Journal**, journal from the Austrian CEPIS society OCG
- **Pliroforiki**, journal from the Cyprus CEPIS society CCS
- **Pro Dialog**, journal from the Polish CEPIS society PTI-PIPS

Publisher

UPGRADE is published on behalf of CEPIS (Council of European Professional Informatics Societies, <http://www.cepis.org/>) by **Novática** (<http://www.ati.es/novatica/>), journal of the Spanish CEPIS society ATI (*Asociación de Técnicos de Informática*, <http://www.ati.es/>)

UPGRADE monographs are also published in Spanish (full version printed; summary, abstracts and some articles online) by **Novática**, and in Italian (summary, abstracts and some articles online) by the Italian CEPIS society ALSI (*Associazione nazionale Laureati in Scienze dell'informazione e Informatica*, <http://www.alsi.it/>) and the Italian IT portal **Tecnoteca** (<http://www.tecnoteca.it/>)

UPGRADE was created in October 2000 by CEPIS and was first published by **Novática** and **INFORMATIK/INFORMATIQUE**, bimonthly journal of SVI/FSI (Swiss Federation of Professional Informatics Societies, <http://www.svifs.ch/>)

Editorial Team

Chief Editor: Rafael Fernández Calvo, Spain, rfcalvo@ati.es

Associate Editors:

François Louis Nicolet, Switzerland, nicolet@acm.org
 Roberto Carniel, Italy, rcarniel@dgt.uniud.it
 Zakaria Maamar, Arab Emirates, <Zakaria.Maamar@zu.ac.ae>
 Soraya Kouadri Mostéfaoui, Switzerland, soraya.kouadrimostefaoui@unifr.ch

Editorial Board

Prof. Wolfried Stucky, CEPIS Past President
 Prof. Nello Scarabottolo, CEPIS Vice President
 Fernando Piera Gómez and
 Rafael Fernández Calvo, ATI (Spain)
 François Louis Nicolet, SI (Switzerland)
 Roberto Carniel, ALSI – Tecnoteca (Italy)

UPENET Advisory Board

Franco Filippazzi (Mondo Digitale, Italy)
 Rafael Fernández Calvo (Novática, Spain)
 Veith Risak (OCG Journal, Austria)
 Panicos Masouras (Pliroforiki, Cyprus)
 Andrzej Marciniak (Pro Dialog, Poland)

English Editors: Mike Andersson, Richard Butchart, David Cash, Arthur Cook, Tracey Darch, Laura Davies, Nick Dunn, Rodney Fennemore, Hilary Green, Roger Harris, Michael Hird, Jim Holder, Alasdair MacLeod, Pat Moody, Adam David Moss, Phil Parkin, Brian Robson

Cover page designed by Antonio Crespo Foix, © ATI 2005

Layout Design: François Louis Nicolet

Composition: Jorge Llácer-Gil de Ramales

Editorial correspondence: Rafael Fernández Calvo rfcalvo@ati.es

Advertising correspondence: novatica@ati.es

UPGRADE **Newsletter** available at

<http://www.upgrade-cepis.org/pages/editinfo.html#newsletter>

Copyright

© Novática 2005 (for the monograph and the cover page)

© CEPIS 2005 (for the sections MOSAIC and UPENET)

All rights reserved. Abstracting is permitted with credit to the source. For copying, reprint, or republication permission, contact the Editorial Team

The opinions expressed by the authors are their exclusive responsibility

ISSN 1684-5285

Monograph of next issue (August 2005):

"Normalisation & Standardisation in IT Security"

(The full schedule of UPGRADE is available at our website)

Monograph: *Libre Software as A Field of Study* (published jointly with *Novática**, in cooperation with the European project CALIBRE)

Guest Editors: *Jesús M. González-Barahona and Stefan Koch*

- 2 Presentation
Libre Software under The Microscope — *Jesús M. González-Barahona and Stefan Koch*
- 5 CALIBRE at The Crest of European Open Source Software Wave — *Andrea Deverell and Par Agerfalk*
- 6 *Libre Software Movement: The Next Evolution of The IT Production Organization?* — *Nicolas Jullien*
- 13 Measuring *Libre Software* Using Debian 3.1 (Sarge) as A Case Study: Preliminary Results — *Juan-José Amor-Iglesias, Jesús M. González-Barahona, Gregorio Robles-Martínez, and Israel Herráiz-Tabernero*
- 17 An Institutional Analysis Approach to Studying *Libre Software* 'Commons' — *Charles M. Schweik*
- 28 About Closed-door Free/*Libre*/Open Source (FLOSS) Projects: Lessons from the Mozilla Firefox Developer Recruitment Approach — *Sandeep Krishnamurthy*
- 33 Agility and *Libre Software* Development — *Alberto Sillitti and Giancarlo Succi*
- 38 The Challenges of Using Open Source Software as A Reuse Strategy — *Christian Neumann and Christoph Breidert*

MOSAIC

- 43 Computational Linguistics
Multilingual Approaches to Text Categorisation — *Juan-José García-Adeva, Rafael A. Calvo, and Diego López de Ipiña*
- 52 Software Engineering
A Two Parameter Software Reliability Growth Model with An Implicit Adjustment Factor for Better Software Failure Prediction — *S. Venkateswaran, K. Ekambavanan, and P. Vivekanandan*
- 59 News & Events: Proposal of Directive on Software Patents Rejected by The European Parliament

UPENET (UPGRADE European Network)

- 61 From **Pliroforiki** (CCS, Cyprus)
Informatics Law
Security, Surveillance and Monitoring of Electronic Communications at The Workplace — *Olga Georgiades-Van der Pol*
- 66 From **Mondo Digitale** (AICA, Italy)
Evolutionary Computation
Evolutionary Algorithms: Concepts and Applications — *Andrea G. B. Tettamanzi*

* This monograph will be also published in Spanish (full version printed; summary, abstracts, and some articles online) by **Novática**, journal of the Spanish CEPIS society ATI (*Asociación de Técnicos de Informática*) at <http://www.ati.es/novatica/>, and in Italian (online edition only, containing summary, abstracts, and some articles) by the Italian CEPIS society ALSI (*Associazione nazionale Laureati in Scienze dell'informazione e Informatica*) and the Italian IT portal Tecnoteca at <http://www.tecnoteca.it/>.

Measuring *Libre* Software Using Debian 3.1 (Sarge) as A Case Study: Preliminary Results

Juan-José Amor-Iglesias, Jesús M. González-Barahona, Gregorio Robles-Martínez, and Israel Herráiz-Tabernero



This paper is copyrighted under the Creative Commons Attribution-NonCommercial-NoDerivs 2.5 license available at <http://creativecommons.org/licenses/by-nc-nd/2.5/>

The Debian operating system is one of the most popular GNU/Linux distributions, not only among end users but also as a basis for other systems. Besides being popular, it is also one of the largest software compilations and thus a good starting point from which to analyse the current state of libre (free, open source) software. This work is a preliminary study of the new Debian GNU/Linux release (3.1, codenamed Sarge) which was officially announced recently. In it we show the size of Debian in terms of lines of code (close to 230 million source lines of code), the use of the various programming languages in which the software has been written, and the size of the packages included within the distribution. We also apply a 'classical' and well-known cost estimation method which gives an idea of how much it would cost to create something on the scale of Debian from scratch (over 8 billion USD).

Keywords: COCOMO, Debian, *Libre* Software, *Libre* Software Engineering, Lines of Code, Linux.

1 Introduction

On June 6, 2005, the Debian Project announced the official release of the Debian GNU/Linux version 3.1, codenamed "Sarge", after almost three years of development [6]. The Debian distribution is produced by the Debian project, a group of nearly 1,400 volunteers (a.k.a. maintainers) whose main task is to adapt and package all the software included in the distribution [11]. Debian maintainers package software which they obtain from the original (upstream) authors, ensuring that it works smoothly with the rest of the programs in the Debian system. To ensure this, there is a set of rules that a package should comply with, known as the Debian Policy Manual [5].

Debian 3.1 includes all the major *libre* software packages available at the time of its release. In its main distribution alone, composed entirely of *libre* software (according to Debian Free Software Guidelines), there are more than 8,600 source packages. The whole release comprises almost 15,300 binary packages, which users can install easily from various media or via the Internet.

In this paper we analyse the system, showing its size and comparing it to other contemporary GNU/Linux systems¹. We decided to write this paper as an update of *Counting Potatoes* (see [8]), and *Measuring Woody* (see [1]) which were prompted by previous Debian releases. The paper is structured as follows. The first section briefly presents the methods we used for collecting the data used in this paper. Later, we present the results of our Debian 3.1 count (including total counts, counts by language, counts for the largest packages, etc.). The following section provides some comments on these figures and how they should be interpreted and some comparisons with Red Hat Linux distributions and other free and proprietary operating systems. We close with some conclusions and references.

2 Collecting The Data

In this work we have considered only the **main** distribution, which is the most important and by far the largest

part of any Debian release. It is composed exclusively of free software (according to Debian Free Software Guidelines, DFSG [7]). Other sections, such as **non-free** or

Juan-José Amor-Iglesias has an MSc in Computer Science from the *Universidad Politécnica de Madrid*, Spain, and he is currently pursuing a PhD at the *Universidad Rey Juan Carlos* in Madrid, Spain. Since 1995 he has collaborated in several free software related organizations: he is a co-founder of LuCAS, the best known free documentation portal in Spanish, and Hispalinux, and collaborates with Barrapunto.com. <jjamor@gsyc.esctet.urjc.es>

Jesús M. González-Barahona teaches and researches at the *Universidad Rey Juan Carlos*, Madrid, Spain. He started working in the promotion of *libre* software in the early 1990s. Since then he has been involved in several activities in this area, such as the organization of seminars and courses, and the participation in working groups on *libre* software. He currently collaborates in several *libre* software projects (including Debian), and participates in or collaborates with associations related to *libre* software. He writes in several media about topics related to *libre* software, and consults for companies on matters related to their strategy regarding these issues. His research interests include *libre* software engineering and, in particular, quantitative measures of *libre* software development and distributed tools for collaboration in *libre* software projects. He is editor of the Free Software section of *Novática* since 1997 and has been guest editor of several monographs of *Novática* and *UPGRADE* on the subject. <jgb@gsyc.esctet.urjc.es>

Gregorio Robles-Martínez is a PhD candidate at the *Universidad Rey Juan Carlos* in Madrid, Spain. His main research interest lies in *libre* software engineering, focusing on acquiring knowledge of *libre* software and its development through the study of quantitative data. He was formerly involved in the FLOSS project and now participates in the CALIBRE coordinated action and the FLOSSWorld project, all European Commission IST-program sponsored projects. <grex@gsyc.esctet.urjc.es>

Israel Herráiz-Tabernero has an MSc in Chemical and Mechanical Engineering, a BSc in Chemical Engineering and he is currently pursuing his PhD in Computer Science at the *Universidad Rey Juan Carlos* in Madrid, Spain. He 'discovered' free software in 2000, and has since developed several free tools for chemical engineering. <herraiz@gsyc.esctet.urjc.es>

¹ GNU/Linux systems are also known as 'distributions'.

distributions in general, and Debian 3.1 in particular, are some of the largest pieces of software ever put together by a group of maintainers.

5 Conclusions and Related Work

Debian is one of the largest software systems in the world, probably the largest. Its size has grown with every release, 3.1 being twice the size of 3.0. For the last few releases, the main languages used to develop packages included in Debian are C and C++. In fact C, C++ and Shell represent more than 75% of all source code in Debian. The number of packages continues to grow steadily, doubling almost every two years.

The Debian GNU/Linux distribution, put together by a group of volunteers dispersed all over the world, would, at first sight, appear to show a healthy and fast-growing trend. Despite its enormous size it continues to deliver stable releases. However, there are some aspects that put into doubt the future sustainability of this progress. For instance, mean package size is showing an unstable behaviour, probably due to the number of packages growing faster than the number of maintainers. Nor can we forget that we have had to wait almost three years for a new stable release and that the release date has been seriously delayed on several occasions.

Regarding other software systems, there are few detailed studies of the size of modern, complete operating systems. The work by David A. Wheeler, counting the size of Red Hat 6.2 and Red Hat 7.1 is perhaps the most comparable. Some other references provide total counts of some Sun and Microsoft operating systems, but while they do provide estimates for the system as a whole, they are not detailed enough. Debian is by far the largest of them, although this comparison has to be taken with a degree of caution.

To conclude, it is important to stress that this paper aims to provide estimations based only on a preliminary study (since the release is not yet officially published). However, we believe they are accurate enough to allow us to draw some conclusions and compare them with other systems.

| Operating System | Source Lines of Code (SLOC) |
|--|-----------------------------|
| Microsoft Windows 3.1 (April 1992) | 3,000,000 |
| Sun Solaris (October 1998) | 7,500,000 |
| Microsoft Windows 95 (August 1995) | 15,000,000 |
| Red Hat Linux 6.2 (March 2000) | 17,000,000 |
| Microsoft Windows 2000 (February 2000) | 29,000,000 |
| Red Hat Linux 7.1 (April 2001) | 30,000,000 |
| Microsoft Windows XP (2002) | 40,000,000 |
| Red Hat Linux 8.0 (September 2002) | 50,000,000 |
| Fedora Core 4 (previous version; May 2005) | 76,000,000 |
| Debian 3.0 (July 2002) | 105,000,000 |
| Debian 3.1 (June 2005) | 229,500,000 |

Table 2: Size Comparison of Several Operating Systems.

Acknowledgements

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

References

- [1] Juan José Amor, Gregorio Robles y Jesús M. González-Barahona. *Measuring Woody: The size of Debian 3.0* (pending publication). Will be available at <<http://people.debian.org/~jgb/debian-counting/>>.
- [2] Barry W. Boehm. *Software Engineering Economics*, Prentice Hall, 1981.
- [3] Computer World, *Salary Survey 2000*. <<http://www.computerworld.com/cwi/careers/surveysandreports>>.
- [4] Jesús M. González Barahona, Gregorio Robles, and Juan José Amor, *Debian Counting*. <<http://libresoft.urjc.es/debian-counting/>>.
- [5] Debian Project, *Debian Policy Manual*. <<http://www.debian.org/doc/debian-policy/>>.
- [6] Debian Project, *Debian GNU/Linux 3.1 released (June 6th 2005)*. <<http://www.debian.org/News/2005/20050606>>.
- [7] Debian Project, *Debian Free Software Guidelines (part of the Debian Social Contract)*. <http://www.debian.org/social_contract>.
- [8] Jesús M. González Barahona, Miguel A. Ortuño Pérez, Pedro de las Heras Quirós, José Centeno González, and Vicente Matellán Olivera. *Counting potatoes: The size of Debian 2.2*. UPGRADE, vol. 2, issue 6, December 2001, <<http://upgrade-cepis.org/issues/2001/6/up2-6Gonzalez.pdf>>; Novática, nº 151 (nov.-dic. 2001), <<http://www.ati.es/novatica/2001/154/154-30.pdf>> (in Spanish).
- [9] Jesús M. González-Barahona, Gregorio Robles, Miguel Ortuño-Pérez, Luis Roderó-Merino, José Centeno-González, Vicente Matellán-Olivera, Eva Castro-Barbero, and Pedro de-las-Heras-Quirós. *Anatomy of two GNU/Linux distributions*. Chapter in book "Free/Open Source Software Development" edited by Stefan Koch and published by Idea Group, Inc., 2004.
- [10] Gregorio Robles, Jesús M. González-Barahona, Luis López, and Juan José Amor, *Toy Story: an analysis of the evolution of Debian GNU/Linux*, November 2004 (pending publication). Draft available at <<http://libresoft.urjc.es/debian-counting/>>.
- [11] Gregorio Robles, Jesús M. González-Barahona, and Martin Michlmayr. *Evolution of Volunteer Participation in Libre Software Projects: Evidence from Debian*, julio 2005, Proceedings of the First International Conference on Open Source Systems. Genova, Italy, pp. 100-107. <<http://gsyc.escet.urjc.es/~grex/volunteers-robles-jgb-martin.pdf>>.
- [12] David Wheeler. *SLOCCount*. <<http://www.dwheeler.com/sloccount/>>.
- [13] David A. Wheeler. *More Than a Gigabuck: Estimating GNU/Linux's Size*. <<http://www.dwheeler.com/sloc>>.



Scholar



Jesus M. Gonzalez-Barahona

Measuring libre software using Debian 3.1 (Sarge) as a case study: preliminary results [\[PDF\]](#) from [thalix.com](#)

Authors Juan-José Amor-Iglesias, Jesús M González-Barahona, Gregorio Robles-Martínez, Israel Herráiz-Tabernero

Publication date 2005/6

Journal Upgrade Magazine, August

Description On June 6, 2005, the Debian Project announced the official release of the Debian GNU/Linux version 3.1, codenamed "Sarge", after almost three years of development [6]. The Debian distribution is produced by the Debian project, a group of nearly 1,400 volunteers (aka maintainers) whose main task is to adapt and package all the software included in the distribution [11]. Debian maintainers package software which they obtain from the original (upstream) authors, ensuring that it works smoothly with the rest of the ...

Total citations [Cited by 42](#)



Scholar articles [Measuring libre software using Debian 3.1 \(Sarge\) as a case study: preliminary results](#)
 JJ Amor-Iglesias, JM González-Barahona... - Upgrade Magazine, August, 2005
[Cited by 42](#) - [Related articles](#) - [All 8 versions](#)

Dates and citation counts are estimated and are determined automatically by a computer program.

[Help](#) [Privacy](#) [Terms](#) [Provide feedback](#) [My Citations](#)

Applying Social Network Analysis Techniques to Community-Driven Libre Software Projects

Luis López-Fernández, Universidad Rey Juan Carlos, Spain

Gregorio Robles, Universidad Rey Juan Carlos, Spain

Jesus M. Gonzalez-Barahona, Universidad Rey Juan Carlos, Spain

Israel Herraiz, Universidad Rey Juan Carlos, Spain*

ABSTRACT

Source code management repositories of large, long-lived libre (free, open source) software projects can be a source of valuable data about the organizational structure, evolution, and knowledge exchange in the corresponding development communities. Unfortunately, the sheer volume of the available information renders it almost unusable without applying methodologies which highlight the relevant information for a given aspect of the project. Such methodology is proposed in this article, based on well known concepts from the social networks analysis field, which can be used to study the relationships among developers and how they collaborate in different parts of a project. It is also applied to data mined from some well known projects (Apache, GNOME, and KDE), focusing on the characterization of their collaboration network architecture. These cases help to understand the potentials of the methodology and how it is applied, but also shows some relevant results which open new paths in the understanding of the informal organization of libre software development communities.

Keywords: community-driven development; mining software repositories; social networks analysis; software understanding

INTRODUCTION

Software projects are usually the collective work of many developers. In most cases, and especially in the case of large projects, those developers are formally organized in a well defined (usually hierarchical) structure, with clear guidelines about how to interact with each

other, and the procedures and channels to use. Each team of developers is assigned certain modules of the project, and only in rare cases do they work outside that realm. However, this is usually not the case with libre software¹ projects, where only loose (if any) formal structures are acknowledged. On the contrary, libre software

Luis López obtained his PhD in electrical and electronic engineering at Universidad Rey Juan Carlos in 2003 and his MS in electrical and electronic engineering at Universidad Politécnica de Madrid and at ENST Télécom-Paris in 1998. He is author of more than 50 publications including 10 papers published in different research international journals and 20 contributions to conferences and workshops.

Gregorio Robles received his Telecommunication Engineering degree from the Universidad Politécnica de Madrid (2001) and has recently defended his PhD thesis at the Universidad Rey Juan Carlos (2006). His research work is centered on the empirical study of libre software development, especially from but not limited to a software engineering perspective. He has developed or collaborated in the design and implementation of software programmes to automate the analysis of libre software and the tools used to produce them. He has also been involved in several projects related to the study and promotion of libre software financed by the European Commission IST programmes, such as FLOSS (2000-1), CALIBRE (2004-6) or FLOSSWorld (2005-7).

Jesus M. Gonzalez-Barahona teaches and researches in Universidad Rey Juan Carlos, Mostoles (Spain). He started to be involved in the promotion of libre software in 1991. Since then, he has carried on several activities in this area, including the organization of seminars and courses, and the participation in working groups on libre software, both at the Spanish and European levels. Currently he collaborates with several libre software projects (including Debian) and associations, writes in several media about topics related to libre software, and consults for companies and public administrations on issues related to their strategy on these topics. His research interests include libre software engineering, and in particular quantitative measures of libre software development and distributed tools for collaboration in libre software projects. In this area, he has published several papers, and is participating in some international research projects (more info in <http://libresoft.urjc.es>). He is also one of the promoters of the idea of an European master program on libre software, and has specific interest in the education in that area.

Israel Herraiz holds a MSc in chemical and mechanical engineering, a BSc in chemical engineering and he is currently pursuing his PhD in computer science at the Universidad Rey Juan Carlos in Madrid, Spain. He discovered free software in 2000, and has since then developed several free tools for chemical engineering.

Computing journals sorted by alphabetical title

| | | | |
|-------|------------|--|------|
| 32087 | C | International Journal of Information Technology and Management | 0806 |
| 32088 | Not ranked | International Journal of Information Technology and the Systems Approach | 0806 |
| 31261 | C | International Journal of Information Technology and Web Engineering | 0805 |
| 32089 | C | International Journal of Information Technology Education | 0899 |
| 41255 | C | International Journal of Innovative Computing and Applications | 0801 |
| 40913 | Not ranked | International Journal of Intelligent Defence Support Systems | 1099 |
| 41257 | C | International Journal of Intelligent Information and Database Systems | 0801 |
| 31262 | C | International Journal of Intelligent Information Technologies | 08 |
| 17950 | B | International Journal of Intelligent Systems | 0801 |
| 40428 | B | International Journal of Intelligent Systems Technologies and Applications | 0906 |
| 32091 | C | International Journal of Intelligent Technology | 0801 |
| 31263 | C | International Journal of Internet and Enterprise Management | 0805 |
| 32092 | C | International Journal of Internet Protocol Technology | 0805 |
| 32093 | C | International Journal of Internet Science | 0899 |
| 41260 | C | International Journal of Internet Technology and Secured Transactions | 0804 |
| 32094 | C | International Journal of Interoperability in Business Information Systems | 0806 |
| 32095 | C | International Journal of IT Standards and Standardization Research | 0806 |
| 20201 | C | International Journal of Knowledge and Learning | 0807 |
| 17951 | C | International Journal of Knowledge Management | 0804 |
| 31264 | C | International Journal of Knowledge Management Studies | 0806 |
| 17183 | C | International Journal of Law and Information Technology | 0899 |
| 42185 | C | International Journal of Man-Machine Studies | 0806 |
| 42184 | C | International Journal of Management and Systems | 0806 |
| 41273 | Not ranked | International Journal of Medical Engineering and Informatics | 0806 |
| 13602 | A | International Journal of Medical Informatics | 0807 |
| 32097 | C | International Journal of Metadata Semantics and Ontologies | 0804 |
| 32098 | C | International Journal of Mobile Communications | 0805 |
| 41277 | C | International Journal of Mobile Network Design and Innovation | 0805 |
| 17955 | C | International Journal of Modelling and Simulation | 0802 |
| 41278 | C | International Journal of Modelling, Identification and Control | 0801 |
| 5087 | C | International Journal of Network Management | 1005 |
| 32099 | B | International Journal of Neural Systems | 0801 |
| 17957 | A | International Journal of Parallel Programming | 0805 |
| 18101 | B | International Journal of Parallel, Emergent and Distributed Systems | 0802 |
| 17958 | B | International Journal of Pattern Recognition and Artificial Intelligence | 0801 |

CORE Extract of Journals
allocated codes
of 08 to 0899

MEDIA



International Journal of Information Technology and Web Engineering (IJITWE)

An Official Publication of the [Information Resources Management Association](#)

[Ghazi I. Alkhatib](#) (Princess Sumaya University for Technology, Jordan) and [Ernesto Damiani](#) (University of Milan, Italy)

Published Quarterly, Est. 2006.

Select a Format: **Institutions: Print** Subscription Year: **2013** **\$595.00** [Add to Cart](#)

DOI: 10.4018/IJITWE, ISSN: 1554-1045, EISSN: 1554-1053

[Cite Journal](#) [Favorite](#)

[Send](#) [Me gusta](#) [Tweet](#)

[Description](#) | [Contents](#) | [Mission](#) | [Reviews & Testimonials](#) | [Indices](#) | [Topics Covered](#) | [Editor\(s\)-in-Chief Bio](#) | [Editorial Board](#)

Free Content

Sample Issue: [IJITWE4\(1\)](#)

More Information

- [Search this Journal](#)
- [Call for Papers](#)
- [Guidelines for Submission](#)
- [Subscribe](#)
- [Current Issue](#)

Recommend

- [Send to a librarian](#)
- [Send to a colleague](#)

Available In

[InfoSci-Journals](#)

Browse Subjects

- [Business Technologies](#)
- [Database Technologies](#)
- [E-Government](#)
- [Educational Technologies](#)
- [Intelligent Technologies](#)
- [Knowledge Management](#)
- [Medical Technologies](#)
- [Multimedia Technologies](#)
- [Security Technologies](#)
- [Social Technologies](#)
- [Software & Engineering](#)

Description

Organizations are continuously overwhelmed by a variety of new information technologies, many are Web based. These new technologies are capitalizing on the widespread use of network and communication technologies for seamless integration of various issues in information and knowledge sharing within and among organizations. This emphasis on integrated approaches is unique to this journal and dictates cross platform and multidisciplinary strategy to research and practice.

Journal Contents

- [Volume 7: 2 Issues \(2012\)](#)
- [Volume 6: 4 Issues \(2011\)](#)
- [Volume 5: 4 Issues \(2010\)](#)
- [Volume 4: 4 Issues \(2009\)](#)
- [Volume 3: 4 Issues \(2008\)](#)
- [Volume 2: 4 Issues \(2007\)](#)
- [Volume 1: 4 Issues \(2006\)](#)

[View Complete Journal Contents Listing](#)

Mission

The main objective of the **International Journal of Information Technology and Web Engineering (IJITWE)** is to publish refereed papers in the area covering information technology (IT) concepts, tools, methodologies, and ethnography in the contexts of global communication systems and Web engineered applications. In accordance with this emphasis on the Web and communication systems, this journal publishes papers on IT research and practice that support seamless end-to-end information and knowledge flow among individuals, teams, and organizations. This end-to-end strategy for research and practice requires emphasis on integrated research among the various steps involved in data/knowledge (structured and unstructured) capture (manual or automated), classification and clustering, storage, analysis, synthesis, dissemination, display, consumption, and feedback. The secondary objective is to assist in the evolving and maturing of IT-dependent organizations, as well as individuals, in information and knowledge based culture and commerce, including e-commerce.

Reviews and Testimonials

With the increasing reliance on Web-based systems, individuals and organizations, and developing societies in general are looking for new ways to share experiences, knowledge, and innovative methodologies through the different technologies and innovations of Web engineering. Researchers, practitioners, and academicians will find in the *International Journal of IT and Web Engineering* an outlet to publish and share their original work and experiences on the use of current IT in engineering Web systems. This journal can also be a source for current and up-to-date processes, models, and techniques for the different types of Web-based systems such as integrating transactions processing and operations in inter- and intra-enterprise systems. The journal can promote previously well-studied critical concepts of integration with a more innovative and dynamic approach to seamless information services management in gross platform and gross organizational systems that are either Web-based or Web-enabled. In global economies, where the Web now plays such a major role in linking enterprises and governments together, this journal advocates a much needed strategy in research and applications that emphasizes exchange of data, information, and knowledge to foster intelligent and innovative cooperation. Moreover, the Journal advocates an increased efficiency in using and maintaining information and knowledge as a major resource and technology in organizations. The *International Journal of IT and Web Engineering* will lead to the development of systematic and disciplined approaches to data, information and knowledge capture, storage, access, and dissemination across and among different systems deployed separately over the Web and within and among all entities in information economies. The journal is timely for assisting, through new ideas, the prevention of many systems that are deployed over the Web from becoming problematic legacy systems.

— *Moataz Ahmed, LEROS Technologies Corporation and SONEX Enterprises Inc., USA*

Individual Articles

\$30.00 [Browse](#)

Purchase individual articles from this journal and receive a PDF download link upon order completion.



Full-text search over 55,000 research articles and chapters.

[All Subjects](#)

Related Journals

[Journal of Electronic Commerce in Organization...](#)
© 2003

[Information Technology Management](#)
© 1988

[International Journal of Game-Based Learning ...](#)
© 2011

[International Journal of Knowledge Management...](#)
© 2005

[International Journal of Digital Literacy and...](#)
© 2010

Digital Communications

There is growing interest in Web engineered systems and these are the basis of next generation enterprise IT systems with a grid architecture. This approach is being applied across commerce, entertainment, government, industry, education, and research. The new *International Journal of Information Technology and Web Engineering* should see many important papers in such areas as how distributed systems are transformed by Web technology. I look forward to publishing in this journal.

– Geoffrey Fox, Indiana University, USA

Indices

Top

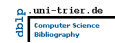
Bacon's Media Directory
Burrelle's Media Directory
[Cabell's Directories](#)
Compendex (Elsevier Engineering Index)
CSA Illumina
[DBLP](#)

Gale Directory of Publications & Broadcast Media
GetCited
[Google Scholar](#)

[INSPEC](#)
[JournalTOCs](#)
[MediaFinder](#)

[SCOPUS](#)

The Index of Information Systems Journals
The Standard Periodical Directory
Ulrich's Periodicals Directory



Topics Covered

Top

Topics to be discussed in this journal include (but are not limited to) the following:

- Case studies validating Web-based IT solutions
- Competitive/intelligent information systems
- Data analytics for business and government organizations
- Data and knowledge capture and quality issues
- Data and knowledge validation and verification
- Human factors and cultural impact of IT-based systems
- Information filtering and display adaptation techniques for wireless devices
- Integrated heterogeneous and homogeneous workflows and databases within and across organizations, suppliers, and customers
- Integrated user profile, provisioning, and context-based processing
- IT education and training
- IT readiness and technology transfer studies
- Knowledge structure, classification, and search algorithms or engines
- Metrics-based performance measurement of IT-based and Web-based organizations
- Mobile, location-aware, and ubiquitous computing
- Ontology and Semantic Web studies
- Quality of service and service level agreement issues among integrated systems
- Radio frequency identification (RFID) research and applications in Web engineered systems
- Security, integrity, privacy, and policy issues
- Software agent-based applications
- Strategies for linking business needs and IT

- Strategies for linking business needs and IT
- Virtual teams and virtual enterprises: communication, policies, operation, creativity, and innovation
- Web systems architectures, including distributed grid computers and communication systems processing
- Web systems engineering design
- Web systems performance engineering studies
- Web user interfaces design, development, and usability engineering studies

Editor(s)-in-Chief Biography

[Top](#)

Ghazi I. Alkhatib

Ghazi Alkhatib is an assistant professor of software engineering at the College of Computer Science and Information Technology, Applied Science University (Amman, Jordan). In 1984, he obtained his Doctor of Business Administration from Mississippi State University in information systems with minors in computer science and accounting. Since then, he has been engaged in teaching, consulting, training and research in the area of computer information systems in the US and gulf countries. In addition to his research interests in databases and systems analysis and design, he has published several articles and presented many papers in regional and international conferences on software processes, knowledge management, e-business, Web services and agent software, workflow and portal/grid computing integration with Web services.

Ernesto Damiani

Ernesto Damiani is a professor at the Dept. of Information Technology, University of Milan, where he leads the Software Architectures Lab. Prof. Damiani holds/has held visiting positions at several international institutions, including George Mason University (Fairfax, VA, US) and LaTrobe University (Melbourne, Australia). Prof. Damiani is an Adjunct Professor at the Sydney University of Technology (Australia). He has written several books and filed international patents; also, he co-authored more than two hundred research papers on advanced secure service-oriented architectures, open source software and business process design, software reuse and Web data semantics. Prof. Damiani is the Vice Chair of IFIP WG 2.12 on Web Data Semantics and the secretary of IFIP WG 2.13 on Open Source Software Development. He coordinates several research projects funded by the Italian Ministry of Research and by private companies including Siemens Mobile, Cisco Systems, ST Microelectronics, BT Exact, Engineering, Telecom Italy.

Editorial Board

[Top](#)

Editorial Board Coordinator

Zakaria Maamar, Zayed University, UAE

Publicity Coordinator

Wael Toghuj, Isra University, Jordan

Associate Editors

Michael Berger, Siemens Corporate Technology, Germany
 Walter Binder, EPFL, Switzerland
 Schahram Dustdar, Vienna University of Technology, Austria
 N.C. Narendra, IBM Software Labs, India
 Andres Iglesias Prieto, University of Cantabria, Spain
 David Taniar, Monash University, Australia

International Editorial Review Board

Jamal Bentahar, Concordia University, Canada
 Sara Comai, Politecnico di Milano, Italy
 Marie-Pierre Gleizes, IRIT - Université Paul Sabatier, France
 Yih-Feng Hwang, America On Line (AOL), USA
 Ali Jaoua, Qatar University, Qatar
 Leon Jololian, Zayed University, UAE
 Seok-won Lee, The University of North Carolina, USA
 Farid Meziane, The University of Salford, UK
 Soraya Kouadri Mostéfaoui, The British Open University, UK
 Michael Mrissa, Claude Bernard Lyon 1 University, France
 Manuel Nunez, Universidad Complutense de Madrid, Spain
 Quan Z. Sheng, Adelaide University, Australia
 Amund Tveit, Norwegian University of Science and Technology, Norway
 Leandro Krug Wives, Federal University of Rio Grande do Sul, Brazil
 Kok-Wai Wong, Murdoch University, Australia
 Hamdi Yahyaoui, King Fahd University of Petroleum and Minerals, Saudi Arabia

LEARN MORE:

[About IGI Global](#) | [Contact](#) | [Careers](#) | [Sitemap](#) | [FAQ](#)

RESOURCES FOR:

[Librarians](#) | [Authors/Editors](#) | [Distributors](#) | [Instructors](#) | [Translators](#)

MEDIA CENTER:

[Online Symposium](#) | [Blogs](#) | [Catalogs](#) | [Newsletters](#)

[Privacy Policy](#) | [Content Reuse Policy](#) | [Ethics and Malpractice](#)

IGI Global - All Rights Reserved



Scholar



Jesus M. Gonzalez-Barahona

Applying social network analysis techniques to community-driven libre software projects

[PDF] from flosshub.org

Authors Luis López-Fernández, Gregorio Robles, Jesus M Gonzalez-Barahona, Israel Herraiz
Publication date 2006
Journal International Journal of Information Technology and Web Engineering (IJITWE)
Volume 1
Issue 3
Pages 27-48
Publisher IGI Global

Description Abstract Source code management repositories of large, long-lived libre (free, open source) software projects can be a source of valuable data about the organizational structure, evolution, and knowledge exchange in the corresponding development communities. Unfortunately, the sheer volume of the available information renders it almost unusable without applying methodologies which highlight the relevant information for a given aspect of the project. Such methodology is proposed in this article, based on well known ...

Total citations [Cited by 56](#)



Scholar articles [Applying social network analysis techniques to community-driven libre software projects](#)
 L López-Fernández, G Robles, JM Gonzalez-Barahona... - International Journal of Information Technology and ..., 2006
[Cited by 56](#) - [Related articles](#) - [All 12 versions](#)

[Applying Social Network Analysis Techniques to Community-Driven Libre Software Projects ★](#)
 L LÁpez-FernÁ, G Robles, JM Gonzalez-Barahona... - International Journal of Information Technology and ..., 1993

Dates and citation counts are estimated and are determined automatically by a computer program.

[Help](#) [Privacy](#) [Terms](#) [Provide feedback](#) [My Citations](#)



Tools for the Study of the Usual Data Sources found in Libre Software Projects

Gregorio Robles, Universidad Rey Juan Carlos, Spain

Jesús M. González-Barahona, Universidad Rey Juan Carlos, Spain

Daniel Izquierdo-Cortazar, Universidad Rey Juan Carlos, Spain

Israel Herraiz, Universidad Rey Juan Carlos, Spain

ABSTRACT

Due to the open nature of Free/Libre/Open Source software projects, researchers have gained access to a rich set of development-related information. Although this information is publicly available on the Internet, obtaining and analyzing it in a convenient way is not an easy task and many considerations have to be taken into account. In this paper we present the most important data sources that can be found in libre software projects and that are studied by the research community: source code, source code management systems, mailing lists and bug tracking systems. We will give advice for the problems that can be found when retrieving and preparing the data sources for a posterior analysis, as well as provide information about the tools that support these tasks.

Keywords: bug tracking; mailing list; open source software; software metrics; software repository mining; source code management

INTRODUCTION

The fact that communication and organization are heavily tied in libre software¹ projects to the use of telematic means and that these interactions are, in general, stored and publicly offered over the Internet makes the number of data sources where development information can be extracted from grow beyond source code.

In addition, the ability of having memory (as data from activities in the past can be obtained) offers the possibility of performing longitudinal analysis as well. Research groups from all around the world have already taken benefit from the availability of such a rich amount of data sources in the last years. Nonetheless, the access, retrieval and fact extraction is by no means a simple task and many considerations

libre software projects (including Debian) and associations, writes in several media about topics related to libre software, and consults for companies and public administrations on issues related to their strategy on these topics. His research interests include understanding libre software development, where he has published several papers, and is participating in some international research projects. He is also one of the promoters of the idea of an European master program on libre software, and has specific interest in the education in that area.

Daniel Izquierdo-Cortazar is a PhD student at the Universidad Rey Juan Carlos in Móstoles, Spain. He earned a degree in computer science from the same university and obtained his master's degree in computer networks and computer science systems in 2006. His research work is centered in the assesment of libre software communities from an engineering point of view and especially with regard to quantitative and empirical issues. Right now he holds a grant from the Universidad Rey Juan Carlos to dedicate part of his time to his PhD thesis. He is also involved in European-funded projects such as QualOSS or FLOSS-World. He has also had the opportunity of attending to Wirtschaftsuniversität Wien (3 months in 2007) as a research visitor.

Israel Herraiz is a PhD student at the Universidad Rey Juan Carlos in Móstoles, Spain. Israel Herraiz holds a bachelor's degree in chemical engineering and master's degree in chemical and mechanical engineering from University of Cadiz (Spain). Right now he holds a grant from the Government of Madrid, to dedicate his full time to his PhD thesis, whose main topic is "Software Evolution of Large Libre Software Projects". In particular, he is using time series analysis and other statistical methods to characterize the evolution of software projects. He has participated in several research projects funded by the Framework Programme of the European Commision such as QualOSS or CALIBRE. He has also collaborated on other projects funded by companies such as Vodafone and Telefonica. He has participated in the writing of manuals about managing and starting libre software projects.

MEDIA



International Journal of Open Source Software and Processes (IJOSSP)

An Official Publication of the [Information Resources Management Association](#)

[Stefan Koch](#) (Bogazici University, Turkey)

Published Quarterly, Est. 2009.

Select a Format:

Institutions: Print

Subscription Year:

2013

\$595.00

Add to Cart

DOI: 10.4018/IJOSSP, ISSN: 1942-3926, EISSN: 1942-3934

Cite Journal Favorite

 Send

Me gusta

 Tweet



0

[Description](#) | [Contents](#) | [Mission](#) | [Indices](#) | [Topics Covered](#) | [Editor\(s\)-in-Chief Bio](#) | [Editorial Board](#)

Free Content

Sample Issue: [IJOSSP3\(3\)](#)

More Information

- [Search this Journal](#)
- [Call for Papers](#)
- [Guidelines for Submission](#)
- [Subscribe](#) 
- [Current Issue](#) 

Recommend

- [Send to a librarian](#)
- [Send to a colleague](#)

Available In

[InfoSci-Journals](#)

Browse Subjects

- [Business Technologies](#)
- [Database Technologies](#)
- [E-Government](#)
- [Educational Technologies](#)
- [Intelligent Technologies](#)
- [Knowledge Management](#)
- [Medical Technologies](#)
- [Multimedia Technologies](#)
- [Security Technologies](#)
- [Social Technologies](#)
- [Software & Engineering](#)

Description

[Top](#)

The **International Journal of Open Source Software and Processes (IJOSSP)** publishes high-quality peer-reviewed and original research articles on the large field of open source software and processes. This wide area entails many intriguing question and facets, including the special development process performed by a large number of geographically dispersed programmers, community issues like coordination and communication, motivations of the participants, and also economic and even legal issues. Even beyond this topic, open source software is an example of a highly distributed innovation process led by the users. Many aspects therefore have relevance even beyond the realm of software and its development. In this tradition, IJOSSP also publishes papers on these topics. IJOSSP is a multi-disciplinary outlet, and welcomes submissions from all relevant fields of research and applying a multitude of research approaches.

Journal Contents

[Top](#)

- [Volume 3: 4 Issues \(2011\)](#)
- [Volume 2: 4 Issues \(2010\)](#)
- [Volume 1: 4 Issues \(2009\)](#)

[View Complete Journal Contents Listing](#)

Mission

[Top](#)

The **International Journal of Open Source Software and Processes (IJOSSP)** publishes high-quality original research articles on the large field of open source software and processes. The primary mission is to enhance our understanding of this field and neighbouring areas by providing a focused outlet for rigorous research employing a multitude of approaches.

Indices

[Top](#)

- [Bacon's Media Directory](#)
- [Cabell's Directories](#)
- [Compendex \(Elsevier Engineering Index\)](#)
- [DBLP](#)

- [GetCited](#)
- [Google Scholar](#)

- [INSPEC](#)
- [JournalTOCs](#)
- [MediaFinder](#)

- [Norwegian Social Science Data Services \(NSD\)](#)
- [SCOPUS](#)

- [The Index of Information Systems Journals](#)
- [The Standard Periodical Directory](#)
- [Ulrich's Periodicals Directory](#)



Individual Articles

\$30.00 [Browse](#)

Purchase individual articles from this journal and receive a PDF download link upon order completion.



Full-text search over 55,000 research articles and chapters.

Full text search term(s)

All Subjects

Related Journals

 [International Journal of Information Systems in the Service Sector](#)
© 2009

 [International Journal of Robotics Application...](#)
© 2013

 [International Journal of Privacy and Health I...](#)
© 2013

 [International Journal of Adult Vocational Edu...](#)
© 2010

 [International Journal of Synthetic Emotions \(...\)](#)
© 2010

Topics Covered

[Top](#)

IJOSSP adopts an inclusive approach in its coverage. Therefore papers from software engineering, management, sociology, and other areas, as well as different research approaches are welcome. Possible

management, sociology, and other areas, as well as different research approaches are welcome. Possible topics include, but are not limited to, the following:

- Business models for open source and other community-created artifacts
- Case studies of open source projects, their participants and/or their development process
- Characteristics of open source software projects, products, and processes
- Communication and coordination in open source projects
- Customer co-creation and user participation in (software) design
- Economic analyses of open source
- Economics of a distributed innovation process
- Evolution of both open source software artifacts and open source communities
- Implications of open source software for functional areas like public administration or teaching
- Legal issues of open source software
- Motivation of participants in open source projects and other distributed development efforts
- Open science and open knowledge
- Open source adoption and quality
- Open source software development processes
- Usage and adoption of open source software in different application areas and/or countries
- User-centered innovation processes

Please note that despite the title, IJOSSP acknowledges, embraces, and covers other respective forms and definitions of similar nature, like free software or libre software. Therefore, each occurrence of open source should be read as free/libre/open source.

Editor(s)-in-Chief Biography

[Top](#)

Stefan Koch

Stefan Koch is Professor and Chair at Bogazici University, Department of Management. His research interests include user innovation, cost estimation for software projects, the open source development model, the evaluation of benefits from information systems and ERP systems. He has published over 20 papers in peer-reviewed journals, including *Information Systems Journal*, *Information Economics and Policy*, *Decision Support Systems*, *Empirical Software Engineering*, *Electronic Markets*, *Information Systems Management*, *Journal of Database Management*, *Journal of Software Maintenance and Evolution*, *Enterprise Information Systems* and *Wirtschaftsinformatik*, and over 30 in international conference proceedings and book collections. He has also edited a book titled *Free/Open Source Software Development* for an international publisher in 2004 and serves as Editor-in-Chief of the *International Journal on Open Source Software and Processes*.

Editorial Board

[Top](#)

Editorial Advisory Board

Paul David, Stanford University, USA and The University of Oxford, UK
Jesus Gonzalez-Barahona, Universidad Rey Juan Carlos, Spain
Brian Fitzgerald, University of Limerick, Ireland
Joachim Henkel, Technische Universitaet Muenchen, Germany
Karim Lakhani, Harvard Business School, USA
Eric von Hippel, MIT Sloan School of Management, USA
Georg von Krogh, ETH Zurich, Switzerland

Associate Editors

Jean-Michel Dalle, Universite Paris-Dauphine (Paris IX), France
Ernesto Damiani, University of Milan, Italy
Joe Feller, University College Cork, Ireland
Scott Hissam, Carnegie Mellon, USA
Greg Madey, University of Notre Dame, USA
Dirk Riehle, SAP Labs LLC, USA
Gregorio Robles, Universidad Rey Juan Carlos, Spain
Walt Scacchi, University of California - Irvine, USA
Sebastian Spaeth, ETH Zurich, Switzerland
Ioannis Stamelos, Aristotle University of Thessaloniki, Greece

Editorial Review Board

Ioannis Antoniadis, Aristotle University of Thessaloniki, Greece
Evangelia Berdou, University of Sussex, UK
Cornelia Boldyreff, University of Lincoln, UK
Andrea Capiluppi, University of Lincoln, UK
Carlo Daffara, Conecta Research, Italy
Marina Fiedler, Ludwig-Maximilians-Universitaet Munich, Germany
Daniel German, University of Victoria, Canada
Stefan Haefliger, ETH Zurich, Switzerland
Michael Hahsler, Southern Methodist University - Dallas, USA
Israel Herraiz, Universidad Rey Juan Carlos, Spain
Nicolas Jullien, TELECOM Bretagne, France
Sandeep Krishnamurthy, University of Washington, USA
George Kuk, Nottingham University Business School, UK
Jan Ljungberg, Gothenburg University, Sweden
Bjoern Lundell, University of Skovde, Sweden
Martin Michlmayr, Hewlett-Packard, Austria
Sandro Morasca, Universita degli Studi dell'Insubria, Italy
Gustaf Neumann, Vienna University of Economics and BA, Austria
Bulent Ozel, Istanbul Bilgi University, Turkey
Barbara Russo, Free University of Bolzano/Bozen, Italy
Kirk St. Amant, East Carolina University, USA
Sulayman K. Sowe, Aristotle University of Thessaloniki, Greece
Megan Squire, Elon University, USA

Brian Still, Texas Tech University, USA
Stefan Strecker, University Duisburg-Essen, Germany
Giancarlo Succi, Free University of Bolzano/Bozen, Italy

Frank van der Linden, Philips Medical Systems, The Netherlands
Andreas Wiebe, Vienna University of Economics and BA, Austria
Donald Wynn Jr., University of Dayton, USA

LEARN MORE:

[About IGI Global](#) | [Contact](#) | [Careers](#) | [Sitemap](#) | [FAQ](#)

RESOURCES FOR:

[Librarians](#) | [Authors/Editors](#) | [Distributors](#) | [Instructors](#) | [Translators](#)

[Privacy Policy](#) | [Content Reuse Policy](#) | [Ethics and Malpractice](#)

[IGI Global - All Rights Reserved](#)

MEDIA CENTER:

[Online Symposium](#) | [Blogs](#) | [Catalogs](#) | [Newsletters](#)

Web Images More... jgbarah@gmail.com

Google

Scholar

[←](#) [Edit](#) [🗑](#) [Export ▾](#)



Jesus M. Gonzalez-Barahona

Tools for the study of the usual data sources found in libre software projects

[\[PDF\] from herraiz.org](#)

Authors: Gregorio Robles, Jesús M González-Barahona, Daniel Izquierdo-Cortazar, Israel Herraiz
Publication date: 2009
Journal: International Journal of Open Source Software and Processes (IJOSSP)
Volume: 1
Issue: 1
Pages: 24-45
Publisher: IGI Global

Description: Abstract Due to the open nature of Free/Libre/Open Source software projects, researchers have gained access to a rich set of development-related information. Although this information is publicly available on the Internet, obtaining and analyzing it in a convenient way is not an easy task and many considerations have to be taken into account. In this paper we present the most important data sources that can be found in libre software projects and that are studied by the research community: source code, source code management ...

Total citations: [Cited by 41](#)



Scholar articles: [Tools for the study of the usual data sources found in libre software projects](#)
G Robles, JM González-Barahona... - International Journal of Open Source Software and ..., 2009
[Cited by 41](#) - [Related articles](#) - [All 9 versions](#)

Dates and citation counts are estimated and are determined automatically by a computer program.

[Help](#) [Privacy](#) [Terms](#) [Provide feedback](#) [My Citations](#)

1.A.3.
LIBROS Y CAPÍTULOS DE LIBROS

37. Hosting of Libre Software Projects: A Distributed Peer-to-Peer Approach*

Jesús M. González-Barahona and Pedro de-las-Heras-Quirós

Universidad Rey Juan Carlos, Grupo de Sistemas y Comunicaciones
Móstoles, Spain

{jgb, pheras}@gsyc.escet.urjc.es

Summary. While current hosting systems for libre software projects are mainly centralized or client-server systems, several benefits arise from using distributed peer-to-peer architectures. The peculiarities of libre software projects make them a perfect test-bed for experimenting with this architecture. Among those peculiarities we can mention: the clear need for decentralization, the highly distributed nature of the user base, the high reliability requirements and the rich set of interactions among users and developers. Here we present a first attempt to describe the characteristics and complexities of this application area, and the expected future developments.

37.1 Hosting Services for Libre Software Projects

Libre software¹ projects are unique in several ways. One of the most noticeable is the distributed nature of their developer and user base. Most libre software projects are managed by groups of geographically distributed developers who design, build and maintain the software. They use Internet, almost exclusively, for communication and coordination.

Traditionally, libre software projects set up and maintained their own tools and resources for the management of the software, including communication systems (mail list managers, IRC servers), information downloading systems (ftp and www servers), version control systems (usually CVS), bug tracking systems, etc.

During the last few years, the trend has been to move to centralized hosting services for libre software projects. They provide the same set of resources, and integrate them as much as possible. Some of them are really huge (for instance, SourceForge had in early 2002 more than 32,000 registered projects and well above 300,000 registered users, mostly developers [37.1]), and suffer the usual problems of centralized services: unique point of control and failure, bottlenecks, etc.

Due both to technical and ‘political’ reasons [37.2]) some groups have begun to explore how to make those systems more distributed using peer-to-peer models [37.3]), evolving towards a network of nodes providing hosting services. There are several advantages in this approach, and the model fits well with the common practices in the libre

* Work supported in part by CICYT grants No. TIC-98-1032-C03-03 and TIC2001-0447 and by Comunidad Autónoma de Madrid, grant No. 07T/0004/2001

¹ We use the term *libre software* to refer to software which complies with the usual definitions for *free software* and *open source software*.

References

- 37.1 SourceForge. SourceForge.Net update. 2002-01-22 edition, January 2002.
- 37.2 Loïc Dachary. SourceForge drifting, 2001.
<http://fsfeurope.org/news/article2001-10-20-01.en.html>.
- 37.3 Loïc Dachary. Savannah the next generation, September 2001.
<http://savannah.gnu.org/docs/savannah-plan.html>.
- 37.4 Clip2. The Gnutella protocol specification 0.4. Document revision 1.2.
<http://www.clip2.com/GnutellaProtocol04.pdf>.
- 37.5 Sean Rhea, Chris Wells, Patrick Eaton, Dennis Geels, Ben Zhao, Hakim Weatherspoon, and John Kubiawicz. Maintenance-free global data storage. *IEEE Internet Computing*, 5(5):40–49, September/October 2001.
- 37.6 Frank Dabek, M. Frans Kaashoek, David Karger, Robert Morris, and Ion Stoica. Wide-area cooperative storage with CFS. In *Proceedings of the 18th ACM Symposium on Operating Systems Principles*, Chateau Lake Louise, Banff, Canada, October 2001.
- 37.7 P. Druschel and A. Rowstron. PAST: A large-scale, persistent peer-to-peer storage utility. In *HotOS VIII*, Schloss Elmau, Germany, May 2001.
- 37.8 Brian Hayes. Collective wisdom. *American Scientist*, March-April 1998.
- 37.9 Eric Korpela, Dan Werthimer, David Anderson, Jeff Cobb, and Matt Lebofsky. SETI@Home: Massively distributed computing for SETI. *Computing in Science and Engineering*, 3(1), 2001.
- 37.10 J. Miller, P. Saint-Andre, and J. Barry. Jabber. Internet draft, February 2002.
<http://www.jabber.org/ietf/draft-miller-jabber-00.html>.
- 37.11 SourceForge. SourceForge services, 2001.
http://sourceforge.net/docman/display_doc.php?docid=753&group_id=1.
- 37.12 Ian Clarke, Scott G. Miller, Theodore W. Hong, Oskar Sandbergand, and Brandon Wiley. Protecting free expression online with Freenet. *IEEE Internet Computing*, pages 40–49, January-February 2002.
- 37.13 Thomas Lord. The arch revision control system, 2001.
<http://regexps.com/src/docs.d/arch/html/arch.html>.
- 37.14 Jonathan S. Shapiro and John Vanderburgh. CPCMS: A configuration management system based on cryptographic names. In *2002 USENIX Annual Technical Conference, FREENIX Track*, 2002.
- 37.15 Grant Bowman, MJ Ray, Loïc Dachary, and Michael Erdmann. A conceptual framework for the distribution of software projects over the Internet, 2001.
<http://home.snafu.de/boavista/coopx/coopx.html>.
- 37.16 Li Gong. JXTA: A network programming environment. *Internet Computing Online*, 5(3):88–95, May-June 2001.
- 37.17 Bernard Traversat, Mohamed Abdelaziz, Mike Duigou, Jean-Christophe Hugly, Eric Pouyoul, and Bill Yeager. Project JXTA virtual network, February 2002.
<http://www.jxta.org/docs/JXTAprotocols.pdf>.

Analyzing the anatomy of GNU/Linux distributions: methodology and case studies (Red Hat and Debian)

Jesús M. González-Barahona Gregorio Robles Miguel Ortuño-Pérez
Luis Rodero-Merino José Centeno-González Vicente Matellán-Olivera
Eva Castro-Barbero Pedro de-las-Heras-Quirós

August 2003

Legal Notice

Copyright (c) 2003 Jesús M. González-Barahona et al.

Contact author: jgb@gsync.escet.urjc.es

Abstract

GNU/Linux distributions are probably the largest coordinated pieces of software ever put together. Each one is in some sense a snapshot of a large fraction of the libre software development landscape at the time of the release, and therefore its study is important to understand the appearance of that landscape. They are also the working proof of the possibility of releasing reliable software systems in the range of 50-100 millions of lines of code, even when the components of such systems are built by hundreds of independent groups of developers, with no formal connection to the group releasing the whole system. In this chapter, we provide some quantitative information about the software included in two such distributions: Red Hat and Debian. Differences in policy and organization of both distributions will show up in the results, but some common patterns will also arise. For instance, both are doubling their size every two years, and both present similar patterns in programming language usage and package size distributions. All in all, this study pretends to show how GNU/Linux distributions are with respect to their source code, and how they evolve over time. A methodology of how to make comparable and automated studies on this kind of distributions is also presented.

1 Introduction

Libre software¹ provides software engineering with a unique opportunity: to make detailed characterizations of software projects which can be complete, detailed and reproducible, since the source code is available for anyone to read. This makes it possible to build complete models based on public and

¹Through this chapter, we use “libre software” as a way of referring both to free software and open source software. Though open source software and free software communities are very different, the software is not, since almost all licenses considered to be “free” are also considered “open source”, and the other way around.

- [Lucovsky2000] *From NT OS/2 to Windows 2000 and Beyond - A Software-Engineering Odyssey*, Mark Lucovsky, 4th USENIX Windows Systems Symposium, http://www.usenix.org/events/usenix-win2000/invitedtalks/lucovsky_html/ .
- [McGraw] *Building Secure Software: How to avoid security problems the right way*, Gary McGraw, Briefing cited by David A. Wheeler in <http://www.dwheeler.com/sloc/> .
- [Michlmayr2003] *Quality and the Reliance on Individuals in Free Software Projects*, Martin Michlmayr and Benjamin Mako Hill, <http://opensource.ucc.ie/icse2003/3rd-WS-on-OSS-Engineering.pdf> .
- [Mockus2002] *Two case studies of open source software development: Apache and Mozilla*, Audris Mockus, Roy T. Fielding, and James D. Herbsleb, <http://doi.acm.org/10.1145/567793.567795>.
- [RedHatNames] *The Truth Behind Red Hat Names*, Stephen John Smoogen, http://www.smoogespace.com/documents/behind_the_names.html .
- [Robles2001] *WIDI - Who Is Doing It? A research on Libre Software developers*, Gregorio Robles, Henrik Scheider, Ingo Tretkowski, and Niels Weber, <http://widi.berlios.de/paper/study.pdf> .
- [SLOCCount] *SLOCCount*, David A. Wheeler, <http://www.dwheeler.com/sloccount/> .
- [SLOCCountCh] *SLOCCount ChangeLog*, David A. Wheeler, <http://www.dwheeler.com/sloccount/ChangeLog> .
- [Scach2002] , Steve Scach, Bo Jin, David Wright, Gillian Z. Heller, and Jeff Offut, 2002, IEE Proceedings Journal: Special Issue on Open Source Software Engineering <http://opensource.ucc.ie/icse2002/SchachOffutt.pdf> .
- [Schneier2000] *Software Complexity and Security*, Bruce Schneier, March 15, 2000, Crypto-Gram Newsletter, <http://www.counterpane.com/crypto-gram-0003.html> .
- [SunPressRelease] *Sun Microsystems Announces Availability of StarOffice(TM) Source Code on OpenOffice.org*, SUN Microsystems, http://www.collab.net/news/press/2000/openoffice_live.html .
- [Wheeler2000] *Estimating Linux's Size*, David A. Wheeler, <http://www.dwheeler.com/sloc/> .
- [Wheeler2001] *More Than a Gigabuck: Estimating GNU/Linux's Size*, David A. Wheeler, <http://www.dwheeler.com/sloc> .
- [Young1999] *Giving It Away. How Red Hat Software Stumbled Across a New Economic Model and Helped Improve an Industry*, Robert Young, <http://www.oreilly.com/catalog/opensources/book/young.html> .

Web Images More... jgbarah@gmail.com

Google

Scholar ← Edit 🗑 Export ▾



Jesus M. Gonzalez-Barahona

Analyzing the anatomy of GNU/Linux distributions: methodology and case studies (Red Hat and Debian)

Authors Jesús M González-Barahona, Gregorio Robles, M Ortuno Pérez, Luis Rodero-Merino, J Centeno González, Vicente Matellan-Olivera, Eva Castro-Barbero, Pedro de-las Heras-Quirós

Publication date 2004

Journal Free/Open Source Software Development

Pages 27-58

Publisher Hershey, PA, USA: Idea Group Publishing

Description GNU/Linux distributions are probably the largest coordinated pieces of software ever put together. Each one is in some sense a snapshot of a large fraction of the libre software development landscape at the time of the release and, therefore, its study is important to understand the appearance of that landscape. They are also the working proof of the possibility of releasing reliable software systems in the range of 50-100 millions of lines of code, even when the components of such systems are built by hundreds of independent ...

Total citations [Cited by 26](#)



Scholar articles [Analyzing the anatomy of GNU/Linux distributions: methodology and case studies \(Red Hat and Debian\)](#)
JM González-Barahona, G Robles, MO Pérez... - Free/Open Source Software Development, 2004
[Cited by 26](#) - [Related articles](#) - [All 2 versions](#)

Dates and citation counts are estimated and are determined automatically by a computer program.

[Help](#) [Privacy](#) [Terms](#) [Provide feedback](#) [My Citations](#)

Open sources 2.0

Edited by Chris DiBona, Danese Cooper &
Mark Stone With a Foreword by Kim Polese



**THE CONTINUING
EVOLUTION**

O'REILLY®

Open Sources 2.0: The Continuing Evolution

Edited by Chris DiBona, Danese Cooper, and Mark Stone

Copyright © 2006 Chris DiBona, Danese Cooper, and Mark Stone. All rights reserved.
Printed in the United States of America.

Published by O'Reilly Media, Inc., 1005 Gravenstein Highway North, Sebastopol, CA 95472.

O'Reilly books may be purchased for educational, business, or sales promotional use. Online editions are also available for most titles (safari.oreilly.com). For more information, contact our corporate/institutional sales department: (800) 998-9938 or corporate@oreilly.com.

Executive Editor: Mike Hendrickson

Production Editor: Jamie Peppard

Cover Designer: Mike Kohnke

Interior Designer: Mike Kohnke

Printing History:

October 2005: First Edition.

The O'Reilly logo is a registered trademark of O'Reilly Media, Inc. *Open Sources 2.0* and related trade dress are trademarks of O'Reilly Media, Inc.

The essays in *Open Sources 2.0* are licensed under the Creative Commons Attribution-NonCommercial-NoDerivs 2.5 license. To view a copy of the license, send a letter to Creative Commons, 543 Howard Street, Fifth Floor, San Francisco, CA 94105, visit <http://creativecommons.org/licenses/by-nc-nd/2.5/>, or see Appendix B.

Many of the designations used by manufacturers and sellers to distinguish their products are claimed as trademarks. Where those designations appear in this book, and O'Reilly Media, Inc. was aware of a trademark claim, the designations have been printed in caps or initial caps.

While every precaution has been taken in the preparation of this book, the publisher, editors, and authors assume no responsibility for errors or omissions, or for damages resulting from the use of the information contained herein.



This book uses RepKover™, a durable and flexible lay-flat binding.

ISBN: 0-596-00802-3

[M]



Tab
Forew
Ackno
List of
Intro
S E C T

Open Sources 2.0: The Continuing Evolution

Edited by Chris DiBona, Danese Cooper, and Mark Stone

Copyright © 2006 Chris DiBona, Danese Cooper, and Mark Stone. All rights reserved.
Printed in the United States of America.

Published by O'Reilly Media, Inc., 1005 Gravenstein Highway North, Sebastopol, CA 95472.

O'Reilly books may be purchased for educational, business, or sales promotional use. Online editions are also available for most titles (safari.oreilly.com). For more information, contact our corporate/institutional sales department: (800) 998-9938 or corporate@oreilly.com.

Executive Editor: Mike Hendrickson

Production Editor: Jamie Peppard

Cover Designer: Mike Kohnke

Interior Designer: Mike Kohnke

Printing History:

October 2005: First Edition.

The O'Reilly logo is a registered trademark of O'Reilly Media, Inc. *Open Sources 2.0* and related trade dress are trademarks of O'Reilly Media, Inc.

The essays in *Open Sources 2.0* are licensed under the Creative Commons Attribution-NonCommercial-NoDerivs 2.5 license. To view a copy of the license, send a letter to Creative Commons, 543 Howard Street, Fifth Floor, San Francisco, CA 94105, visit <http://creativecommons.org/licenses/by-nc-nd/2.5/>, or see Appendix B.

Many of the designations used by manufacturers and sellers to distinguish their products are claimed as trademarks. Where those designations appear in this book, and O'Reilly Media, Inc. was aware of a trademark claim, the designations have been printed in caps or initial caps.

While every precaution has been taken in the preparation of this book, the publisher, editors, and authors assume no responsibility for errors or omissions, or for damages resulting from the use of the information contained herein.



This book uses RepKover™, a durable and flexible lay-flat binding.

ISBN: 0-596-00802-3

[M]

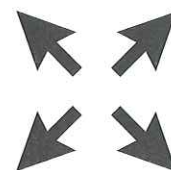


Table of Contents

| | |
|--|----------|
| Foreword: Source Is Everything | ix |
| <i>Kim Polese</i> | |
| Acknowledgments | xiii |
| List of Contributors | xv |
| Introduction | xxv |
| <i>Chris DiBona, Danese Cooper, and Mark Stone</i> | |
| SECTION 1. Open Source: Competition and Evolution | 1 |
| 1. The Mozilla Project: Past and Future | 3 |
| <i>Mitchell Baker</i> | |
| 2. Open Source and Proprietary Software Development | 21 |
| <i>Chris DiBona</i> | |
| 3. A Tale of Two Standards | 37 |
| <i>Jeremy Allison</i> | |
| 4. Open Source and Security | 57 |
| <i>Ben Laurie</i> | |
| 5. Dual Licensing | 71 |
| <i>Michael Olson</i> | |
| 6. Open Source and the Commoditization of Software | 91 |
| <i>Ian Murdock</i> | |

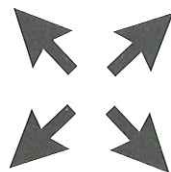
| | |
|--|-----|
| 7. Open Source and the Commodity Urge: Disruptive Models for a Disruptive Development Process. | 103 |
| <i>Matthew N. Asay</i> | |
| 8. Under the Hood: Open Source and Open Standards Business Models in Context. | 121 |
| <i>Stephen R. Walli</i> | |
| 9. Open Source and the Small Entrepreneur | 137 |
| <i>Russ Nelson</i> | |
| 10. Why Open Source Needs Copyright Politics | 149 |
| <i>Wendy Seltzer</i> | |
| 11. Libre Software in Europe. | 161 |
| <i>Jesus M. Gonzalez-Barahona</i> <i>Gregorio Robles</i> | |
| 12. OSS in India | 189 |
| <i>Alolita Sharma and</i> <i>Robert Adkins</i> | |
| 13. When China Dances with OSS. | 197 |
| <i>Boon-Lock Yeo, Louisa Liu, and Sunil Saxena</i> | |
| 14. How Much Freedom Do You Want? | 211 |
| <i>Bruno Souza</i> | |

SECTION 2. Beyond Open Source: Collaboration and Community. 229

| | |
|---|-----|
| 15. Making a New World | 231 |
| <i>Doc Searls</i> | |
| 16. The Open Source Paradigm Shift | 253 |
| <i>Tim O'Reilly</i> | |
| 17. Extending Open Source Principles Beyond Software Development. . . | 273 |
| <i>Pamela Jones</i> | |
| 18. Open Source Biology. | 281 |
| <i>Andrew Hessel</i> | |
| 19. Everything Is Known | 297 |
| <i>Eugene Kim</i> | |
| 20. The Early History of Nupedia and Wikipedia: A Memoir | 307 |
| <i>Larry Sanger</i> | |
| 21. Open Beyond Software | 339 |
| <i>Sonali K. Shah</i> | |
| 22. Patterns of Governance in Open Source | 361 |
| <i>Steven Weber</i> | |
| 23. Communicating Many to Many | 373 |
| <i>Jeff Bates and Mark Stone</i> | |

| | |
|---|------------|
| 7. Open Source and the Commodity Urge: Disruptive Models for a Disruptive Development Process | 103 |
| <i>Matthew N. Asay</i> | |
| 8. Under the Hood: Open Source and Open Standards Business Models in Context | 121 |
| <i>Stephen R. Walli</i> | |
| 9. Open Source and the Small Entrepreneur | 137 |
| <i>Russ Nelson</i> | |
| 10. Why Open Source Needs Copyright Politics | 149 |
| <i>Wendy Seltzer</i> | |
| 11. Libre Software in Europe | 161 |
| <i>Jesus M. Gonzalez-Barahona</i> <i>Gregorio Robles</i> | |
| 12. OSS in India | 189 |
| <i>Alolita Sharma and</i> <i>Robert Adkins</i> | |
| 13. When China Dances with OSS | 197 |
| <i>Boon-Lock Yeo, Louisa Liu, and Sunil Saxena</i> | |
| 14. How Much Freedom Do You Want? | 211 |
| <i>Bruno Souza</i> | |
| SECTION 2. Beyond Open Source: Collaboration and Community | 229 |
| 15. Making a New World | 231 |
| <i>Doc Searls</i> | |
| 16. The Open Source Paradigm Shift | 253 |
| <i>Tim O'Reilly</i> | |
| 17. Extending Open Source Principles Beyond Software Development | 273 |
| <i>Pamela Jones</i> | |
| 18. Open Source Biology | 281 |
| <i>Andrew Hessel</i> | |
| 19. Everything Is Known | 297 |
| <i>Eugene Kim</i> | |
| 20. The Early History of Nupedia and Wikipedia: A Memoir | 307 |
| <i>Larry Sanger</i> | |
| 21. Open Beyond Software | 339 |
| <i>Sonali K. Shah</i> | |
| 22. Patterns of Governance in Open Source | 361 |
| <i>Steven Weber</i> | |
| 23. Communicating Many to Many | 373 |
| <i>Jeff Bates and Mark Stone</i> | |

| | |
|--|------------|
| SECTION 3. Appendixes | 397 |
| A. The Open Source Definition | 399 |
| B. Referenced Open Source Licenses | 401 |
| C. Columns from Slashdot | 417 |
| Index | 423 |



CHAPTER 11

Jesus M. Gonzalez-Barahona
Gregorio Robles

Libre Software in Europe

The libre (free, open source) software¹ community is probably one of the more global and internationalized. Therefore, it may be a little artificial to try to separate the European share of it in the hope of finding peculiar characteristics. But at the same time, Europe is so diverse, so full of national, cultural, and linguistic boundaries, that it may be difficult to find common patterns in this already diverse libre software world. However, our feeling is that in between these two facts, there is plenty of room for writing about what is happening in the European libre software scene. While preparing the material for this chapter, we have come to the idea that, in fact, this is not such a global world, nor does its European fraction lack common patterns despite its diversity. With this focus in mind, we have looked for both the peculiarities and the commonalities. We have walked through the enormous amount of data concerning what is happening in the vibrant European libre software scene with the aim of offering the reader the more relevant and revealing trends and facts, providing a vision of a complex and diverse, but also uniform, landscape. And, for sure, a very personal one.

¹ In this chapter, we will use the term *libre software* to refer both to *free software* (according to the Free Software Foundation definition) and *open source software* (according to the Open Source Initiative), except where making distinctions makes sense. *Libre* is a term well understood in romance languages, such as Spanish, French, Portuguese, and Italian, and is understandable for speakers of many others. It lacks the ambiguity of “free” in English (“libre” means only “free” as in “free speech”) and is used by some people especially in Europe (although the term is rooted in the early U.S. free software community; see <http://sinetgy.org/jgb/articulos/libre-software-origin/libre-software-origin.html> for details). In this respect, it is important to notice that although the communities, the motivations, and the rationales behind “free” and “open source” are different, the software to which they refer is basically (although not exactly) the same.

whole new industry around it, and promoting not only companies, but also the individual developers who are making this a real possibility.

In case libre software provides real advantages in terms of innovation, competence, and social benefits, Europe is well placed for advancing in that direction. Are those opportunities not worth exploring? Can we risk losing our advantageous position in what could be the next revolution in the information society?

Open Sources 2.0

Robert Adkins

Jeremy Allison

Matt Asay

Mitchell Baker

Jeff Bates

Jesus M. Gonzalez-Barahona

Chris DiBona

Open Sources 2.0 collects thought-provoking essays from today's technology leaders, continuing the evolutionary picture developed in the 1999 book *Open Sources: Voices from the Revolution*.

Andrew Hessel

These essays explore open source's impact on the software industry and reveal how open source concepts are infiltrating other areas of commerce and society. The essays appeal to a broad audience. The software developer will find thoughtful reflections on practices and methodology from leading open source developers. The business executive will find analyses of business strategies from veteran open source business leaders. And those interested in an international perspective will find essays that describe the developing world's efforts to join the technology forefront and use open source to take control of their high tech destiny. For anyone with a strong interest in technology trends, this collection of essays is a must-read.

Pamela Jones

The enduring significance of open source goes well beyond high technology, however. Driving this new paradigm is network-enabled distributed collaboration. The growing impact of this model on all forms of online collaboration is fundamentally challenging our modern notion of community.

Eugene Kim

Ben Laurie

What does the future hold? Veteran open source commentators offer their perspectives, as do leading open source scholars. From Wikipedia to Slashdot, from computer technology to biotechnology, these essays reveal frontline views of functioning, flourishing, online collaborative communities.

Lawrence Lessig

Louisa Liu

The power of collaboration, enabled by the Internet and open source software, is changing the world in ways we can only begin to imagine. *Open Sources 2.0* further develops the evolutionary picture that emerged with the original *Open Sources* and expands on the transformative open source philosophy.

Ian Murdock

Doc Searls

Sunil Saxena

Larry Sanger

Gregorio Robles

Tim O'Reilly

Michael Olson

Russ Nelson

OREILLY®

www.oreilly.com

0-596-00802-3

US \$29.95

CAN \$41.95



Safari
BOOKS ONLINE
ENABLED

Includes
FREE 45-Day
Online Edition

Chapter I

Volunteers in Large Libre Software Projects: A Quantitative Analysis Over Time

Martin Michlmayr, University of Cambridge, UK

Gregorio Robles, Universidad Rey Juan Carlos, Spain

Jesus M. Gonzalez-Barahona, Universidad Rey Juan Carlos, Spain

Abstract

Most libre (free, open source) software projects rely on the work of volunteers. Therefore, attracting people who contribute their time and technical skills is of paramount importance, both in technical and economic terms. This reliance on volunteers leads to some fundamental management challenges: Volunteer contributions are inherently difficult to predict, plan, and manage, especially in the case of large projects. In this chapter we present an analysis of the evolution over time of the human resources in large libre software projects, using the Debian project, one of the largest and most complex libre software projects based mainly in voluntary work, as a case study. We have performed a quantitative investigation of data corresponding to roughly seven years, studying how volunteer involvement has affected the software released by the project, and the developer community itself.

Proceedings of the 1st International Conference on Open Source Systems, Genoa, Italy (pp. 100-107).

Endnotes

- ¹ In this chapter we will use the term “libre software” to refer to any software licensed under terms compliant with the Free Software Foundation definition of “free software,” and the Open Source Initiative definition of “open source software,” thus avoiding the controversy between those two terms.
- ² <http://popcon.debian.org>
- ³ All the code used has been released as libre software, and can be obtained from <http://libresoft.dat.escet.urjc.es/index.php?menu=Tools>
- ⁴ The Gini coefficient is a normalized measure of inequality; values near 0 point out equal distributions while values close to 1 are indicative for high inequalities.

Web Images More... jgbarah@gmail.com

Google

Scholar ← Edit 🗑 Export ▾



Jesus M. Gonzalez-Barahona

Volunteers in large libre software projects: A quantitative analysis over time

Authors: Martin Michlmayr, Gregorio Robles, Jesus M Gonzalez-Barahona
Publication date: 2007
Journal: Emerging Free and Open Source Software Practices
Pages: 1-24

Description: Abstract Most libre (free, open source) software projects rely on the work of volunteers. Therefore, attracting people who contribute their time and technical skills is of paramount importance, both in technical and economic terms. This reliance on volunteers leads to some fundamental management challenges: Volunteer contributions are inherently difficult to predict, plan, and manage, especially in the case of large projects. In this chapter we present an analysis of the evolution over time of the human resources in large libre ...

Total citations [Cited by 13](#)



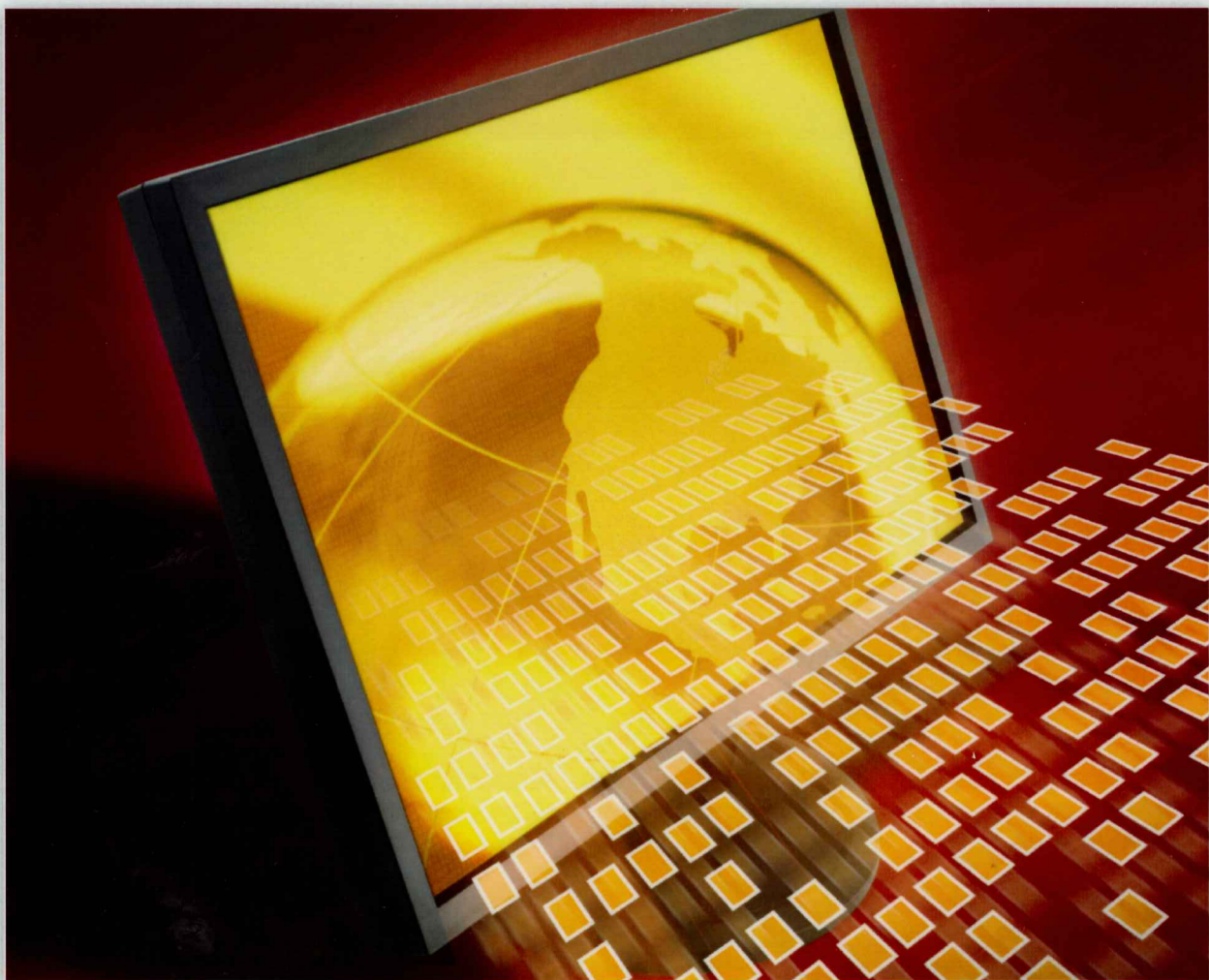
Scholar articles [Volunteers in large libre software projects: A quantitative analysis over time](#)
M Michlmayr, G Robles, JM Gonzalez-Barahona - Emerging Free and Open Source Software Practices, 2007
[Cited by 13](#) - [Related articles](#) - [All 2 versions](#)

Dates and citation counts are estimated and are determined automatically by a computer program.

[Help](#) [Privacy](#) [Terms](#) [Provide feedback](#) [My Citations](#)

PREMIER REFERENCE SOURCE

Multi-Disciplinary Advancement in Open Source Software and Processes



Stefan Koch

Senior Editorial Director: Kristin Klinger
Director of Book Publications: Julia Mosemann
Editorial Director: Lindsay Johnston
Acquisitions Editor: Erika Carter
Development Editor: Mike Killian
Production Coordinator: Jamie Snavelly
Typesetters: Keith Glazewski & Natalie Pronio
Cover Design: Nick Newcomer

Published in the United States of America by
Information Science Reference (an imprint of IGI Global)
701 E. Chocolate Avenue
Hershey PA 17033
Tel: 717-533-8845
Fax: 717-533-8661
E-mail: cust@igi-global.com
Web site: <http://www.igi-global.com>

Copyright © 2011 by IGI Global. All rights reserved. No part of this publication may be reproduced, stored or distributed in any form or by any means, electronic or mechanical, including photocopying, without written permission from the publisher. Product or company names used in this set are for identification purposes only. Inclusion of the names of the products or companies does not indicate a claim of ownership by IGI Global of the trademark or registered trademark.

Library of Congress Cataloging-in-Publication Data

Multi-disciplinary advancement in open source software and processes / Stefan Koch, editor.

p. cm.

Includes bibliographical references and index.

Summary: "This book reviews the development, design, and use of free and open source software, providing relevant topics of discussion for programmers, as well as researchers in human-computer studies, online and virtual collaboration, and e-learning"--Provided by publisher.

ISBN 978-1-60960-513-1 (hardcover) -- ISBN 978-1-60960-514-8 (ebook) 1. Open source software. 2. Computer software--Development. I. Koch, Stefan, 1974-

QA76.76.S46M97 2011

005.3--dc22

2011000067

British Cataloguing in Publication Data

A Cataloguing in Publication record for this book is available from the British Library.

All work contributed to this book is new, previously-unpublished material. The views expressed in this book are those of the authors, but not necessarily of the publisher.

Senior Editorial Director: Kristin Klinger
 Director of Book Publications: Julia Mosemann
 Editorial Director: Lindsay Johnston
 Acquisitions Editor: Erika Carter
 Development Editor: Mike Killian
 Production Coordinator: Jamie Snavely
 Typesetters: Keith Glazewski & Natalie Pronio
 Cover Design: Nick Newcomer

Published in the United States of America by
 Information Science Reference (an imprint of IGI Global)
 701 E. Chocolate Avenue
 Hershey PA 17033
 Tel: 717-533-8845
 Fax: 717-533-8661
 E-mail: cust@igi-global.com
 Web site: http://www.igi-global.com

Copyright © 2011 by IGI Global. All rights reserved. No part of this publication may be reproduced, stored or distributed in any form or by any means, electronic or mechanical, including photocopying, without written permission from the publisher. Product or company names used in this set are for identification purposes only. Inclusion of the names of the products or companies does not indicate a claim of ownership by IGI Global of the trademark or registered trademark.

Library of Congress Cataloging-in-Publication Data

Multi-disciplinary advancement in open source software and processes / Stefan Koch, editor.
 p. cm.

Includes bibliographical references and index.
 Summary: "This book reviews the development, design, and use of free and open source software, providing relevant topics of discussion for programmers, as well as researchers in human-computer studies, online and virtual collaboration, and e-learning"--Provided by publisher.

ISBN 978-1-60960-513-1 (hardcover) -- ISBN 978-1-60960-514-8 (ebook) 1. Open source software. 2. Computer software--Development. I. Koch, Stefan, 1974-
 QA76.76.S46M97 2011
 005.3--dc22

2011000067

British Cataloguing in Publication Data

A Cataloguing in Publication record for this book is available from the British Library.

All work contributed to this book is new, previously-unpublished material. The views expressed in this book are those of the authors, but not necessarily of the publisher.

Table of Contents

Preface xii

Section 1

Chapter 1
 Open Source Software Adoption: Anatomy of Success and Failure..... 1
Brian Fitzgerald, Lero – Irish Software Engineering Research Centre and University of Limerick, Ireland

Chapter 2
 Tools and Datasets for Mining Libre Software Repositories..... 24
Gregorio Robles, Universidad Rey Juan Carlos, Spain
Jesús González-Barahona, Universidad Rey Juan Carlos, Spain
Daniel Izquierdo-Cortazar, Universidad Rey Juan Carlos, Spain
Israel Herraiz, Universidad Alfonso X el Sabi, Spain

Chapter 3
 Integrating Projects from Multiple Open Source Code Forges..... 43
Megan Squire, Elon University, USA

Chapter 4
 Bridging the Gap between Agile and Free Software Approaches: The Impact of Sprinting..... 54
Paul J. Adams, Sirius Corporation Ltd., UK
Andrea Capiluppi, University of Lincoln, UK

Chapter 5
 Teaching Software Engineering with Free/Libre Open Source Projects 67
Ioannis Stamelos, Aristotle University of Thessaloniki, Greece

Chapter 2

Tools and Datasets for Mining Libre Software Repositories

Gregorio Robles

Universidad Rey Juan Carlos, Spain

Jesús González-Barahona

Universidad Rey Juan Carlos, Spain

Daniel Izquierdo-Cortazar

Universidad Rey Juan Carlos, Spain

Israel Herraiz

Universidad Alfonso X el Sabi, Spain

ABSTRACT

Thanks to the open nature of libre (free, open source) software projects, researchers have gained access to a rich set of data related to various aspects of software development. Although it is usually publicly available on the Internet, obtaining and analyzing the data in a convenient way is not an easy task, and many considerations have to be taken into account. In this chapter we introduce the most relevant data sources that can be found in libre software projects and that are commonly studied by scholars: source code releases, source code management systems, mailing lists and issue (bug) tracking systems. The chapter also provides some advice on the problems that can be found when retrieving and preparing the data sources for a later analysis, as well as information about the tools and datasets that support these tasks.

1. INTRODUCTION

In libre software¹ projects communication and organization are heavily dependent on the use of telematic means. Face-to-face communication is rare, and Internet-based tools are the most com-

mon means for a developer to interact with the code and with other developers.

Fortunately for researchers, the data produced by those interactions is usually stored and offered publicly over the Internet. The repositories for these data contain information valuable to understand the development process, and can be analyzed in combination with the most classical

data source: source code. In addition, the ability of having detailed information from the past (since it is usually archived for long periods of time) offers the possibility of performing also longitudinal and evolutionary analysis.

Research groups worldwide have already taken benefit from the availability of such a rich amount of data sources in the last years. Nonetheless, the access, retrieval and fact extraction is by no means a simple task and many considerations and details have to be taken into account to successfully retrieve and mine the data sources.

This chapter offers a detailed description of the most common data sources that can generally be found for libre software projects on the Internet, and of the data that can be found in them: **source code releases**, **source code management systems** (in the following, SCM), **mailing lists archives**, and **issue or bug tracking system** (in the following, BTS). In addition, we present some tools and datasets that might help researchers in their data retrieval and analysis tasks.

Mining and analyzing these data sources offer an ample amount of possibilities that surpass or complement other data-acquiring methodologies such as surveys, interviews or experiments. The amount of data that can be obtained, in a detailed way and in many cases for the whole lifetime of a software project, gives a precise description of the history of a project (Bauer and Pizka, 2003). In this sense, we have access to the activities (the what), the points in time (the when), the actors (the who) and sometimes even the reason (the why) (Hahsler and Koch, 2005). Compared to surveys, mining these data sources allow to access data for thousands of developers and a wide range of software projects. Most of these efforts can be considered as non-intrusive, as researchers can analyze the projects without interacting with developers, which is friendly to them. But even in a small environment, e.g., when evaluating the impact of software tools in a small team (Atkins et al., 2002), the use of data from one or more of these sources provides additional insight. Fur-

thermore, mining software repositories has many advantages compared to conducting experiments as real-world software projects are taken into consideration (Mockus and Votta, 2000, Graves and Mockus, 1998).

2. FIRST STEPS BEFORE THE ANALYSIS

There are some steps to be walked before the analysis of data from libre software projects can be started. First of all, the relevant data sources have to be identified. After that, the data has to be retrieved from the corresponding data repositories. Only then, the researcher can really start to analyze the data.

It is important to notice that there may be several ways of accessing the same kind of data, depending on the project and how it handles it. There are several different tools and systems that projects use, and they also have different usage conventions. For instance, the use of tags, comments, among others, may differ from one project to another, and can be of paramount importance to tell bugs apart from new feature requests in a BTS. The complexity and feasibility of both identification and retrieval depend, therefore, of the project. Figure 1 shows a diagram with all the steps that have to be accomplished for any source considered in the studies.

In general terms, the identification of the data source depends mostly on its significance for the software development of a project. Hence, identifying the source code releases, the SCM system, the mailing lists or the BTS is usually not problematic, since it lies in the interest of the projects to publish them clearly and openly. In these cases, the largest drawback may be is the lack of historical data. Sometimes a project only maintains a partial set of the data (eg, the most recent), and can even not maintain any set at all (eg, because a given tool is available but not used by developers). This situation is common for software re-

- ⁹ The ones shown next are the ones that can be found for the GNOME Bugzilla system. Bugzilla can be adapted and modified, so the fields may (and will) change from project to project.
- ¹⁰ <http://projects.libresoft.es/wiki/bicho>
- ¹¹ For instance, bug #55,000 from the KDE BTS, which can be accessed through the web interface at http://bugs.kde.org/show_bug.cgi?id=55000 may also be obtained in XML at following URL: <http://bugs.kde.org/xml.cgi?id=55000>.
- ¹² The location of the binaries may depend from system to system, although the standard location for them is the /usr/bin directory.
- ¹³ <http://tools.libresoft.es/cmtrics>
- ¹⁴ In the GNU coding standards, some conventions for change log files are given, see http://www.gnu.org/prep/standards/html_node/Change-Logs.html
- ¹⁵ *doceval* can be obtained from <https://forja.rediris.es/projects/csl-doceval/>.

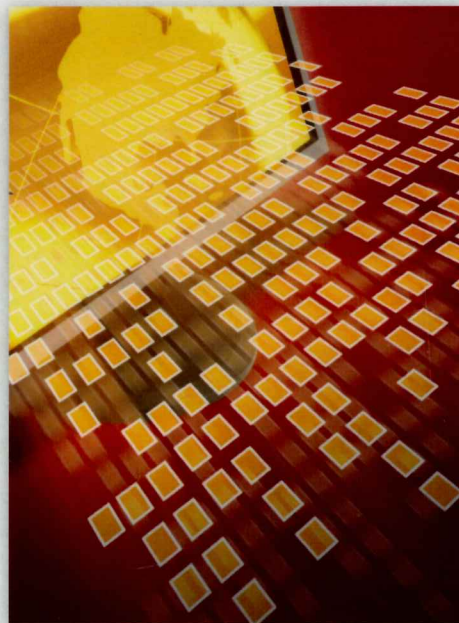
Multi-Disciplinary Advancement in Open Source Software and Processes

By its very nature, free and open source software encourages collaboration within and across virtual teams and promotes interdisciplinary methods and perspectives.

Multi-Disciplinary Advancement in Open Source Software and Processes reviews the development, design, and use of free and open source software, providing relevant topics of discussion for programmers, as well as researchers in human-computer studies, online and virtual collaboration, and e-learning. This reference explores successes and failures in the discipline, providing a foundation for future research and new approaches for the development and application of free and open source projects.

Topics Covered:

- Agile and Free Software Approaches
- Collaboration in Open Source
- Competition Between Open Source and Proprietary Software
- Floss Projects
- Multiple Open Source Code Forges
- Open Source Communities
- Open Source Software Adoption
- Open Source Software Adoption
- Software Engineering Education
- Unusual Data Sources



**Information Science
REFERENCE**

INFORMATION SCIENCE REFERENCE
701 E. Chocolate Avenue
Hershey, PA 17033, USA
www.igi-global.com

ISBN 978-1-60960-513-1



1.A.5. CONGRESOS

Studying the evolution of libre software projects using publicly available data

Gregorio Robles-Martínez, Jesús M. González-Barahona,
José Centeno-González, Vicente Matellán-Olivera, and Luis Roderó-Merino
GSyC, Universidad Rey Juan Carlos
{grex,jgb,jcenteno,vmo,lrodero}@gsync.es

Abstract

Libre software projects offer abundant information about themselves in publicly available storages (source code snapshots, CVS repositories, etc), which are a good source of quantitative data about the project itself, and the software it produces. The retrieval (and partially the analysis) of all those data can be automated, following a simple methodology aimed at characterizing the evolution of the project. Since the base information is public, and the tools used are libre and readily available, other groups can easily reproduce and review the results. Since the characterization offers some insight on the details of the project, it can be used as the basis for qualitative analysis (including correlations and comparative studies). In some cases, this methodology could also be used for proprietary software (although usually losing the benefits of peer review). This approach is shown, as an example, applied to MONO, a libre software project implementing parts of the .NET framework.

1 Introduction

Since its birth, software engineering has been trying to gain knowledge on the software development process in order to quantify the timing, human costs and technical resources that lead to a successful software development. Even so, in most cases experience has been acquired by studying in detail a handful of software projects that were accessible only to the researcher doing the study (due to intellectual propriety constraints). Important facts, like the detailed evolution of the source code or the developers working in the project at any given time were not publicly available for peer review. Quantitative characterization of those projects were, usually, rather incomplete, making it difficult to compare and correlate data.

However, since several years ago we have at our disposal a good quantity of data about thousands of libre (free,

open source) software projects. Most of them provide the researcher with very rich and complete information about their state at any given time¹. Being the number of projects and data about them so huge, it seems important to use a consistent and as much automated as possible methodology to go from the data available to the characterization. This should make the analysis of a large fraction of the available information possible, and simplify comparative analysis.

On the other hand, focusing on libre software engineering, there is little work on the quantitative characterization of libre software projects². To be able of getting some conclusions about libre software development, a lot of work on merely retrieving data about the projects has to be done. Fortunately, it can be performed on an uniform and semi-automated way.

From the different data available for libre software projects, we propose a methodology based on the analysis of the source code in CVS repositories, from three different points of view: source code size, interaction with the versioning system and authorship information. The combination of these approaches provide a detailed and complete picture of the project and its historical evolution.

2 Tools and methodology

The proposed methodology is based on the use of CVS³. Fortunately, there are a lot of projects in this situation: for

¹In addition to the source code itself, several systems are used which store historical and well structured information. The most relevant among them is CVS (and its derivatives), but there are others: bug tracking systems, mailing lists, release snapshots, etc.

²There are several well known papers trying to cast libre software projects from a more qualitative point of view (being [12] probably the most popular), but they have to face a lot of criticism due to the lack of quantitative data and methodology on which to base the analysis [1]

³Concurrent Version System, the versioning system most popular in the libre software community. Many libre software projects use a repository where all the source code and documentation are stored. Developers interact with the central repository by checking out the latest version, modifying it and committing back the changes. With every commit some information is stored in the CVS log files: the committer, the date, the lines that have

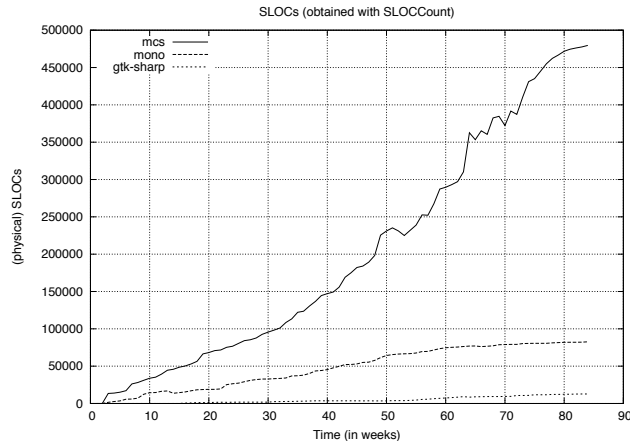


Figure 6. Source lines of code vs. time

4 Conclusions and further work

Libre software development permit analysis of unprecedented depth and detail for a fully reviewable and repeatable software engineering study [8]. The huge amount of information available for lots of libre software projects, with a great variety in size, programming language, programming tools, programming methods, etc. offers the possibility of creating a comparison framework from which knowledge and experience can be gained. The qualitative comparisons that are nowadays usually made could be completed with quantitative data taken at any given point of the life of a project. This allows for the having realistic pictures of the status of a project and its evolution, which should be very valuable to the managers and developers of the project. In addition, data crossing and comparisons between the different sources used in the described methodology gives software engineers a wider perspective of the studied projects.

Many work has still to be done to enhance the tools used to implement the methodology, and the methodology itself, in order to obtain more insight on the process of creating libre software. In this respect, although this paper gives an idea of a methodological approach, it is in fact very limited. Future research should include many other parameters that have not been taken into account yet. For instance, the use of complexity measures (McCabe [9] or Halstead [6]) is one of the considerations that are missing and should be introduced. From other point of view, more specific correlations should be studied so that characterization of a project from some points of view would permit the inference of other information more difficult to obtain.

References

- [1] N. Bezroukov. A second look at the cathedral and the bazar. *First Monday*, 1997. http://www.firstmonday.dk/issues/issue4_12/bezroukov/.
- [2] Codd. <http://codd.berlios.de/>.
- [3] R. A. Ghosh and V. V. Prakash. The orbiten free software survey, May 2000. http://www.firstmonday.dk/issues/issue5_7/ghosh/.
- [4] R. A. Ghosh, G. Robles, and R. Glott. Software source code survey (free/libre and open source software: Survey and study). Technical report, International Institute of Infonomics. University of Maastricht, The Netherlands, June 2002. <http://www.infonomics.nl/FLOSS/report>.
- [5] J. M. González-Barahona, M. A. Ortuño Pérez, P. de las Heras Quirós, J. Centeno González, and V. Matellán Olivera. Counting potatoes: The size of Debian 2.2. *Upgrade Magazine*, II(6):60–66, Dec. 2001. <http://people.debian.org/~jgb/debian-counting/counting-potatoes/>.
- [6] M. H. Halstead. *Elements of Software Science*. Elsevier, New York, USA, 1977.
- [7] K. Healy and A. Schussman. The ecology of open-source software development. Technical report, University of Arizona, USA, Jan. 2003. <http://opensource.mit.edu/papers/healyschussman.pdf>.
- [8] S. Koch and G. Schneider. Results from software engineering research into open source development projects using public data. *Diskussionspapiere zum Tätigkeitsfeld Informationsverarbeitung und Informationswirtschaft*, (22), 2000. <http://www.wi.wu-wien.ac.at/~koch/forschung/sw-eng/wp22.pdf>.
- [9] T. McCabe. A complexity measure. *IEEE Transactions on Software Engineering*, 1976.
- [10] Mono. <http://www.go-mono.com/>.
- [11] .NET developer framework. <http://msdn.microsoft.com/library/default.aspx?url=/nhp/default.asp?cont%entid=28000519>.
- [12] E. S. Raymond. The cathedral and the bazar. *First Monday*, 1997. http://www.firstmonday.dk/issues/issue3_3/raymond/.
- [13] Sloccount. <http://www.dwheeler.com/sloccount/>.
- [14] D. A. Wheeler. More than a gigabuck: Estimating gnu/linux's size, June 2001. <http://www.dwheeler.com/sloc/redhat71-v1/redhat71sloc.html>.

Web Images More... jgbarah@gmail.com

Google

Scholar ← Edit 🗑 Export ▾



Jesus M. Gonzalez-Barahona

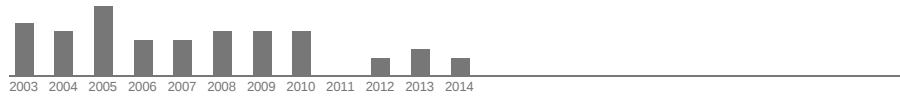
Studying the evolution of libre software projects using publicly available data

Authors [Gregorio Robles](#), [Jesús M González Barahona](#), [José Centeno González](#), [Vicente Matellán Olivera](#), [Luis Rodero Merino](#)

Publication date 2003/5/3

Journal [Proceedings of the 3rd Workshop on Open Source Software Engineering at the 25th International Conference on Software Engineering](#)

Total citations [Cited by 52](#)



Scholar articles [Studying the evolution of libre software projects using publicly available data](#)
G Robles, JM González Barahona... - Proceedings of the 3rd Workshop on Open Source ..., 2003
[Cited by 50](#) - [Related articles](#)

[Studying the evolution of libre software projects using publicly available data](#) ★
V Matellán Olivera - 2012
[Cited by 2](#) - [Related articles](#) - [All 7 versions](#)

Dates and citation counts are estimated and are determined automatically by a computer program.

[Help](#) [Privacy](#) [Terms](#) [Provide feedback](#) [My Citations](#)

Unmounting the “code gods” assumption

Jesús M. González-Barahona, Gregorio Robles-Martínez

GSyC, Universidad Rey Juan Carlos {jgb,grex}@gsync.escet.urjc.es

Abstract

The call for contributions of this workshop states: “The best known F/OSS projects appear to have succeeded because of a single person/small group who did the original development and supervised the subsequent evolution. Is it necessary to have a single strong leader or a small leadership team? What can be done to make F/OSS work without unusually-gifted leaders?” When reading it, our thought was not to deal with the questions, but to check the introduction: do libre software projects usually have those “code gods”? While trying to answer this latter question, we came out with a methodology that could be used to study the evolution of the core of developers of any project, given the information in its CVS repository.

1 The methodology

There are several assumptions about how libre software projects evolve. One of them is that a core of key developers guide the project, usually from its very beginning, and that their contributions remain of paramount importance during the life of the project. In some sense, that core team would “control” the evolution of the project. Of course, it is easy to find projects where this model seems not to be followed, but we wondered how one could tell from public data about a given project to which extent this “code gods” pattern was found in it.

Our approach has been to analyze the information in the CVS repository of the project, and more precisely, the log history of the CVS repository. For the purposes of analyzing the information, we split the duration of the project in several intervals of the same duration. For each interval, we identify the 20% of developers (rounded by excess) who have contributed with more commits during it (the “core” during that interval). We then study the evolution of the commits made by the different cores during the whole history of the project. It is important to notice that a given core can have developers in common with the core of any other interval (in general, they will, and in some cases, the core for several intervals can be exactly the same).

For visualizing the evolution of the project, we have plotted the resulting data in three different graphs: absolute number of commits by each core for each interval vs. time (absolute graph); aggregated number of commits by each core since the beginning of the project vs. time (aggregated graph); and fraction of the total commits during an interval done by each core for each interval vs. time (fractional graph). It is worth to mention that the aggregated graph is just the integral of the absolute graph, while the fractional graph is the absolute graph normalized by the number of commits during each period.

When looking at the resulting graphs, it is usually simple to decide whether the same core group rides the project from its beginning to current days.

2 Some case examples

- **A case with code gods: GnomeMeeting.** GnomeMeeting (a videoconference program, part of the GNOME project) is probably a canonical example of a project with “code gods”. It

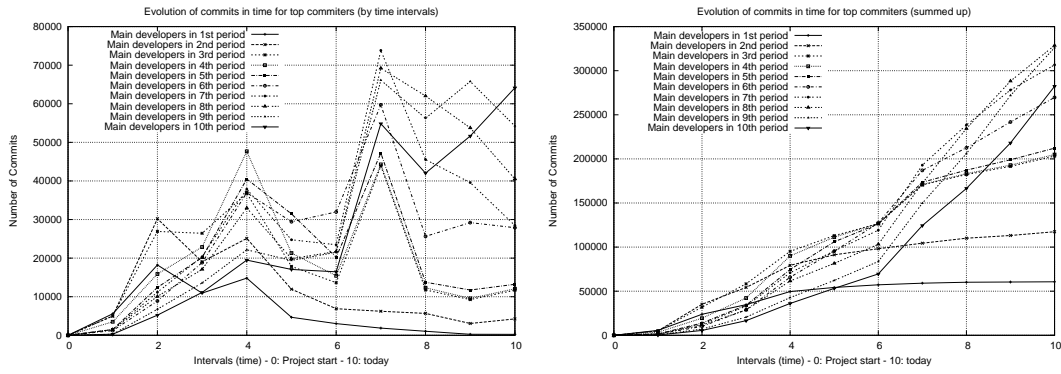


Figure 5: Absolute and aggregated graphs for the FreeBSD kernel

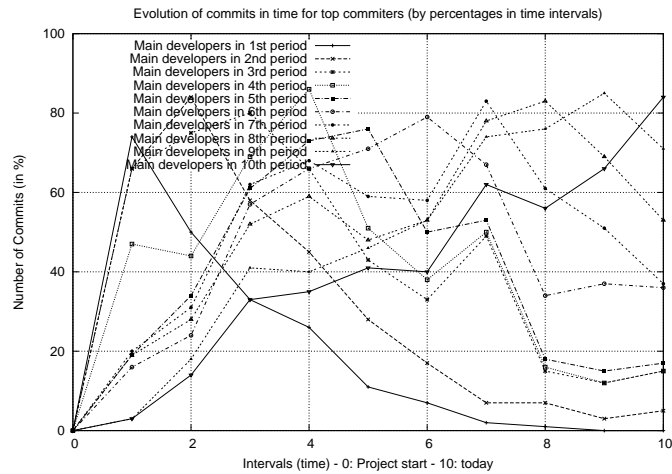


Figure 6: Fractional graph for the FreeBSD kernel

C Graphs for the BSD kernel

Data obtained from src/src directory of root module of the FreeBSD project CVS. First commit: 1993.03.21. Each interval: about 1 year. 528,601 commits in total.

Web Images More... jgbarah@gmail.com

Google

Scholar



Jesus M. Gonzalez-Barahona

Unmounting the code god assumption

Authors: Jesús M González Barahona, Gregorio Robles

Publication date: 2003

Journal: En: Proceedings of the Fourth International Conference on eXtreme Programming and Agile Processes in Software Engineering. Genoa, Italy

Total citations: [Cited by 21](#)



Scholar articles: [Unmounting the code god assumption](#)
JM González Barahona, G Robles - En: Proceedings of the Fourth International Conference ..., 2003
[Cited by 21 - Related articles](#)

Dates and citation counts are estimated and are determined automatically by a computer program.

[Help](#) [Privacy](#) [Terms](#) [Provide feedback](#) [My Citations](#)

Applying Social Network Analysis to the Information in CVS Repositories

Luis Lopez-Fernandez, Gregorio Robles, Jesus M. Gonzalez-Barahona
GSyC, Universidad Rey Juan Carlos
{llopez,grex,jgb}@gsync.es

Abstract

The huge quantities of data available in the CVS repositories of large, long-lived libre (free, open source) software projects, and the many interrelationships among those data offer opportunities for extracting large amounts of valuable information about their structure, evolution and internal processes. Unfortunately, the sheer volume of that information renders it almost unusable without applying methodologies which highlight the relevant information for a given aspect of the project. In this paper, we propose the use of a well known set of methodologies (social network analysis) for characterizing libre software projects, their evolution over time and their internal structure. In addition, we show how we have applied such methodologies to real cases, and extract some preliminary conclusions from that experience.

Keywords: source code repositories, visualization techniques, complex networks, libre software engineering

1 Introduction

The study and characterization of complex systems is an active research area, with many interesting open problems. Special attention has been paid recently to techniques based on network analysis, thanks to their power to capture some important characteristics and relationships. Network characterization is widely used in many scientific and technological disciplines, ranging from neurobiology [14] to computer networks [1] [3] or linguistics [9] (to mention just some examples). In this paper we apply this kind of analysis to software projects, using as a base the data available in their source code versioning repository (usually CVS). Fortunately, most large (both in code size and number of developers) libre (free, open source) software projects maintain such repositories, and grant public access to them.

The information in the CVS repositories of libre software projects has been gathered and analyzed using several methodologies [12] [5], but still many other approaches are possible. Among them, we explore here how to apply some

techniques already common in the traditional (social) network analysis. The proposed approach is based on considering either modules (usually CVS directories) or developers (committers to the CVS) as vertices, and the number of common commits as the weight of the link between any two vertices (see section 3 for a more detailed definition). This way, we end up with a weighted graph which captures some relationships between developers or modules, in which characteristics as information flow or communities can be studied.

There have been some other works analyzing social networks in the libre software world. [7] hypothesizes that the organization of libre software projects can be modeled as self-organizing social networks and shows that this seems to be true at least when studying SourceForge projects. [6] proposes also a sort of network analysis for libre software projects, but considering source dependencies between modules. Our approach explores how to apply those network analysis techniques in a more comprehensive and complete way. To expose it, we will start by introducing some basic concepts of social network analysis which are used later (section 2), and the definition of the networks we consider 3. In section 4 we introduce the characterization we propose for those networks, and later, in section 5, we show some examples of the application of that characterization to Apache, GNOME and KDE. To finish, we offer some conclusions and discuss some future work.

2 Basic concepts on Social Network Analysis

The Theory of Complex Networks is based on representing complex systems as graphs. There are many examples in the literature where this approach has been successfully used in very different scientific and technological disciplines, identifying vertices and links as relevant for each specific domain. For example, in ecological networks each vertex may represent a particular specie, with a link between two species if one of them “eats” the other. When dealing with social networks, we may identify vertices with persons or groups of people, considering a link when there is some kind of relationship between them.

Among the different kinds of networks that can be con-

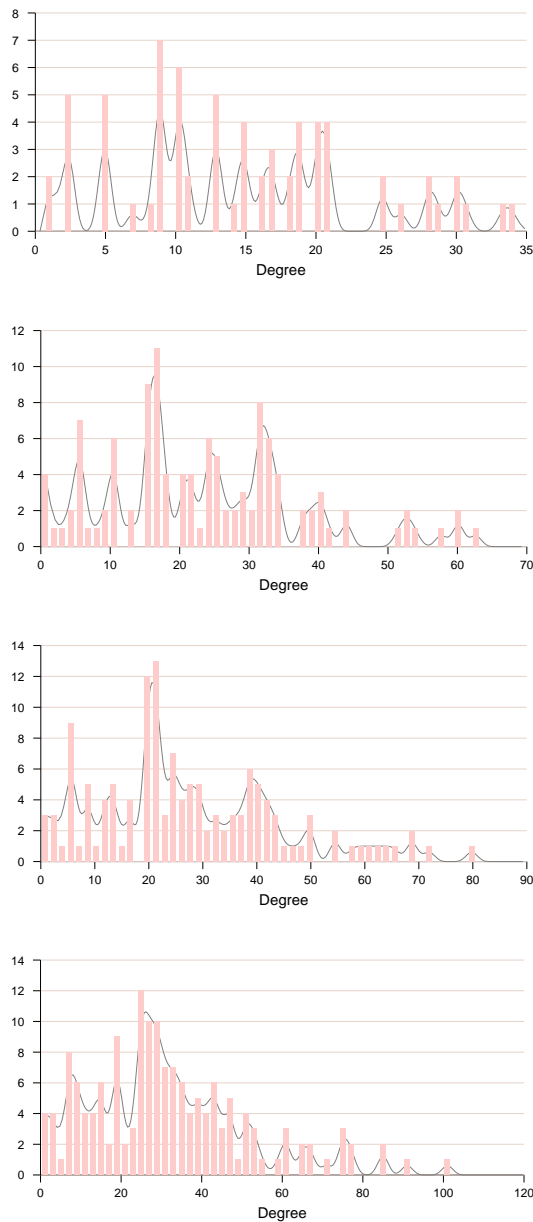


Figure 4. Connection degree of modules in Apache circa February from 2001 (top) to 2004 (bottom) (distribution)

usual in small-world and other social networks.

We feel that these research paths will allow for the more complete understanding of how libre software projects differentiate from each other, and also will help to identify common patterns and invariants.

References

- [1] R. Albert, A. L. Barabasi, H. Jeong, and G. Bianconi. Power-law distribution of the world wide web. *Science*, 287, 2000.
- [2] J. Anthonisse. The rush in a directed graph. Technical report, Stichting Mathematisch Centrum, Amsterdam, The Netherlands, 1971.
- [3] Cancho and R. Sole. The small world of human language. *Proceedings of the Royal Society of London. Series B, Biological Sciences*, 268:2261–2265, Nov. 2001.
- [4] C. Freeman. A set of measures of centrality based on betweenness. *Sociometry* 40, 35-41, 1977.
- [5] D. Germn and A. Mockus. Automating the measurement of open source projects. In *Proceedings of the 3rd Workshop on Open Source Software Engineering, 25th International Conference on Software Engineering*, Portland, Oregon, 2003.
- [6] R. A. Ghosh. Clustering and dependencies in free/open source software development: Methodology and tools. *First Monday*, 2003.
http://www.firstmonday.dk/issues/issue8_4/ghosh/index.html.
- [7] V. F. Greg Madey and R. Tynan. The open source development phenomenon: An analysis based on social network theory. In *Americas Conference on Information Systems (AMCIS2002)*, pages 1806–1813, Dallas, TX, USA, 2002.
http://www.nd.edu/~oss/Papers/amcis_oss.pdf.
- [8] S. Koch and G. Schneider. Effort, cooperation and coordination in an open source software project: Gnome. *Information Systems Journal*, 12(1):27–42, 2002.
- [9] R. Kumar, P. Raghavan, S. Rajagopalan, and A. Tomkins. The web and social networks. *IEEE Computer*, 35(11):32–36, 2002.
- [10] V. Latora and M. Marchiori. Economic small-world behavior in weighted networks. *Euro Physics Journal B* 32, 249-263, 2003.
- [11] A. Mockus, R. Fielding, and J. Herbsleb. A case study of open source software development: The Apache server. In *Proceedings of the 22nd International Conference on Software Engineering (ICSE 2000)*, pages 263–272, Limerick, Ireland, 2000.
- [12] G. Robles-Martinez, J. M. Gonzalez-Barahona, J. Centeno-Gonzalez, V. Matellan-Olivera, and L. Roderio-Merino. Studying the evolution of libre software projects using publicly available data. In *Proceedings of the 3rd Workshop on Open Source Software Engineering, 25th International Conference on Software Engineering*, pages 111–115, Portland, Oregon, 2003.
- [13] G. Sabidussi. The centrality index of a graph. *Psychometirka* 31, 581-606, 1996.
- [14] D. Watts and S. Strogatz. Collective dynamics of small-world networks. *Nature* 393, 440-442, 1998.

Web Images More... jgbarah@gmail.com

Google

Scholar ← Edit 🗑 Export ▾



Jesus M. Gonzalez-Barahona

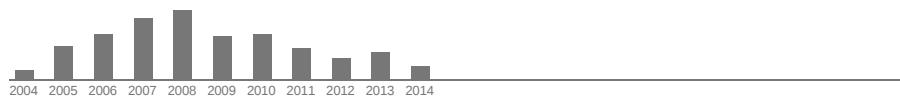
Applying social network analysis to the information in CVS repositories

[\[PDF\] from uwaterloo.ca](#)

Authors Luis Lopez-Fernandez, Gregorio Robles, Jesus M Gonzalez-Barahona
 Publication date 2004/5/25
 Journal International Workshop on Mining Software Repositories
 Pages 101-105

Description The huge quantities of data available in the CVS repositories of large, long-lived libre (free, open source) software projects, and the many interrelationships among those data offer opportunities for extracting large amounts of valuable information about their structure, evolution and internal processes. Unfortunately, the sheer volume of that information renders it almost unusable without applying methodologies which highlight the relevant information for a given aspect of the project. We propose the use of a well known set of ...

Total citations [Cited by 133](#)



Scholar articles [Applying social network analysis to the information in CVS repositories](#)
 L Lopez-Fernandez, G Robles, JM Gonzalez-Barahona - International Workshop on Mining Software ..., 2004
[Cited by 133](#) - [Related articles](#) - [All 22 versions](#)

Dates and citation counts are estimated and are determined automatically by a computer program.

[Help](#) [Privacy](#) [Terms](#) [Provide feedback](#) [My Citations](#)

GlueTheos: Automating the Retrieval and Analysis of Data from Publicly Available Software Repositories

Gregorio Robles
Universidad Rey Juan Carlos
grex@gsync.escet.urjc.es

Jesus M. González-Barahona
Universidad Rey Juan Carlos
jgb@gsync.escet.urjc.es

Rishab A. Ghosh
MERIT - Univ. Maastricht
rishab@merit.unimaas.nl

Abstract

For efficient, large scale data mining of publicly available information about libre (free, open source) software projects, automating the retrieval and analysis processes is a must. A system implementing such automation must have into account the many kinds of repositories with interesting information (each with its own structure and access methods), and the many kinds of analysis which can be applied to the retrieved data. In addition, such a system should be capable of interfacing and reusing as much existing software for both retrieving and analyzing data as possible.

As a proof of concept of how that system could be, we started sometime ago to implement the GlueTheos system, featuring a modular, flexible architecture which has been already used in several of our studies of libre software projects. In this paper we show its structure, how it can be used, and how it can be extended.

Keywords: Mining source code repositories, proposals for exchange formats, meta-models, and infrastructure tools, integration of mined data with other project data

1 Introduction

Libre software projects¹ range from very small ones (with just one developer committed to his own toy) to large-scale global projects with thousands of collaborating developers [9]. Specially, most of the larger projects follow a way of organization that has been called the ‘bazaar’-style development [14], open to everybody willing to participate. Thus, all elements taking part in the software development

¹In an attempt to avoid any confusion regarding the meaning of free in free software, throughout this article, the term libre software is used instead. It was chosen because of its meaning pointing towards liberation, and not of merely being costless. The term Open Source is refused for its ignorance about the philosophical foundations of what free software meant in the first place. “Libre software” is a term which is more and more usual in some communities, among them many European and Latin American countries.

process are as much open as possible, in the sense that the generated information is publicly available so that it is easier for ‘newcomers’ to become integrated in the project. Fortunately, this strategy offers to researchers the chance to access large amounts of data about the development process, the participants and, of course, the output product: the software.

Previous studies have taken advantage of this situation, and several research groups have focused their attention on the libre software phenomenon in the last years. For instance, [6] offers a software evolution analysis of the Linux kernel versions -without doubt the most known libre software project- following the classical software evolution point of view [11]. Others have paid attention to economic parameters [10] and have investigated how well classical software cost prediction models (as among others COCOMO [1]) can be applied. In [13] it is shown how libre software projects are composed usually of 10 to 15 core developers who lead the software process, a group of around one order of magnitude larger that participate in minor development tasks (bug fixes, etc.) and a final group around another order of magnitude that helps by other means (bug reports, etc.). In any case, the availability of data has proven to be very positive for research in the libre software environment.

But the amount of data and information available for inspection is that big that these analysis are often regarded as being too superficial. An example where this is common case are source code repositories. In such systems, not only the last state of the code is available for download but also all previous states. The amount of information that is ready for being extracted and analyzed is enormous and two factors become key points: automation and data mining.

When analyzing the data available in publicly accessible repositories, the automation of the data retrieval and the quantitative analysis is of great importance[15][3]. In the case of libre (free, open source) software projects, repositories are managed with very similar software (if not the

analysis of publicly available information about libre software projects. Currently, it can access CVS repositories and archives of some GNU/Linux distributions. By using external tools it can make several different analysis on the fetched data, and produce several kinds of reports (from tables with organized data to graphs or information suitable for being offered in a website. GlueTheos pretends to fill the gap that exists for in-depth, fully-automated analysis.

Our group is working currently in stabilizing the system, making it more versatile (including more downloading, analyzing and reporting modules), and exploring data formats for the exchange of information about libre software projects. We are planning also to put a big effort in the reporting modules, so that information from different sources can be integrated and correlated giving a wider picture than the one that a unique tool may offer. Special attention is being given in showing the huge amount of data in a way that it is comprehensible avoiding the problem of information overload that is common in these scenarios.


Future plans also include to set up an interactive website where libre software developers can request their projects to be analyzed. Developers would have only to fill out a form where the location of the publicly available data sources should be specified and the system will automatically retrieve and analyze them, putting up a web-sites with the results and finally notifying the developers that they can see results there.

All the GlueTheos system, and the external tools it uses, are libre (free, open source) software.

References

- [1] B. Boehm. *Software Engineering Economics*. Prentice Hall, 1981.
- [2] Codd website.
<http://codd.berlios.de/>.
- [3] D. Germán and A. Mockus. Automating the measurement of open source projects. In *Proceedings of the 3rd Workshop on Open Source Software Engineering, 25th International Conference on Software Engineering*, Portland, Oregon, 2003.
- [4] R. A. Ghosh. Clustering and dependencies in free/open source software development: Methodology and preliminary analysis. In *Open Source Workshop*, Toulouse, France, June 2002.
- [5] C. Gini. *On the Measure of Concentration with Espacial Reference to Income and Wealth*. Cowles Commission, 1936.
- [6] M. W. Godfrey and Q. Tu. Evolution in open source software: A case study. Oct. 2000.
- [7] J. M. González-Barahona, M. A. Ortuño Pérez, P. de las Heras Quirós, J. Centeno González, and V. Matellán Olivera. Counting potatoes: The size of Debian 2.2. *Upgrade Magazine*, II(6):60–66, Dec. 2001.
<http://people.debian.org/~jgb/debian-counting/counting-potatoes/>.
- [8] M. H. Halstead. *Elements of Software Science*. Elsevier, New York, USA, 1977.
- [9] K. Healy and A. Schussman. The ecology of open-source software development. Technical report, University of Arizona, USA, Jan. 2003.
<http://opensource.mit.edu/papers/healyschussman.pdf>.
- [10] S. Koch and G. Schneider. Results from software engineering research into open source development projects using public data. *Diskussionspapiere zum Tätigkeitsfeld Informationsverarbeitung und Informationswirtschaft*, (22), 2000.
<http://www.wi.wu-wien.ac.at/~koch/forschung/sw-eng/wp22.pdf>.
- [11] M. Lehman, J. Ramil, P. Wernick, and D. Perry. Metrics and laws of software evolution - the nineties view. 1997.
- [12] T. McCabe. A complexity measure. *IEEE Transactions on Software Engineering*, 1976.
- [13] A. Mockus, R. Fielding, and J. Herbsleb. A case study of open source software development: The Apache server. In *Proceedings of the 22nd International Conference on Software Engineering (ICSE 2000)*, pages 263–272, Limerick, Ireland, 2000.
- [14] E. S. Raymond. The cathedral and the bazaar. *First Monday*, 1997.
http://www.firstmonday.dk/issues/issue3_3/raymond/.
- [15] G. Robles-Martinez, 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, 25th International Conference on Software Engineering*, pages 111–115, Portland, Oregon, 2003.
- [16] Sloccount.
<http://www.dwheeler.com/sloccount/>.
- [17] D. A. Wheeler. More than a gigabuck: Estimating gnu/linux's size, June 2001.
<http://www.dwheeler.com/sloc/redhat71-v1/redhat71sloc.html>.

Web Images More... jgbarah@gmail.com



Scholar



Jesus M. Gonzalez-Barahona

Gluetheos: Automating the retrieval and analysis of data from publicly available software repositories [\[PDF\] from urjc.es](#)

Authors Gregorio Robles, Jesus M González-Barahona, Rishab A Ghosh
Publication date 2004/5
Journal Proceedings of the International Workshop on Mining Software Repositories
Pages 28-31
Description For efficient, large scale data mining of publicly available information about libre (free, open source) software projects, automating the retrieval and analysis processes is a must. A system implementing such automation must have into account the many kinds of repositories with interesting information (each with its own structure and access methods), and the many kinds of analysis which can be applied to the retrieved data. In addition, such a system should be capable of interfacing and reusing as much existing software for both ...

Total citations [Cited by 31](#)



Scholar articles [Gluetheos: Automating the retrieval and analysis of data from publicly available software repositories](#)
 G Robles, JM González-Barahona, RA Ghosh - Proceedings of the International Workshop on Mining ..., 2004
[Cited by 31](#) - [Related articles](#) - [All 15 versions](#)

Dates and citation counts are estimated and are determined automatically by a computer program.

[Help](#)
[Privacy](#)
[Terms](#)
[Provide feedback](#)
[My Citations](#)

Community structure of modules in the Apache project

Jesús M. González-Barahona, Luis López, Gregorio Robles
Grupo de Sistemas y Comunicaciones – Universidad Rey Juan Carlos
{jgb,llopez,grex}@gsyc.esct.urjc.es

Abstract

The relationships among modules in a software project of a certain size can give us much information about its internal organization and a way to control and monitor development activities and evolution of large libre software projects. In this paper, we show how information available in CVS repositories can be used to study the structure of the modules in a project when they are related by the people working in them, and how techniques taken from the social networks fields can be used to highlight the characteristics of that structure. As a case example, we also show some results of applying this methodology to the Apache project in several points in time. Among other facts, it is shown how the project evolves and is self-structuring, with developer communities of modules corresponding to semantically related families of modules.

1. Introduction

Large libre software¹ projects are usually organized as a set of modules. Each one of them can correspond to a given program, library, or any other unit identified by the project as distinct. It is common that developers work in several modules, according to their interests, skills, and constraints, and that those modules to which they contribute change over time. The relationships among modules due to the people working in them constitutes a sort of social structure of the project. In some sense, those human relationships are the glue that maintain the whole project together, and the chains that contribute to spread information and uses from one part of the project to others. In this paper, we explore how those developer connections contribute to the making of the community structure of the project, how it can be identified and visualized and how it evolves over time.

The study and characterization of complex systems is a very fruitful research area nowadays with many interesting open problems. Special attention has been paid recently to complex networks, where graph and network analysis

plays an important role and is gaining great popularity due to its intrinsic power to reduce a particular system to its simple components and relationships. Thanks to this, network characterization is widely used in many scientific and technological disciplines as neurobiology [1], computer networks [2][3], linguistics [4], etc.

Among complex networks, social networks appear in a quite natural way as a method for analyzing the structure and interactions of people and groups of people within complex organizations [5][6][7][8][9]. To understand the structure of those networks, we are interested in determining how the different nodes interact and form groups that, in turn, interact with each other giving rise to higher order groups. The set of groups obtained, as well as their relationships, is which we call the community structure of the network.

All the information we need for such an analysis is available in the CVS repository of the project. Using it, we construct the network of modules at a given time, considering a link when there is a common set of developers that have contributed to both modules. Later, we apply on it some techniques from the social networks field.

There have been some other works analyzing social networks in the libre software world. [10] hypothesizes that the organization of libre software projects can be modeled as self-organizing social networks and shows that this seems to be true at least when studying SourceForge projects. [11] proposes also a sort of network analysis for libre software projects, but taking into account this time a technical connection between modules as code dependency is.

How libre software projects organize themselves in subprojects is an issue that has been already discussed in literature. For instance, in [12] it is argued that projects that require a number of core developers that is larger than a given amount (10 to 15 persons) create in effect several related projects if no other means of code control are introduced. In the conclusions of this paper we will discuss the validity of this hypothesis.

In the rest of this paper we detail the methodology we are using to build the network of modules, and how we identify communities in that network. Later, we show some results of applying this methodology to the Apache

¹ Through this position paper we mean by libre software any program which is either open source software (according to the Open Source Initiative definition) or free software (according to the Free Software Foundation definition).

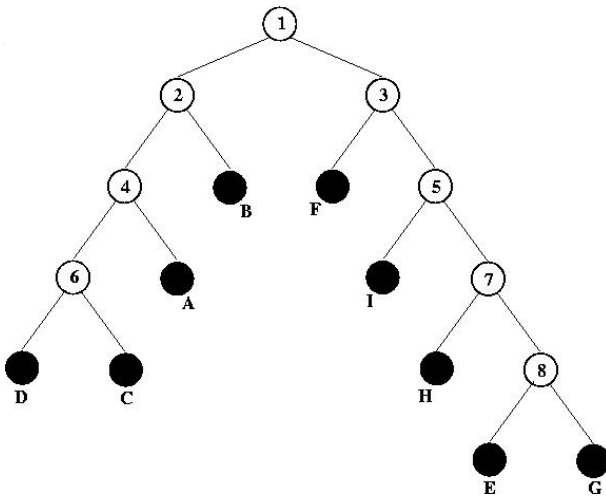


Illustration 9 Example of GN algorithm: output network

The GN algorithm is based on a parameter called the betweenness of edges, which measures the number of shortest paths connecting pairs of nodes which go through that edge [5]. Edges connecting high clustered communities must have higher betweenness. So, the GN algorithm proceeds by calculating the betweenness of all edges and eliminating the edge with the highest betweenness. These steps are repeated until the network is split into two connected components. After that, the process is recursively executed on each of the two separate components. To represent the splitting process, we use a binary tree which is built in the following way: each time a split is carried out, we add a virtual node in the tree (marked with numbers in Figure A-2) which is connected to the two novel components. When these components are single vertices of the original network, we add them as terminal nodes of the tree (marked with letters in Figure A-2).

10. References

- [1] D.J. Watts and S.H. Strogatz, Collective dynamics of small-world networks, *Nature* 393, 440-442, 1998 .
- [2] R. Albert, A.-L. Barabasi, H. Jeong, and G. Bianconi, Power-law distribution of the World Wide Web, *Science* 287 2115a (2000).
- [3] R.F. Cancho and R. Sole, The small world of human language, *Proc. R. Soc.* 268, 2261-2265, 2001.
- [4] R. Kumar, P. Raghavan, S. Rajagopalan, and A. Tomkins, The Web and social networks, *IEEE Computer* 35(11) 32-36, 2002.
- [5] M.E.J. Newman, Scientific collaboration networks: I. Network construction and fundamental results, *Phys. Rev. E* 64, 016131 (2001).
- [6] M.E.J. Newman, Scientific collaboration networks: II. Shortest paths, weighted networks, and centrality, *Phys. Rev. E* 64, 016132, 2001.
- [7] R. Guimera, A. Daz-Aguilera, F. Vega-Redondo, A. Cabrales, and A. Arenas, Optimal network topologies for local search with congestion, *Phys. Rev. Let.* 89, 248701, 2002.
- [8] L. Lopez and M.A.F. Sanjuan, Relation between structure and size in social networks, *Phys. Rev. E.* 65, 036107, 2002.
- [9] L. Lopez, J.F. Mendes, and M.A.F. Sanjuan, Hierarchical social networks and information flow, *Physica A*, 316, 591-604, 2002.
- [10] Greg Madey, V. Freeh, and R. Tynan, The Open Source Software Development Phenomenon: An Analysis Based on Social Network Theory, 2002, http://www.nd.edu/~oss/Papers/amcis_oss.pdf.
- [11] Rishab Aiyer Ghosh, Clustering and dependencies in free/open source software development: Methodology and tools, April 2003, http://www.firstmonday.dk/issues/issue8_4/ghosh/index.html.
- [12] Audris Mockus, Roy T. Fielding, and James D. Herbsleb, Two Case Studies of Open Source Software Development: Apache and Mozilla, <http://www.research.avayalabs.com/techreport/ALR-2002-003-paper.pdf>.
- [13] Gregorio Robles, Jesús González-Barahona, José Centeno-González, Vicente Matellán-Olivera, and Luis Rodero-Merino. <http://opensource.ucc.ie/icse2003/3rd-WS-on-OSS-Engineering.pdf>.
- [14] M. Girvan and M. E. J. Newman, Community structure in social and biological networks, *Proc. Natl. Acad. Sci. USA* 99, 7821-7826, 2002.
- [15] M.E.J. Newman, Detecting community structure in networks., *Eur. Phys. J. B*, in press. <http://www.santafe.edu/~mark/pubs.html>.
- [16] M.E.J. Newman and M. Girvan, Finding and evaluating community structure in networks., *Phys. Rev. E*, in press. <http://www.santafe.edu/~mark/pubs.html>.
- [17] R. Guimera, L. Danon, A. Daz-Aguilera, F. Giralt, and A. Arenas, Self-similar community structure in organizations, <http://arxiv.org/pdf/cond-mat/0211498>.

Web Images More... jgbarah@gmail.com

Google

Scholar ← Edit 🗑 Export ▾



Jesus M. Gonzalez-Barahona

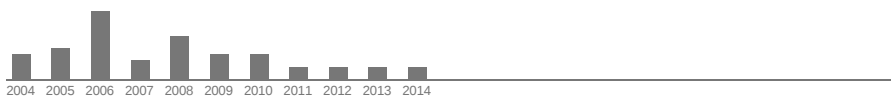
Community structure of modules in the apache project

[\[PDF\]](#) from ifipwg213.org

Authors: Jesús M González-Barahona, Luis López, Gregorio Robles
Publication date: 2004/5/25
Journal: Proceedings of the 4th Workshop on Open Source Software Engineering, 26th International Conference on Software Engineering, Edinburgh, Scotland, UK

Description: The relationships among modules in a software project of a certain size can give us much information about its internal organization and a way to control and monitor development activities and evolution of large libre software projects. In this paper, we show how information available in CVS repositories can be used to study the structure of the modules in a project when they are related by the people working in them, and how techniques taken from the social networks fields can be used to highlight the characteristics of that structure. ...

Total citations [Cited by 49](#)



Scholar articles [Community structure of modules in the apache project](#)
JM González-Barahona, L López, G Robles - Proceedings of the 4th Workshop on Open Source ..., 2004
[Cited by 49](#) - [Related articles](#) - [All 9 versions](#)

Dates and citation counts are estimated and are determined automatically by a computer program.

[Help](#) [Privacy](#) [Terms](#) [Provide feedback](#) [My Citations](#)

Remote analysis and measurement of libre software systems by means of the CVSanaly tool

Gregorio Robles
Universidad Rey Juan Carlos
grex@gsync.esct.urjc.es

Stefan Koch
Wirtschaftsuniversität Wien
stefan.koch@wu-wien.ac.at

Jesús M. González-Barahona
Universidad Rey Juan Carlos
jgb@gsync.esct.urjc.es

Abstract

Libre (free, open source) software is one of the paradigmatic cases where heavy use of telematic tools and user-driven software development are key points. This paper proposes a methodology for measuring and analyzing remotely big libre software projects using publicly-available data from their version control repositories. By means of a tool called CVSanaly that has been implemented following this methodology, measurements and analyses can be made in an automatic and non-intrusive way, providing real-time and historical data about the project and its contributors.

Keywords: Mining source code repositories, empirical software engineering, libre software engineering

1 Introduction

The way software is produced has changed radically in the last two decades. Among other circumstances, the arise of the Internet has brought a change in software production paradigms and an increase in the number of end users. That way, it is not uncommon that software development is done with a distributed team and that fast user feedback is possible, both things by means of telematic tools. Also, versions of the software are released often in order to gain momentum from user's feedback.

One of the software production fields where all the aforementioned characteristics are given is the libre (free/open source) software world. In that area there exists a big synergy between developers and users to the point that sometimes it is often not possible to distinguish these groups. In fact, many (if not all) developers are really power-users that have the required programming capabilities to find a solution to their software needs [15]. Hence, the study of libre software can be seen as a paradigmatic case of a production environment where users have a big implication, even leading the development.

Due to the distributed nature of this development paradigm, cooperation and communication needs to be

achieved using telematic tools. Data from these sources allows for an almost-automatic remote analysis and measurement of both software product and underlying processes. In fact, remote analysis is necessary, as a central authority or location like office is absent, preventing on-site evaluations. In addition, all interested parties including users and developers are distributed, and therefore need to perform and also access all measurements and analyses remotely.

Libre software projects go from very small ones with one developer committed to his program to large-scale global projects where thousands of developers interact [10]. Especially most of the bigger projects follow a way of organization that has been called the "bazaar"-style development [15] whose aim is to be as near to the end user as possible, giving users even a co-developer status. In [14] it is shown that such big libre software projects are composed mostly of 10 to 15 core developers who lead the software process, a group of around one order of magnitude larger that participate in minor development tasks (bug fixes, etc.) and a final group around another order of magnitude that helps by other means (bug reports, etc.).

Several research groups have focused their attention to the libre software phenomenon in the last years, so that several views of this paradigm can be found. For instance, [7] offers a software evolution analysis of the Linux kernel - without doubt the most known libre software project- following the classical software evolution point of view [13]. Others have paid attention to economic parameters [12] and have investigated how well classical software cost prediction models as among others [2] can be applied.

This paper presents an empirical analysis of libre software projects that can be made automatically, non-intrusively and remotely from public-available data. The source of the data that is measured and afterwards analyzed is taken from the source versioning systems that most libre software projects use, the CVS (Concurrent Versions System). The CVS contains the current state of the source code as well as all the previous versions of the code. It serves as a basis for developer interaction and group work.

Further possible publicly available data sources would

different and total numbers of modules they are active, the most common filetypes, and of course measures for their participation like commits or changed LOCs. Table 5 shows briefly what kind of data we are able to get from a commiter. Notice that committers have to have write permission into CVS, so that the analysis of commits in CVS may differ from the one of usual changelogs [3].

Table 5. Statistics for commiter 'acs'

| Module | Commits | LOC | First | Last |
|--------------|---------|------|----------|----------|
| mrproject | 181 | 5402 | 02.03.22 | 02.07.31 |
| libmrproject | 39 | 496 | 02.03.24 | 02.07.09 |

6 Conclusions and further work

As this paper and the presented implementation show, insights into both the current state and the evolution of libre software systems can indeed be gained on a remote basis, even without personal involvement in a project. The information that can be gathered from publicly-available version control systems allows us to have a global perspective of the project and the human resources committed to it not only in present times but also in any point in time since the beginning of the project (or at least the establishment of the source code repository).

This information can also of course be used to try to predict a project's future evolution for control, management and releasing policies [4]. Although some interesting facts on the human resources of these type of projects have been shown as for instance the assumption of 'generations' of leading groups that guide the project temporarily, an enormous research effort should be invested in the near future to gain insight into the dynamics of developer integration into libre software projects. In this sense, there are some proposals that try to use ideas from other knowledge areas as for instance the study and characterization of complex systems [8] and the application of classical (social) network analysis in order to understand them.

In addition, it has to be regarded that software is in any case an important valuable good and that all measurements are key points for the calculation of economic parameters. It has to be noted that if cost estimation is already a problematic task in classical (proprietary) software environments where human and technical resources (and their disposal) are known, in the libre software world this is by far more complex [12]. Any attempt with the aim of solving this lack of knowledge is welcome and having accurate data and information on the process is a good start.

Finally, it should be remembered that the source code repository is not the only public information source available for libre software projects. There exist others that may

provide with complementary data. One suite that looks for the integration of software measurement and analysis systems has been proposed by the authors of this paper [16].

References

- [1] A. Atkinson. On the measurement of inequality. *Journal of Economic Theory*, (2):244–263, 1970.
- [2] B. Boehm. *Software Engineering Economics*. Prentice Hall, 1981.
- [3] A. Capiluppi, P. Lago, and M. Morisio. Evidences in the evolution of os projects through changelog analyses. 2003.
- [4] J. R. Ehrenkrantz. Release management within open source projects. 2003.
- [5] D. Germán and A. Mockus. Automating the measurement of open source projects. Portland, Oregon, 2003.
- [6] C. Gini. *On the Measure of Concentration with Espacial Reference to Income and Wealth*. Cowles Commission, 1936.
- [7] M. W. Godfrey and Q. Tu. Evolution in open source software: A case study. 2000.
- [8] J. M. González-Barahona, L. López-Fernández, and G. Robles. Community structure of modules in the apache project. 2004.
- [9] J. M. González-Barahona and G. Robles. Unmounting the "code gods" assumption. Technical report, 2003.
- [10] K. Healy and A. Schussman. The ecology of open-source software development. Technical report, University of Arizona, USA, Jan. 2003.
<http://opensource.mit.edu/papers/healyschussman.pdf>.
- [11] O. Herfindahl. *Copper Costs and Prices: 1870 - 1957*. Baltimore: The John Hopkins Press, 1959.
- [12] S. Koch and G. Schneider. Results from software engineering research into open source development projects using public data. *Diskussionspapiere zum Tätigkeitsfeld Informationsverarbeitung und Informationswirtschaft*, (22), 2000.
<http://www.wi.wu-wien.ac.at/~koch/forschung/sw-eng/wp22.pdf>.
- [13] M. Lehman, J. Ramil, P. Wernick, and D. Perry. Metrics and laws of software evolution - the nineties view. 1997.
- [14] A. Mockus, R. Fielding, and J. Herbsleb. A case study of open source software development: The Apache server. In *Proceedings of the 22nd International Conference on Software Engineering (ICSE 2000)*, pages 263–272, Limerick, Ireland, 2000.
- [15] E. S. Raymond. The cathedral and the bazaar. *First Monday*, 1997.
http://www.firstmonday.dk/issues/issue3_3/raymond/.
- [16] G. Robles, J. M. Gonzalez-Barahona, and R. A. Ghosh. Gluetheos: Automating the retrieval and analysis of data from publicly available software repositories. 2004.
- [17] G. Robles-Martinez, 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. Portland, Oregon, 2003.



Scholar



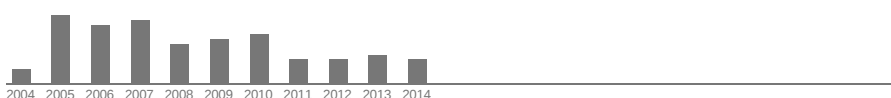
Jesus M. Gonzalez-Barahona

Remote analysis and measurement of libre software systems by means of [PDF] from researchgate.net the CVSAnaY tool

Authors Gregorio RoBIES, Stefan KoCH, Jesús M GonZÁIEZ-BARAHonA, Juan Carlos
Publication date 2004/5/24
Journal Proceedings of the 2nd ICSE Workshop on Remote Analysis and Measurement of Software Systems (RAMSS)
Pages 51-55

Description Libre (free, open source) software is one of the paradigmatic cases where heavy use of telematic tools and user-driven software development are key points. This paper proposes a methodology for measuring and analyzing remotely big libre software projects using publicly-available data from their version control repositories. By means of a tool called CVSAnaY that has been implemented following this methodology, measurements and analyses can be made in an automatic and non-intrusive way, providing real-time and historical data about ...

Total citations [Cited by 94](#)



Scholar articles [Remote analysis and measurement of libre software systems by means of the CVSAnaY tool](#)
 G RoBIES, S KoCH, JM GonZÁIEZ-BARAHonA... - Proceedings of the 2nd ICSE Workshop on Remote ..., 2004
[Cited by 94](#) - [Related articles](#) - [All 9 versions](#)

Dates and citation counts are estimated and are determined automatically by a computer program.

[Help](#) [Privacy](#) [Terms](#) [Provide feedback](#) [My Citations](#)

Evolution and Growth in Large Libre Software Projects*

Gregorio Robles, Juan Jose Amor, Jesus M. Gonzalez-Barahona, Israel Herraiz
GSyC, Universidad Rey Juan Carlos (Madrid, Spain)
{grex,jjamor,jgb,herraiz}@gsvc.escet.urjc.es

Abstract

Software evolution research has recently focused on new development paradigms, studying whether laws found in more classic development environments also apply. Previous works have pointed out that at least some laws seem not to be valid for these new environments and even Lehman has labeled those (up to the moment few) cases as anomalies and has suggested that further research is needed to clarify this issue. In this line, we consider in this paper a large set of libre (free, open source) software systems featuring a large community of users and developers. In particular, we analyze a number of projects found in literature up to now, including the Linux kernel. For comparison, we include other libre software kernels from the BSD family, and for completeness we consider a wider range of libre software applications. In the case of Linux and the other operating system kernels we have studied growth patterns also at the subsystem level. We have observed in the studied sample that super-linearity occurs only exceptionally, that many of the systems follow a linear growth pattern and that smooth growth is not that common. These results differ from the ones found generally in classical software evolution studies. Other behaviors and patterns give also a hint that development in the libre software world could follow different laws than those known, at least in some cases.

1. Introduction and research goals

The number of studies on software evolution is relatively low, despite being a field opened more than 30 years ago. The lessons learned are many, and are summarized in a set of laws, stated by Lehman, which have

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

grown to eight in their latest version [17]. These laws have been validated empirically with some large industrial software projects. Recent research is exploring whether they are applicable to other domains, such as systems developed using eXtreme Programming models, based on the COTS paradigm, etc.

One of this ‘other’ domains is libre software¹. Although its basic difference with ‘traditional’ software lies in the licensing terms, many argue that there are also significant differences in the way they are built. For instance, most of the procedures in libre software are open and public, targeted to ease the followup and joining by new developers, with the aim of forming a developer community in which individuals can play several roles (from core developers to casual bug report submitters). Although there is some literature showing that projects with a surrounding community are exceptions if we consider the whole libre software landscape [13], they are still the most notorious, larger in size and user population, and those which have featured most attention by the public, the industry and the research community (consider for instance Mozilla [6, 18, 19], Linux [9, 19, 21], Apache [18], GNOME [14, 8], or FreeBSD [4]).

For the study presented in this paper, we have considered exactly this kind of libre software projects: large in size (at least in the order of 100K lines of code), and with a large user and developer community. Our intention is to explore how they behave in the context of the laws of software evolution, specially regarding software growth. For this matter, we started by reproducing (with current data) the classical study performed five years ago on the Linux kernel [9], which seemed to question the conformance of libre software projects to some of those laws. Later, we extended the study by doing a similar analysis on other libre software systems in the same domain (operating system kernels): the *BSD family. Fi-

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

ware systems, and even if that were the case (as it seems from our study), why that happens, and to which extent that contradicts Fourth Lehman's Law. Therefore, more projects should be studied, to improve the evidence (or find cases where growth is not linear), and detailed analysis should be performed of how human resources are allocated in libre software projects, with the aim of explaining the linear growth we have found.


8. Acknowledgments

We thank Juan Antonio Almendral, from the Mathematics, Physics and Natural Sciences Department of the Universidad Rey Juan Carlos for his invaluable help with the statistics of this paper.

References

- [1] E. Burd and M. Munro. Evaluating the evolution of a C application. In *Intl Workshop on Principles of Software Evolution*, Fukuoka, Japan, June 1999.
- [2] A. Capiluppi. Models for the evolution of os projects. In *Proceedings of the Intl Conf on Software Maintenance*, pages 65–74, Amsterdam, The Netherlands, 2003.
- [3] A. Capiluppi, M. Morisio, and P. Lago. Evolution of understandability in oss projects. In *Proceedings of the 8th European Conf on Software Maintenance and Reengineering*, Tampere, Finland, 2004.
- [4] T. Dinh-Trong and J. M. Bieman. Open source software development: A case study of freebsd. In *Proceedings of the 10th Intl Software Metrics Symposium*, Chicago, IL, USA, September 2004.
- [5] J. R. Ehrenkrantz. Release management within open source projects. In *Proceedings 3rd Workshop on Open Source Software Engineering*, Portland, Oregon, 2003.
- [6] M. Fischer, M. Pinzger, and H. Gall. Populating a release history database from version control and bug tracking systems. In *Proceedings of the Intl Conf on Software Maintenance*, pages 23–32, Amsterdam, The Netherlands, September 2003.
- [7] H. Gall, M. Jazayeri, R. Klösch, and G. Trausmuth. Software evolution observations based on product release history. In *Proceedings of the Intl Conf on Software Maintenance*, pages 160–170, 1997.
- [8] D. Germán. The GNOME project: a case study of open source, global software development. *J of Softw Process: Improvement and Practice*, 8(4):201–215, 2004.
- [9] M. Godfrey and Q. Tu. Evolution in Open Source software: A case study. In *Proceedings of the Intl Conf on Software Maintenance (ICSM 2000)*, pages 131–142, San Jose, California, 2000.
- [10] M. Godfrey and Q. Tu. Growth, evolution, and structural change in open source software. In *Intl Workshop on Principles of Software Evolution*, Vienna, Austria, September 2001.
- [11] 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.
- [12] J. M. Gonzalez-Barahona, G. Robles, M. Ortuño Pérez, L. Rodero-Merino, J. Centeno-Gonzalez, 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, Hershey, PA, 2004.
- [13] K. Healy and A. Schussman. The ecology of open-source software development. Technical report, University of Arizona, USA, January 2003.
- [14] S. Koch and G. Schneider. Effort, cooperation and coordination in an open source software project: Gnome. *Information Systems Journal*, 12(1):27–42, 2002.
- [15] M. Lehman and J. F. Ramil. Rules and tools for software evolution planning and management. *Annals of Software Engineering*, 11(1):15–44, 2001.
- [16] M. Lehman, J. F. Ramil, and U. Sandler. An approach to modelling long-term growth trends in software systems. In *Intl Conf on Software Maintenance*, pages 219–228, Florence, Italy, November 2001.
- [17] M. Lehman, J. F. Ramil, P. Wernick, and D. Perry. Metrics and laws of software evolution - the nineties view. In *Proceedings of the Fourth Intl Software Metrics Symposium*, Portland, Oregon, 1997.
- [18] 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.
- [19] J. W. Paulson, G. Succi, and A. Eberlein. An empirical study of open-source and closed-source software products. *Transactions on Softw Eng*, 30(4), April 2004.
- [20] 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 Intl Workshop on Mining Software Repositories*, Edinburg, Scotland, UK, 2004.
- [21] 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.
- [22] G. Succi, J. Paulson, and A. Eberlein. Preliminary results from an empirical study on the growth of open source and commercial software products. In *EDSER-3 Workshop*, Toronto, Canada, May 2001.
- [23] W. M. Turski. Reference model for smooth growth of software systems. *IEEE Transactions on Software Engineering*, 22(8):599–600, 1996.
- [24] T. Yamamoto, M. Matsushita, T. Kamiya, and K. Inoue. Measuring similarity of large software systems based on source code correspondence. In *6th Intl PROFES (Product Focused Software Process Improvement) conference, PROFES 2005*, Oulu, Finland, June 2005.

Web Images More... jgbarah@gmail.com



Scholar ← Edit 🗑️ Export ▾



Jesus M. Gonzalez-Barahona

Evolution and growth in large libre software projects

[\[PDF\] from herraiz.org](#)
[Texto completo para UC3M](#)

Authors: Gregorio Robles, Juan Jose Amor, Jesus M Gonzalez-Barahona, Israel Herraiz
 Publication date: 2005/9/5
 Conference: Principles of Software Evolution, Eighth International Workshop on
 Pages: 165-174
 Publisher: IEEE

Description: Abstract Software evolution research has recently focused on new development paradigms, studying whether laws found in more classic development environments also apply. Previous works have pointed out that at least some laws seem not to be valid for these new environments and even Lehman has labeled those (up to the moment few) cases as anomalies and has suggested that further research is needed to clarify this issue. In this line, we consider in this paper a large set of libre (free, open source) software systems ...

Total citations [Cited by 102](#)



Scholar articles [Evolution and growth in large libre software projects](#)
 G Robles, JJ Amor, JM Gonzalez-Barahona, I Herraiz - Principles of Software Evolution, Eighth International ..., 2005
[Cited by 102](#) - [Related articles](#) - [All 10 versions](#)

Dates and citation counts are estimated and are determined automatically by a computer program.

[Help](#) [Privacy](#) [Terms](#) [Provide feedback](#) [My Citations](#)

Developer identification methods for integrated data from various sources

Gregorio Robles, Jesus M. Gonzalez-Barahona

{grex, jgb}@gsyc.escet.urjc.es
Grupo de Sistemas y Comunicaciones
Universidad Rey Juan Carlos
Madrid, Spain

ABSTRACT

Studying a software project by mining data from a single repository has been a very active research field in software engineering during the last years. However, few efforts have been devoted to perform studies by integrating data from various repositories, with different kinds of information, which would, for instance, track the different activities of developers. One of the main problems of these multi-repository studies is the different identities that developers use when they interact with different tools in different contexts. This makes them appear as different entities when data is mined from different repositories (and in some cases, even from a single one). In this paper we propose an approach, based on the application of heuristics, to identify the many identities of developers in such cases, and a data structure for allowing both the anonymized distribution of information, and the tracking of identities for verification purposes. The methodology will be presented in general, and applied to the GNOME project as a case example. Privacy issues and partial merging with new data sources will also be considered and discussed.

1. INTRODUCTION

Most research in the area of mining software repositories has been performed on a single source of data. The reason for this is that tools are usually targeted towards accessing a specific kind of data, which can be retrieved and analyzed uniformly. Data mining for control versioning systems [11], bug-tracking systems, mailing lists and other sources is currently state of the art. The focus of these studies is more on the analysis than in the data extraction process, which can be automated, as has already been discussed [2, 9].

However, there is a wide interest in considering data from several sources and integrating them into a single database, getting richer evidence from the observed matter [5]. The data gathered following this approach can be used for studying several kinds of artifacts relevant to the software develop-

ment process, such as source code files or, as we will discuss in this paper, developers.

As an example of the usefulness of this approximation, let's consider collaboration in libre software¹ projects, which is an active research field. Libre software is produced in part (in many cases a large part) by volunteers, which makes it difficult to predict the future evolution. However, it has at least in some cases produced high-quality software, used by millions of persons around the world. It has been shown that this collaboration follows a Pareto law for commits [11], source code contributions [4], bug reports [8] or mailing list posts [6]; i.e. a small amount of developers of around 20% is responsible for a huge amount of the produced artifacts (around 80%). But although this research on different sources coincide in results, there is still no evidence of coherence. In other words, although it is known that the Pareto distribution appears in several data sources for a given project, are the most active actors for each of those sources (mailing lists, code repositories, bug report systems, etc.) the same ones?

In the specific case of merging information about developers from different repositories, the main difficulty is caused by the many identities that they use from repository to repository, and even for the same one, making tracking difficult. That is the reason why we need methods and tools that can find the different identities of a given developer. These methods, and the data they produce, should be designed to be sharable among research groups, not only for validation purposes but also for enabling the merging of partial data obtained by different teams from different sources.

In general, any study considering individuals in libre software projects, even when using a single data source, is sensible to identity variety. Before performing any analysis on the data set, it is necessary to merge the identities corresponding to the same person. This is for instance the case in the promising case of clustering [3] and social network analysis [7], which are trying to get insight in the structure of libre software projects.

The structure of this paper is as follows. The next section deals with the kinds of identities which are usually found in software-related repositories. The third section is devoted to the extraction of data, its structure and verification. Section

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage, and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

MSR'05, May 17, 2005, St. Louis, Missouri, USA.

Copyright 2005 ACM 1-59593-123-6/05/0005 ...\$5.00.

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

development landscape.

7. ACKNOWLEDGEMENTS

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 Spanish CICYT under project TIN2004-07296.

8. REFERENCES

- [1] A. Capiluppi, P. Lago, and M. Morisio. Evidences in the evolution of os projects through changelog analyses. In *Proceedings of the 3rd Workshop on Open Source Software Engineering*, 2003.
- [2] D. German and A. Mockus. Automating the measurement of open source projects. In *Proceedings of the 3rd Workshop on Open Source Software Engineering*, Portland, USA, 2003.
- [3] R. A. Ghosh. Clustering and dependencies in free/open source software development: Methodology and tools. *First Monday*, 8(4), Apr. 2003.
- [4] R. A. Ghosh and V. V. Prakash. The orbiten free software survey. *First Monday*, 7(5), May 2002.
- [5] J. M. Gonzalez-Barahona and G. Robles. Getting the global picture. In *Proceedings of the Oxford Workshop on Libre Software 2004*, Oxford, UK, June 2004.
- [6] S. Koch and G. Schneider. Effort, cooperation and coordination in an open source software project: Gnome. *Information Systems Journal*, 12(1):27–42, 2002.
- [7] L. Lopez, J. M. Gonzalez-Barahona, and G. Robles. Applying social network analysis to the information in cvs repositories. In *Proceedings of the International Workshop on Mining Software Repositories*, Edinburg, UK, 2004.
- [8] 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.
- [9] G. Robles, J. M. Gonzalez-Barahona, J. Centeno, V. Matellan, and L. Rodero. 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, USA, 2003.
- [10] 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*, Edinburg, Scotland, UK, 2004.
- [11] 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.

APPENDIX

A. A CASE STUDY: GNOME

To debug and complete our methodology, we have applied it to the data from several real libre software repositories. One of the most complete studies we have performed to date has been on the GNOME project, retrieving data from mailing lists, bug tracking system (including bug reports and comments) and from the CVS repository. Next, we offer some results from this study:

- 464,953 messages from 36,399 distinct e-mail addresses have been fetched and analyzed.
- 123,739 bug reports, from 41,835 reporters, and 382,271 comments from 10,257 posters have been retrieved from the bug tracking system.
- Around 2,000,000 commits, made by 1,067 different committers have been found in the CVS repository.
- From these data, 108,170 distinct identities have been identified.
- For those distinct identities, 47,262 matches have been found, of which 40,003 were distinct (therefore, our Matches table contains that number of entries).
- Using the information in the Matches table, we have been able of finding 34,648 unique persons.

This process has been statistically verified by selecting a sample of identities, looking by hand for matches and comparing the results to the corresponding entries in the Matches table. Currently we are completing the Persons table, and performing gender and nationality analysis.


B. AUTOMATIC (POST-IDENTIFICATION) ANALYSIS

The reader has probably noted that the Persons table in Figure 2 includes some fields with personal information. We have devised some heuristics to infer some of them from data in the repositories, usually from the structure of identities. For instance, nationality can be guessed by several means:

- Analyzing the top level domain (TLD) of the various e-mail addresses found in the identities could be a first possibility. The algorithm in this case consists of listing all e-mail addresses, extracting the TLD from them, rejecting those TLD that cannot be directly assigned to a country (.com, .net, .org, etc.) or those who are from “fake” countries (.nu, etc.), and finally looking at the remaining TLDs and count how often they occur. The TLD that is more frequent gives a hint about the nationality of the person. Of course this heuristic is specially bad for US-based actors (since they are not likely to use the US TLD), and for those using .org or .com addresses, quite common in libre software projects.
- Another approach is to use whois data for the second level domain in e-mail address, considering that the whois contact information (which includes a physical mail address) is valid as an estimator of the country of the actor. Of course, this is not always the case.

Other case example of information which can be obtained from identities is the gender. Usually we can infer the gender from the name of the person. However, in some cases it depends on the nationality, since some names may be assigned to males in one country and to females in another. This is for instance the case for Andrea, which in Italy is a male name while in Germany, Spain and other countries is usually for females.

Web Images More... jgbarah@gmail.com



Scholar



Jesus M. Gonzalez-Barahona

Developer identification methods for integrated data from various sources

[\[PDF\] from flosshub.org](#)
[Texto completo para UC3M](#)

Authors: Gregorio Robles, Jesus M Gonzalez-Barahona
 Publication date: 2005/7/1
 Journal: ACM SIGSOFT Software Engineering Notes
 Volume: 30
 Issue: 4
 Pages: 1-5
 Publisher: ACM

Description Abstract Studying a software project by mining data from a single repository has been a very active research field in software engineering during the last years. However, few efforts have been devoted to perform studies by integrating data from various repositories, with different kinds of information, which would, for instance, track the different activities of developers. One of the main problems of these multi-repository studies is the different identities that developers use when they interact with different tools in different contexts. This makes ...

Total citations [Cited by 60](#)



Scholar articles [Developer identification methods for integrated data from various sources](#)
 G Robles, JM Gonzalez-Barahona - ACM SIGSOFT Software Engineering Notes, 2005
[Cited by 60](#) - [Related articles](#) - [All 16 versions](#)

Dates and citation counts are estimated and are determined automatically by a computer program.

[Help](#)
[Privacy](#)
[Terms](#)
[Provide feedback](#)
[My Citations](#)

Evolution of Volunteer Participation in Libre Software Projects: Evidence from Debian

Gregorio Robles, Jesus M. Gonzalez-Barahona
Grupo de Sistemas y Comunicaciones
Universidad Rey Juan Carlos (Madrid, Spain)
{grex, jgb}@gsyc.esctet.urjc.es

Martin Michlmayr
Centre for Technology Management
University of Cambridge
martin@michlmayr.org

Abstract—Most libre software projects rely on the work of volunteers. Therefore, attracting people who contribute their time and technical skills is of paramount importance, both in technical and economic terms. This reliance on volunteers leads to some fundamental management challenges: volunteer contributions are inherently difficult to predict, plan and manage, especially in the case of large projects. In this paper we analyze the evolution in time of the human resources of one of the largest and most complex libre software projects composed primarily of volunteers, the Debian project. Debian currently has around 1300 volunteers working on several tasks: much activity is focused on packaging software applications and libraries, but there is also major work related to the maintenance of the infrastructure needed to sustain the development. We have performed a quantitative investigation of data from almost seven years, studying how volunteer involvement has affected the software released by the project and the developer community itself.

Index Terms—libre software engineering, human resources, volunteer developers, software evolution.

I. INTRODUCTION

Volunteer contributions are the base of most libre software projects¹. However, the characteristics and way of working of volunteers can be quite different from those of employees, which are the main force behind traditional software development. Volunteers can contribute with the amount of effort they want, can commit for the time period they consider convenient, and can devote their time to the tasks they may prefer, given that the context of the project allows them to do so. Despite of this, some libre software projects have produced software which has gained significant popularity. This shows that the unstructured collaboration of volunteers is a viable software development strategy, even if it is associated with certain challenges related to project management and quality. In this paper we will explore how these voluntary contributions have been working in the specific case of a large libre software project, to have some actual data about their behavior. We started the study stating some questions, for some of which we thought we already knew the answer. To our surprise, we found out that, even with the high knowledge we thought we had about the history of the project, the data told a different tale.

We define in this paper volunteers as those who collaborate in libre software projects in their free time not profiting economically in a direct way from their effort.

¹In this paper we will use the term “libre software” to refer to any software licensed under terms compliant with the FSF definition of “free software”, and the OSI definition of “open source software”, thus avoiding the controversy between those two terms. However, in the specific case of Debian, the project has its own definition of “free software”, the Debian Free Software Guidelines, from which the OSI definition originated later.

Volunteers can be IT-related professionals or not, but their professional activity is not the one they perform on a given libre software project. Although the vast majority of participants in libre software projects are following our definition volunteers there also exist non-volunteers (also known as paid employees), i.e. those whose professional activity is to work on that specific project. In a study on the GNOME project [1], German states that paid employees from various companies are usually responsible for less attractive tasks, such as project design and coordination, testing, documentation and bug fixing. Also, “[m]ost of the paid developers in GNOME were, at some point, volunteers. Essentially for the volunteers, their hobby became their job” [1].

The involvement of volunteers, of course, raises new economic issues that have to be taken into account for business strategies around libre software. Collaboration from volunteers is difficult to predict, but if it is given it may add value to a software system in very economic terms for a software company.

The structure of this paper is as follows. The next section briefly explains the nature of maintainers in the Debian project, and of the Debian operating system. Then we state the questions we aimed to answer before starting this study, including possible answers we expected at that time. After that, we explain the methodology we devised and followed to answer those questions, and the data sources we used. We later show the actual results obtained from the empirical analysis, and contrast them with what we had expected. The paper ends with some conclusions and lessons learned.

II. MAINTAINERS IN THE DEBIAN PROJECT

Debian is an operating system completely based on libre software [2], [3]. It includes a large number of applications, such as the GNU tools and Mozilla, and the system is known for its solid integration of different software components. Debian’s most popular distribution, Debian GNU/Linux, is based on the Linux kernel. Ports to other kernels, such as the Hurd and FreeBSD, are in development.

One of the main characteristics of the Debian distribution is that during the whole life of the project it has been maintained by a group of volunteers, which has grown to quite a substantial number. These individuals devote their own time and technical skills to the creation and integration of software packages, trying to supply users with a robust system which provides a lot of functionality and technical features.

One of the main characteristics of the Debian distribution is that the bulk of work has always been performed by volunteers; furthermore, the project has grown to substantial

no means for forcing any single developer to do any given task or may leave the project during important development phases. It is impossible to infer the behavior of volunteer developers just from the study of a single project, but given the size and relevance of the Debian project, at least some conclusions can be exposed as hypothesis for validating in later research efforts.

One of them is the stability of volunteer work over time. The mean life of volunteers in the project is probably larger than in many software companies, which would have a clear impact on the maintenance of the software (it would be likely that developers with experience in a module be available for its maintenance over long periods of time). Another one is that volunteers tend to take over more work with the passing of time if they manage to stay in the project: in other words, they voluntarily increase their responsibilities in the project. Whether this is because it is easier for them because of their experience, or because they devote more effort to the project, is for now an open question. Yet a third one is the stability of the voluntary effort when some individuals leave the project: most of their work is taken over by other developers. Therefore, despite being completely based on volunteers, the project organizes itself rather well with respect to leavings, which is an interesting lesson about how the project can survive in the long term.

As a final summary, we have found that given that there are no formal ways of forcing a developer to assume any given task, voluntary efforts seem to be more stable over time, and more reliable with respect to individuals leaving the project than we had expected in advance.

VII. ACKNOWLEDGEMENTS

The work of Gregorio Robles and Jesus M. Gonzalez-Barahona has been funded in part by the European Commission under the CALIBRE CA, IST program, contract number 004337, in part by the Universidad Rey Juan Carlos under project PPR-2004-42 and in part by the Spanish CICyT under project TIN2004-07296. The work of Martin Michlmayr has been funded in part by Fotango, the NUUG Foundation and the EPSRC. We also want to thank the anonymous reviewers for their extensive comments.

REFERENCES

- [1] D. M. German, "Decentralized Open Source global software development, the GNOME experience," *Journal of Software Process: Improvement and Practice*, vol. 8, no. 4, pp. 201–215, 2004.
- [2] M. Monga, "From Bazaar to Kibbutz: How freedom deals with coherence in the Debian project," in *Proc 4th Workshop on Open Source Software Engineering*, Edinburg, UK, 2004.
- [3] S. O'Mahony, "Guarding the commons: how community managed software projects to protect their work," *Research Policy*, no. 32, pp. 1179–1198, 2003.
- [4] "Debian Social Contract," http://www.debian.org/social_contract.
- [5] G. Garzarelli and R. Galoppini, "Capability coordination in modular organization: Voluntary FS/OSS production and the case of Debian GNU/Linux," November 2003.
- [6] M. Michlmayr and B. M. Hill, "Quality and the reliance on individuals in free software projects," in *Proc 3rd Workshop on Open Source Software Engineering*, Portland, USA, 2003, pp. 105–109. [Online]. Available: http://www.cyrius.com/publications/michlmayr_hill-reliance.pdf
- [7] M. Michlmayr, "Managing volunteer activity in free software projects," in *Proc USENIX 2004 Annual Technical Conference, FREENIX Track*, Boston, USA, 2004, pp. 93–102. [Online]. Available: <http://www.cyrius.com/publications/michlmayr-mia.pdf>

- [8] 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 *Free/Open Source Software Development*, S. Koch, Ed. Hershey, PA, USA: Idea Group Publishing, 2004, pp. 27–58.
- [9] M. Lehman, J. Ramil, P. Wernick, and D. Perry, "Metrics and laws of software evolution - the nineties view," in *Proceedings of the Fourth International Software Metrics Symposium*, Portland, Oregon, 1997.
- [10] C. Lameter, "Debian GNU/Linux: The past, the present and the future," <http://telemetrybox.org/tokyo/>, 2002.
- [11] 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*, vol. II, no. 6, pp. 60–66, Dec. 2001.
- [12] G. Hertel, S. Niedner, and S. Herrmann, "Motivation of software developers in open source projects: an Internet-based survey of contributors to the Linux kernel," *Research Policy*, vol. 32, no. 7, pp. 1159–1177, 2003.
- [13] "The Open Source maturity model: A methology for assessing open source software," <http://www.navicasoft.com/pages/osmm.htm>.
- [14] B. Golden, *Succeeding with Open Source*. Addison-Wesley Professional, 2004.

Web Images More... jgbarah@gmail.com

Google

Scholar



Jesus M. Gonzalez-Barahona

Evolution of volunteer participation in libre software projects: evidence from Debian [\[PDF\] from cyrius.com](#)

Authors: Gregorio Robles, Jesus M Gonzalez-Barahona, Martin Michlmayr
Publication date: 2005/7/11
Journal: Proceedings of the 1st International Conference on Open Source Systems
Pages: 100-107

Description: Abstract—Most libre software projects rely on the work of volunteers. Therefore, attracting people who contribute their time and technical skills is of paramount importance, both in technical and economic terms. This reliance on volunteers leads to some fundamental management challenges: volunteer contributions are inherently difficult to predict, plan and manage, especially in the case of large projects. In this paper we analyze the evolution in time of the human resources of one of the largest and most complex libre software projects ...

Total citations: [Cited by 58](#)



Scholar articles: [Evolution of volunteer participation in libre software projects: evidence from Debian](#)
G Robles, JM Gonzalez-Barahona, M Michlmayr - Proceedings of the 1st International Conference on ..., 2005
[Cited by 58](#) - [Related articles](#) - [All 16 versions](#)

Dates and citation counts are estimated and are determined automatically by a computer program.

[Help](#) [Privacy](#) [Terms](#) [Provide feedback](#) [My Citations](#)

Self-organized development in libre software projects: a model based on the stigmergy concept*

Gregorio Robles
Universidad Rey Juan Carlos
grex@gsync.escet.urjc.es

Juan Julian Merelo
Universidad de Granada
jmerelo@geneura.ugr.es

Jesus M. Gonzalez-Barahona
Universidad Rey Juan Carlos
jgb@gsync.escet.urjc.es

Extended Abstract

Keywords: Process modeling and simulation, libre (free/open) source software, self-organization, software evolution

Abstract

Libre software development is difficult (if not impossible) to explain by using traditional techniques of software engineering. We propose an stigmergy-based model which explains some behaviors usual in it, specially the selection by developers of the projects where they work, and the growth of those projects. We have also verified it comparing its predictions to the results of studies of real projects.

1. Introduction

Libre (free/open source) software¹ is becoming an increasingly important component of today's software industry. While many studies have focused on the inner working of a few, usually successful, projects, few have paid attention to explain the big picture of how these projects have become successful. This is probably due to its distributed and mostly self-organized development nature, which makes it difficult to understand and research.

One of the most surprising assumptions for the libre software phenomenon is that its development process does not obey a 'classical' software engineering development process: there are no (or few) pre-defined requirements, no detailed design at all and a lack of inter-process documentation [10]. Last but not least, libre software projects do not follow a clear predefined hierarchical structure where a central authority shows the way to go. In short, they show trends of being self-regulated and self-organized.

*The work of Gregorio Robles and Jesus M. Gonzalez-Barahona has been funded in part by the European Commission, under the CALIBRE CA, IST program, contract number 004337.

¹Through this extended abstract we will use the term "libre software" to refer to any code that conforms either to the definition of "free software" (according to the FSF) or "open source software" (according to the OSI).

In this paper, an analogy between the libre software phenomenon and the way some social insects perform large-scale works is proposed. The analogy is based on the stigmergy concept, which states that communication (by means of stimuli) does not happen directly among entities (in our case developers) but through changes in the environment. Stigmergy makes an autocatalytic reaction possible as it has been observed in bazaar-driven self-organized libre software projects.

We have built a model based upon these ideas, implemented a simulation software, calibrated the model with data from previous studies and finally verified its output comparing it to results from investigations performed on libre software.

2. Self-organization through stigmergy

In the late 1950s the french biologist P.P.-Grassé² realized while studying the construction of termites nests that the product of certain behaviors was consequence of the effects produced in the local environment by previous behaviors (of possibly other termites). He called this phenomenon stigmergy² [5].

Grassé² observed that when termites build their nest they start in a randomly manner without any coordination. Once a given point is achieved an area becomes a significant stimulus for other termites which then collaborate in the construction of the nest.

Stigmergy appears mainly in social insects such as termites, ants and some kinds of spiders. Their activity does not depend on the direct interactions with other insect-workers, but on the structure of the environment. Individual behavior is controlled and guided by previous work, i.e. the changes in the environment are dynamically used for self-organization and coordination of the colony. An insect creates, by means of its activity, a structure which stimulates other members of the colony, causing them to perform an specific activity.

²From the Greek 'stigma' (mark/sign) and 'ergos' (work)

model. Data from previous research studies and surveys performed on libre software developers have been used for this purpose. So, for instance, the mean time spent on a project by libre software developers has been taken from several surveys and the productivity has been measured in SLOC in order to be comparable with the COCOMO cost prediction model [1].

Verification of the model has been performed against results obtained by other research studies. We can classify these results in two sets: facts about developer contributions, and evidences about the software produced. In both cases, with the former calibration, our model shows similar trends to the one obtained in these studies.

Regarding developer contributions we have observed whether the model shows inequality patterns for developer contribution to projects, as reported by Koch et al. [7] and if projects are led throughout the lifetime of the project by several generations of core groups [3]. [6] showed that the number of developers in projects follows a power-law distribution, so we also checked for that. Other contribution patterns as stated by Mockus et al. on Apache and Mozilla [9] are also compared with the results we have obtained.

Regarding the size of the developed software, we test if the mean size of software projects remains constant in time and if we obtain a power-law for project size as some studies on GNU/Linux distributions have thrown [4]. Finally, we looked at the software evolution patterns that our model provides and compare it to the current state of the art [2, 8].

5. Conclusions and further work

Our primary conclusion is that the libre software phenomenon can indeed be modeled as a stigmergic one. This means, for instance, that the individual productivity may not be as important as the total production as a community; or that stigmergic mechanisms should be used in order to increase productivity.

The model we propose here shows patterns similar to those reported in real world studies on libre software, although finer calibration and further discussion on the variables and threshold values is necessary.

If this model shows to be valuable, future research could focus on researching how the change of some parameters may affect the development of software in the libre software world. This could be the case of a company wanting to hire developers for a libre software project. An interesting extension could be to study how other ‘communities’ present in libre software development (i.e. translators, documenters, etc.) can be included in a model like the one presented. A reality check of the project could also be necessary, with elimination of artifacts or identification with real-life aspects of software development.

References

- [1] B. Boehm. *Software Engineering Economics*. Prentice Hall, 1981.
- [2] M. W. Godfrey and Q. Tu. Evolution in Open Source software: A case study. In *Proceedings of the International Conference on Software Maintenance (ICSM 2000)*, pages 131–142, San Jose, California, 2000.
- [3] J. M. Gonzalez-Barahona and G. Robles. Unmounting the “code gods” assumption. In *Proceedings of the Fourth International Conference on eXtreme Programming and Agile Processes in Software Engineering*, 2003.
- [4] J. M. Gonzalez-Barahona, G. Robles, M. Ortuño-Perez, L. Rodero-Merino, J. Centeno-Gonzalez, V. Matellan-Olivera, E. Castro-Barbero, and P. de-las Heras-Quirós. *Free/Open Source Software Development*, chapter Analyzing the anatomy of GNU/Linux distributions: methodology and case studies (Red Hat and Debian). Idea Group Inc., 2004.
- [5] P.-P. GrassÃ©. La reconstruction du nid et les coordinations inter-individuelles chez bellicositermes natalensis et cubitermes sp. la thÃ©orie de la stigmergie: Essai d’interpretation du comportement des termites constructeurs. *Insectes Sociaux*, (6):41–81, 1959.
- [6] K. Healy and A. Schussman. The ecology of open-source software development. Technical report, University of Arizona, USA, Jan. 2003.
- [7] S. Koch and G. Schneider. Effort, cooperation and coordination in an open source software project: GNOME. *Information Systems Journal*, 12(1):27–42, 2002.
- [8] M. Lehman, J. Ramil, P. Wernick, and D. Perry. Metrics and laws of software evolution - the nineties view. In *Proceedings of the Fourth International Software Metrics Symposium*, Albuquerque, NM, USA, 1997.
- [9] 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.
- [10] E. S. Raymond. The cathedral and the bazaar. *First Monday*, 3(3), 2000.
- [11] T. Susi and T. Ziemke. Social cognition, artefacts and stigmergy: A comparative analysis of theoretical frameworks for the understanding of artefact-mediated collaborative activity. *Journal of Cognitive Systems Research*, (2):273–290, 2001.

Web Images More... jgbarah@gmail.com

Google

Scholar ← Edit 🗑 Export ▾



Jesus M. Gonzalez-Barahona

Self-organized development in libre software: a model based on the stigmergy concept [\[PDF\]](#) from [simula.no](#)

Authors: Gregorio Robles, Juan Julian Merelo, Jesus M Gonzalez-Barahona
Publication date: 2005/5/14
Journal: ProSim'05
Pages: 16

Description: Abstract—Libre (free, open source) software projects are lately getting increasing attention from the research community; for instance, several studies have focused on the inner working of some successful projects. However, there is still little emphasis on trying to explain the landscape of libre software development at large, maybe due to the distribution of developers, to the (in many cases) non-compulsory nature of their relationships, and to the extreme importance of motivation to attract resources to a project. In this paper we ...

Total citations: [Cited by 35](#)



Scholar articles: [Self-organized development in libre software: a model based on the stigmergy concept](#)
G Robles, JJ Merelo, JM Gonzalez-Barahona - ProSim'05, 2005
[Cited by 35](#) - [Related articles](#)

Dates and citation counts are estimated and are determined automatically by a computer program.

[Help](#) [Privacy](#) [Terms](#) [Provide feedback](#) [My Citations](#)

Comparison between SLOCs and number of files as size metrics for software evolution analysis

Israel Herraiz, Gregorio Robles, Jesús M. González-Barahona
Grupo de Sistemas y Comunicaciones
Universidad Rey Juan Carlos, SPAIN*
{herraiz, grex, jgb}@gsync.escet.urjc.es

Andrea Capiluppi
Dipartimento di Automatica e Informatica
Politecnico di Torino, ITALY
Andrea.Capiluppi@polito.it[†]

Juan F. Ramil
Computing Department
Faculty of Maths and Computing
The Open University, UK
J.F.Ramil@open.ac.uk

Abstract

There are some concerns in the research community about the convenience of using low-level metrics (such as SLOC, source lines of code) for characterizing the evolution of software, instead of the more traditional higher lever metrics (such as the number of modules or files). This issue has been raised in particular after some studies that suggest that libre (free, open source) software evolves differently than ‘traditional’ software, and therefore it does not conform to Lehman’s laws of software evolution. Since those studies on libre software evolution use SLOCs as the base metric, while Lehman’s and other traditional studies use modules or files, it is difficult to compare both cases. To overcome this difficulty, and to explore the differences between SLOC and files/modules counts in libre software projects, we have selected a large sample of programs and have calculated both size metrics over time. Our study shows that in those cases the evolution patterns in both cases (counting SLOCs or files) is the same, and that some patterns not conforming to Lehman’s laws are indeed apparent.

Keywords: metrics, software evolution, libre software, empirical studies

1. Introduction and aims

Thirty years of research on software evolution have resulted in a set of empirical observations, known as

Lehman’s Laws of Software Evolution [6]. Although the number of laws has grown from three (in the seventies) to eight (in their latest version [5]), all of them have been empirically proved, by studying projects developed in traditional industrial software development environments.

In recent times, the rise of a new development phenomenon, libre software¹, has opened new horizons to the analysis of software evolution, at least with respect to two issues. The first one is whether the *laws* of software evolution apply to these new environments, where management is loose and contributions from third parties, mainly volunteers, are fostered. The second one derives from the fact that this type of projects make available to researchers a large quantity of public information about the development process which can be retrieved and analyzed. This offers the possibility of having a general view of the landscape instead of just the results of a small number of selected case studies.

Several authors have analyzed the evolution of libre

*The work by the researchers at URJC has been funded in part by the European Commission, under the CALIBRE CA, IST program, contract number 004337, and by the Spanish CICYT under project TIN2004-07296. Israel Herraiz has been funded in part by Consejería de Educación of Comunidad de Madrid and European Social Fund, under grant number 01/FPI/0582/2005. Part of this work has been developed while Israel Herraiz was visiting The Open University.

[†] Visiting researcher at The Open University, UK

1. Through this paper we will use the term “libre software” to refer 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).


complexity makes more difficult to add functionality to a project, and causes it to slow down. Developers can not improve their productivity forever, as an increasing number of developers working in the same artifact provokes inefficiencies.

Therefore, more studies are needed to enlighten some of this doubts, and to better understand the fast growth of libre software. With time, even the formulation of a theoretical framework to explain this behavior should be possible.

References

- [1] J. J. Amor-Iglesias, J. M. González-Barahona, G. Robles, and I. Herraiz. Measuring libre software using debian 3.1 (sarge) as a case study: Preliminary results. *Upgrade*, VI(3), June 2005.
- [2] M. Godfrey and Q. Tu. Growth, evolution, and structural change in open source software. In *International Workshop on Principles of Software Evolution*, Vienna, Austria, September 2001.
- [3] 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.
- [4] S. Koch. Evolution of open source system software systems - a large scale investigation. In *Proceedings of the First International Conference on Open Source Systems*, 2005.
- [5] M. Lehman, J. Ramil, P. Wernick, and D. Perry. Metrics and laws of software evolution - the nineties view. In *Proceedings of the Fourth International Software Metrics Symposium*, 1997.
- [6] M. M. Lehman and L. A. Belady, editors. *Program evolution: processes of software change*. Academic Press Professional Inc., 1985.
- [7] M. M. Lehman, D. E. Perry, and J. F. Ramil. Implications of evolution metrics on software maintenance. In *ICSM*, pages 208–218, 1998.
- [8] M. M. Lehman, J. F. Ramil, and U. Sandler. An approach to modelling long-term growth trends in software systems. In *Software Maintenance, 2001. Proceedings. IEEE International Conference on*, pages 219–228, 2001.
- [9] G. Robles, J. J. Amor, J. M. Gonzalez-Barahona, and I. Herraiz. Evolution and growth in large libre software projects. In *Proceedings of the 8th International Workshop on Principles of Software Evolution*, pages 165–174, Lisbon, September 2005. IEEE Computer Society.
- [10] 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.
- [11] 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.
- [12] Sloccount. <http://www.dwheeler.com/sloccount/>.
- [13] G. Succi, J. Paulson, and A. Eberlein. Preliminary results from an empirical study on the growth of open source and commercial software products. In *EDSER-3 Workshop, co-located with ICSE 2001*, Toronto, Canada, May 2001.

Web Images More... jgbarah@gmail.com



Scholar

[←](#) [Edit](#) [🗑](#) [Export ▾](#)



Jesus M. Gonzalez-Barahona

Comparison between SLOCs and number of files as size metrics for software evolution analysis

[\[PDF\]](#) from herraiz.org
[Texto completo para UC3M](#)

Authors: Israel Herraiz, Gregorio Robles, Jesús M González-Barahona, Andrea Capiluppi, Juan F Ramil
Publication date: 2006/3/22
Conference: Software Maintenance and Reengineering, 2006. CSMR 2006. Proceedings of the 10th European Conference on
Pages: 8 pp.-213
Publisher: IEEE

Description: Abstract There are some concerns in the research community about the convenience of using low-level metrics (such as SLOC, source lines of code) for characterizing the evolution of software, instead of the more traditional higher lever metrics (such as the number of modules or files). This issue has been raised in particular after some studies that suggest that libre (free, open source) software evolves differently than 'traditional' software, and therefore it does not conform to Lehman's laws of software evolution. Since those studies ...

Total citations [Cited by 43](#)



Scholar articles [Comparison between SLOCs and number of files as size metrics for software evolution analysis](#)
I Herraiz, G Robles, JM González-Barahona... - ... Maintenance and Reengineering, 2006. CSMR 2006., 2006
[Cited by 43](#) - [Related articles](#) - [All 9 versions](#)

Dates and citation counts are estimated and are determined automatically by a computer program.

[Help](#) [Privacy](#) [Terms](#) [Provide feedback](#) [My Citations](#)

Effort Estimation by Characterizing Developer Activity*

Juan Jose Amor
jjamor@gsync.escet.urjc.es

Gregorio Robles
grex@gsync.escet.urjc.es

Jesus M. Gonzalez-Barahona
jgb@gsync.escet.urjc.es

Grupo de Sistemas y Comunicaciones
Universidad Rey Juan Carlos
Mostoles, Spain

ABSTRACT

During the latest years libre (free, open source) software has gained a lot of attention from the industry. Following this interest, the research community is also studying it. For instance, many teams are performing quantitative analysis on the large quantity of data which is publicly available from the development repositories maintained by libre software projects. However, not much of this research is focused on cost or effort estimations, despite its importance (for instance, for companies developing libre software or collaborating with libre software projects), and the availability of some data which could be useful for this purpose. Our position is that classical effort estimation models can be improved from the study of these data, at least when applied to libre software. In this paper, we focus on the characterization of developer activity, which we argue can improve effort estimation. This activity can be traced with a lot of detail, and the resulting data can also be used for validation of any effort estimation model.

Categories and Subject Descriptors

D.2.9 [Software Engineering]: Management—*Cost estimation, productivity*

General Terms

Measurement, Economics

Keywords

effort estimation, open source software, mining software repositories, developer characterization, software economics

*This work has been funded in part by the European Commission under the CALIBRE CA, IST program, contract number 004337.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

EDSER'06, May 27, 2006, Shanghai, China.

Copyright 2006 ACM 1-59593-085-X/06/0005 ...\$5.00.

1. INTRODUCTION

Libre software¹ has gained a lot of attention from the software industry and from software users in general. Following that interest, the research community has also paid attention to the different models of development which have produced the huge quantity of libre software available today. Some findings of this effort have proven to be interesting from a software engineering point of view, such as the fast growth patterns [8] or the low defect density [14] found in several libre software systems.

Many companies and public administrations have been involved in libre software development for several years, and a lot more are entering that way. Their involvement can be quite different: hiring developers to continue their work in successful libre software projects, releasing some of their own programs as libre software or sponsoring the resulting project, promoting the creation of new libre software projects, among others. Companies of all sizes and in all markets are exploring with these approaches, from large multinationals such as Novell, Sun Microsystems or IBM to very small SMEs. Recently many public administrations, whether at national, regional or local level have also shown interest in this kind of software.

However, and despite its significant importance for companies (and for libre software projects themselves), the work on effort estimation in this area is rare. While many classical estimation models take the size of code as one of the main parameters for estimating costs, we argue that a better estimation can be achieved by taking into consideration other metrics based in a detailed characterization of developer activity. In other words, instead of basing the estimation on the characteristics of artifacts produced by the project, better results could be obtained if the starting point is the study of the behavior of the actors involved.

In the case of libre software this approach is specially interesting because the effort can not be easily tracked from, e.g., company records, for the reason that such records do usually not exist. Volunteer work is an important factor in most projects, and the effort contributed by agents outside the development team (for instance, in the form of bug fixes, feedback on design decisions, or patches) can be of great importance. All this effort cannot be tracked by look-

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

- *SLOCCount*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 years [1].

6. CONCLUSIONS

In this position paper, classical effort estimations have been briefly reviewed, showing the problems they present when applied to libre software development. It has been argued that in libre software projects there are several public sources of additional information, which may help in the characterization of developer activity. Several results and tools have been outlined, showing how libre software developer activity can be deeply traced. By using these data, effort and cost can be better estimated.

7. ACKNOWLEDGMENTS

The main ideas presented in this paper have been discussed with many researchers, of which we want to mention Paul David (Oxford and Stanford Univ.), Jean-Michel Dalle (Univ. Pierre and Marie Curie) and Rishab Ghosh (MERIT, Univ. of Maastricht).

8. REFERENCES

- [1] J. J. Amor, G. Robles, J. M. González-Barahona, and I. Herraiz. From pigs to stripes: A travel through debian. In *Proceedings of the DebConf5 (Debian Annual Developers Meeting)*, Helsinki, Finland, July 2005.
- [2] B. B. Boehm. *Software Engineering Economics*. Prentice Hall, 1981.
- [3] B. W. Boehm, E. Horowitz, R. Madachy, D. Reifer, B. K. Clark, B. Steece, W. A. Brown, S. Chulani, and C. Abts. *Software Cost Estimation with Cocomo II*. Prentice Hall PTR, January 2000.
- [4] S. D. Conte, H. E. Dunsmore, and V. Y. Shen. *Software Engineering Metrics and Models*. Menlo Park, Calif. : Benjamin/Cummings Pub. Co., 1986.
- [5] J. J. Cuadrado, A. Amescua, L. García, O. Marbán, and M. I. Sánchez. Revision and classification of current software cost estimation models. In *6th World multi-conference on systemics, cybernetics and informatics. Orlando, FL*, pages 339–341, July 2002.
- [6] D. M. Germán. An empirical study of fine-grained software modifications. In *Proceedings of the International Conference in Software Maintenance*, Chicago, IL, USA, 2004.
- [7] R. A. Ghosh, G. Robles, and R. Glott. Software source code survey (free/libre and open source software: Survey and study). Technical report, International Institute of Infonomics. University of Maastricht, The Netherlands, June 2002. <http://www.infonomics.nl/FLOSS/report>.
- [8] 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.
- [9] J. M. González-Barahona, L. López-Fernández, 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.
- [10] J. M. González-Barahona and G. Robles. Unmounting the “code gods” assumption. Technical report, Grupo de Sistemas y Comunicaciones, Universidad Rey Juan Carlos, Madrid, Spain, 2003. <http://libresoft.urjc.es/html/downloads/xp2003-barahona-robles.pdf>.
- [11] K. Healy and A. Schussman. The ecology of open-source software development. Technical report, University of Arizona, USA, Jan. 2003. <http://opensource.mit.edu/papers/healyschussman.pdf>.
- [12] I. Herraiz, G. Robles, J. J. Amor, T. Romera, and J. M. Gonzalez-Barahona. The processes of joining in global distributed software projects. Accepted in *Global Software Development for the Practitioner Workshop 2006*. Available from the authors at request.
- [13] L. López, J. M. González-Barahona, and G. Robles. Applying social network analysis to the information in CVS repositories. In *Proceedings of the International Workshop on Mining Software Repositories*, pages 101–105, Edinburg, UK, 2004.
- [14] 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.
- [15] 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.
- [16] G. Robles, J. M. González-Barahona, J. Centeno-González, V. Matellán-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, USA, 2003.
- [17] G. Robles, J. M. González-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.
- [18] G. Robles, J. M. González-Barahona, and I. Herraiz. An empirical approach to Software Archaeology. In *Proceedings of the International Conference on Software Maintenance (poster session)*, Budapest, Hungary, September 2005.
- [19] G. Robles, S. Koch, and J. M. González-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)*, pages 51–56, Edinburg, Scotland, UK, 2004.

⁸We use an enhanced version of the software originally authored by David A. Wheeler: <http://www.dwheeler.com/sloccount/>



Scholar



Jesus M. Gonzalez-Barahona

Effort estimation by characterizing developer activity

[\[PDF\] from urjc.es](#)
[Texto completo para UC3M](#)

Authors Juan Jose Amor, Gregorio Robles, Jesus M Gonzalez-Barahona
Publication date 2006/5/27
Conference Proceedings of the 2006 international workshop on Economics driven software engineering research
Pages 3-6
Publisher ACM

Description Abstract During the latest years libre (free, open source) software has gained a lot of attention from the industry. Following this interest, the research community is also studying it. For instance, many teams are performing quantitative analysis on the large quantity of data which is publicly available from the development repositories maintained by libre software projects. However, not much of this research is focused on cost or effort estimations, despite its importance (for instance, for companies developing libre software or collaborating with ...

Total citations [Cited by 24](#)



Scholar articles [Effort estimation by characterizing developer activity](#)
 JJ Amor, G Robles, JM Gonzalez-Barahona - Proceedings of the 2006 international workshop on ..., 2006
[Cited by 24](#) - [Related articles](#) - [All 10 versions](#)

Dates and citation counts are estimated and are determined automatically by a computer program.

[Help](#)
[Privacy](#)
[Terms](#)
[Provide feedback](#)
[My Citations](#)

The Processes of Joining in Global Distributed Software Projects

Israel Herraiz, Gregorio Robles, Juan José Amor, Teófilo Romera and Jesús M. González Barahona*
Universidad Rey Juan Carlos
Madrid, Spain

{herraiz, grex, jjamor, teo, jgb}@gsync.escet.urjc.es

ABSTRACT

Libre (free / open source) software is a good example of global software development. Thousands of projects, in a wide range of domains which involve hundreds of thousands of developers and contributors from all around the world. Some large (both in size and developer community) libre software projects have shown evidence of producing code with complete functionality and fast evolution (with linear or superlinear growth), while maintaining low defect density. Many companies are exploring how to benefit from this situation, considering several approaches related to libre software development. For instance, some of them have hired full-time developers, focusing their work on some libre software projects they consider strategic.

However, before joining the core development team of the project, these hired developers have to follow a process of software comprehension, and get used to the rules and communication mechanisms used in the project. We were interested in the differences between this case and that of volunteer developers working in the same project. Therefore, we studied the duration and basic characteristics of this joining process for the developers of GNOME (a well known, large, libre software project). In our analysis, we have found two groups with clearly different joining patterns. Moreover, we have related those patterns to the different behaviors of volunteer and hired developers: volunteers tend to follow a step-by-step joining process, while hired developers usually experience a “sudden” integration. Some reasons for this different behavior are also discussed.

Categories and Subject Descriptors

D.2.9 [Software Engineering]: Management—*program-*

*This work has been funded in part by the European Commission, under the CALIBRE CA, IST program, contract number 004337. Israel Herraiz has been supported in part by Consejería de Educación of Comunidad de Madrid and European Social Fund, under grant number 01/FPI/0582/2005

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

GSD'06, May 23, 2006, Shanghai, China.

Copyright 2006 ACM 1-59593-085-X/06/0005 ...\$5.00.

ming teams, time estimation; K.6.3 [Management of computing and information systems]: Software Management—*software development, software process*

General Terms

Management, Measurement, Human Factors

Keywords

global software development, libre software, membership integration, empirical studies, onion model

1. INTRODUCTION

Libre (free, open source)¹ software has gained a lot of attention from the public and the software engineering community during the last years. Libre software development processes seem to be better in the sense of fast growth [4] and low defect density [8]. Because of these and other reasons, some companies are interested in libre software projects, maybe allocating employees to contribute in a libre software project which is strategic for the company, maybe outsourcing the development and trying to spin off a new community around a product released as libre software.

We have focused on the first case, and have analyzed the situation in GNOME, a well known and studied [3, 7] libre software project. GNOME is sponsored by several companies via its foundation². These companies are interested in the success of GNOME, and in most cases contribute to the development with full time employees.

In order to be able of improving and increasing the functionality (in other words, to be able of writing code), these hired developers must comprehend a large software artifact, and learn the communication mechanisms used by the project. As it is often found in libre software projects, the main communication stream in GNOME is its collection of mailing lists. Most of the feedback is managed through a bug tracking system (Bugzilla), and all the code is stored in a version control system (CVS, at the time of writing this paper, switching to Subversion). Therefore, these developers must read the source code of the module they are

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

²See <http://foundation.gnome.org>.

| Developer | 1st message | 1st report | 1st fix | 1st commit | # commits | # messages | # bugs | # comments |
|-----------|-------------|------------|----------|------------|-----------|------------|--------|------------|
| 20 | 11/13/96 | 07/18/01 | 02/24/01 | 09/02/99 | 5753 | 8673 | 6 | 19 |

Table 6: Some statistics about the selected developers (rest)

| Samples | t to commit | sd. dev. | t to bug report | sd. dev. | t to bug fix | sd. dev. |
|--------------|---------------|----------|-------------------|----------|----------------|----------|
| Whole sample | 13.49 | 19.86 | 21.92 | 18.29 | 22.42 | 16.48 |
| Group 1 | 29.58 | 17.58 | 29.35 | 16.18 | 35.01 | 13.69 |
| Group 2 | 0.19 | 6.75 | 12.57 | 12.52 | 13.11 | 8.727 |

Table 7: Progress metrics for each group (in months)



Figure 4: Activity diagram for some developers in Group 2

Therefore, in short, we can say that the onion model is followed only by the volunteer developers in our sample, but not by those working for the project as employees. However, for all the developers following the model, the observed joining pattern is quite similar.

Although the sample selected for the study could seem tiny at first sight, it is important to notice that it has been designed to cover one of the most interesting cases of developers joining a project. However, our selection criteria leaves outside very interesting cases, such as the developers of Ximian (now Novell), probably the company with most influence in the development of GNOME, because many of them belong to the group that was active in the first stages of the project. It would be of course worthwhile to design another sample including at least some of them. Those developers, in our opinion, are the best candidates to be hired for companies interested in the project, as they have been contributing since its beginning.

It is for sure also interesting to extend this study to other large, long-term libre software projects, such as KDE, Apache, etc, to find out whether the conclusions shown here are particular for the GNOME project or general for the libre software phenomenon.

Finally we think that companies can learn an important lesson from these results. When one is interested in contributing to a libre software project which is strategic for them, hired developers can gain enough knowledge of the project as to begin to contribute in a short time, even if they did not have a previous contact with it.


7. ACKNOWLEDGMENTS

We thank Carlos Perelló, who is part of the GNOME community, for his assistance and suggestions.

8. REFERENCES

- [1] K. Crowston and J. Howison. The social structure of open source software development teams. In *Proceedings of the International Conference on Information Systems*, Seattle, WA, USA, 2003.
- [2] T. Dinh-Trong and J. M. Bieman. Open source software development: A case study of FreeBSD. In *Proceedings of the 10th International Software Metrics Symposium*, Chicago, IL, USA, 2004.
- [3] D. German. The GNOME project: a case study of open source, global software development. *Journal of Software Process: Improvement and Practice*, 8(4):201–215, 2004.
- [4] 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.
- [5] J. M. Gonzalez-Barahona and G. Robles. Unmounting the "code gods" assumption. Technical report, Universidad Rey Juan Carlos, 2003. <http://libresoft.urjc.es/html/downloads/xp2003-barahona-robles.pdf>.
- [6] C. Jensen and W. Scacchi. Modeling recruitment and role migration processes in OSSD projects. In *Proceedings of 6th International Workshop on Software Process Simulation and Modeling*, St. Louis, May 2005.
- [7] S. Koch and G. Schneider. Effort, cooperation and coordination in an open source software project: Gnome. *Information Systems Journal*, 12(1):27–42, 2002.
- [8] 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.
- [9] G. Robles and J. M. Gonzalez-Barahona. Developer identification methods for integrated data from various sources. In *Proceedings of the International Workshop on Mining Software Repositories*, St. Louis, Missouri, USA, May 2005.
- [10] 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.
- [11] G. von Krogh, S. Spaeth, and K. R. Lakhani. Community, joining, and specialization in open source software innovation: A case study. *MIT Sloan Working Paper No. 4413-03*, 2003.
- [12] Y. Ye, K. Nakakoji, Y. Yamamoto, and K. Kishida. The co-evolution of systems and communities in free and open source software development. In S. Koch, editor, *Free/Open Source Software Development*, pages 59–82. Idea Group Publishing, Hershey, PA, USA, 2004.

Web Images More... jgbarah@gmail.com



Scholar ← Edit 🗑 Export ▾



Jesus M. Gonzalez-Barahona

The processes of joining in global distributed software projects

[\[PDF\] from researchgate.net](#)
[Texto completo para UC3M](#)

Authors: Israel Herraiz, Gregorio Robles, Juan José Amor, Teófilo Romera, Jesús M González Barahona
 Publication date: 2006/5/23
 Conference: Proceedings of the 2006 international workshop on Global software development for the practitioner
 Pages: 27-33
 Publisher: ACM

Description: Abstract Libre (free/open source) software is a good example of global software development. Thousands of projects, in a wide range of domains which involve hundreds of thousands of developers and contributors from all around the world. Some large (both in size and developer community) libre software projects have shown evidence of producing code with complete functionality and fast evolution (with linear or superlinear growth), while maintaining low defect density. Many companies are exploring how to benefit from this ...

Total citations [Cited by 49](#)



Scholar articles [The processes of joining in global distributed software projects](#)
 I Herraiz, G Robles, JJ Amor, T Romera... - Proceedings of the 2006 international workshop on ..., 2006
[Cited by 49](#) - [Related articles](#) - [All 11 versions](#)

Dates and citation counts are estimated and are determined automatically by a computer program.

[Help](#) [Privacy](#) [Terms](#) [Provide feedback](#) [My Citations](#)

Geographic Location of Developers at SourceForge*

Gregorio Robles
grex@gsync.escet.urjc.es

Jesus M. Gonzalez-Barahona
jgb@gsync.escet.urjc.es

Grupo de Sistemas y Comunicaciones
Universidad Rey Juan Carlos
Mostoles, Spain

ABSTRACT

The development of libre (free/open source) software is usually performed by geographically distributed teams. Participation in most cases is voluntary, sometimes sporadic, and often not framed by a pre-defined management structure. This means that anybody can contribute, and in principle no national origin has advantages over others, except for the differences in availability and quality of Internet connections and language. However, differences in participation across regions do exist, although there are little studies about them. In this paper we present some data which can be the basis for some of those studies. We have taken the database of users registered at SourceForge, the largest libre software development web-based platform, and have inferred their geographical locations. For this, we have applied several techniques and heuristics on the available data (mainly e-mail addresses and time zones), which are presented and discussed in detail. The results show a snapshot of the regional distribution of SourceForge users, which may be a good proxy of the actual distribution of libre software developers. In addition, the methodology may be of interest for similar studies in other domains, when the available data is similar (as is the case of mailing lists related to software projects).

Categories and Subject Descriptors

D.2.m [Software Engineering]: Miscellaneous

General Terms

Human Factors

*This work has been funded in part by the European Commission, under the CALIBRE CA, IST program, contract number 004337 and under the FLOSS-World SA, IST program, contract number 015722. This work is based on the SourceForge database provided by University of Notre Dame, see details at <http://www.nd.edu/oss/Data/data.html>.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

MSR'06, May 22–23, 2006, Shanghai, China.

Copyright 2006 ACM 1-59593-085-X/06/0005 ...\$5.00.

Keywords

Geographical location, mining software repositories, libre software, free software, open source software

1. INTRODUCTION

One of the most well known characteristics of libre (free, open source) software¹ is the worldwide distributed pool of developers that collaborate in tens of thousands of projects, using Internet-based tools for coordination. These projects are usually open to participation by anyone, from any corner of the globe, provided Internet access is granted; those with enough knowledge and skills can, in principle, join them. This openness, and the underlying informality, has resulted in an environment where participation is difficult to control, or even understand. One of the most significative examples of open issues in this respect is the geographical distribution of the aforementioned pool of developers. The answer to the question “where do developers live?” is not only interesting for academic reasons; it is also important from both strategic and economic points of view.

In this paper, we present a first approach to deal with this question by analyzing data about a huge sample of developers. We describe how we have mined the database of the largest libre software development supporting platform (SourceForge) looking for indicators to estimate the geographic location of the developers registered in it. Since the number of users of the SourceForge platform is well over one million, we can assume it is a reasonably good and representative proxy of the whole population of libre software developers (although for sure it presents some bias, as will be discussed later, for instance in terms of language knowledge).

The main goals of this paper are two: to show a methodology to estimate country of residence (as a simple quantifier of geographic location) using the indicators available in the SourceForge database, and to obtain a first estimation of the location of libre software developers.

With respect to the first goal, it is noteworthy to mention that SourceForge does not store specific information about the geographical location of developers, which therefore has to be inferred from other indicators, such as the domains in the e-mail address, or the time zone information developers introduce when registering at SourceForge. We believe that

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

many cases statistical relationships to infer the proportion of nationals of a certain country in a population of users with some characteristics. This is certainly a limitation of the proposed approach, specially if we were interested in (individual) developer identification methods as proposed in other works [10].

A future line of research could be to relate our findings with the activity of developers in the projects they are involved. This could be done by tracking developers in control versioning systems, mailing lists, forums, etc., and studying their activity by national origin. This could be an important issue, since previous research has shown that activity in libre software tends to be highly skewed towards a minority group responsible for the vast majority of the work performed. The authors of this work have started to analyze the CVS versioning system logs of all the SourceForge projects with the CVSanaly tool [11], and the FLOSSMole project [1] has also information related to projects in the site. Both data sets could be used for this matter.

An interesting issue is how representative this study is of the whole population of libre software developers. SourceForge is not the only development platform: large libre software projects usually administrate their own infrastructure, and also many other SourceForge-like sites exist, in some cases linked to language or national communities. This means on one hand that we are not considering a lot of libre software which is being developed outside SourceForge (although many of the developers of that software are probably also users of this site), and on the other that the study could be skewed by ignoring some communities which are not represented in SourceForge, but in other facilities. Further studies should address this issue, and determine how good the SourceForge population is as a proxy of the developer population.

On a more socio-economic perspective, the findings presented in this paper could be related to other parameters characterizing the countries, looking for correlations which could explain the different quantities of developers, such as the GDP, the GDP per capita, Nielsen/Netratings, or other economic and technological parameters.

Especially interesting is also the issue of finding projects that are driven by local activity, i.e. projects whose contributors are from the same country, region or cultural environment. This could be a way of finding possible splits of the libre software community, and a first step towards identifying parameters leading to collaboration between developers. Cultural, language and other barriers should also be considered. In this sense, a recent change in the SourceForge platform has been the inclusion of a language field (although up to the moment less than 25% have specified a different language from the default 'English').

All of this could also be extended to a social network analysis, such as performed on libre software developers [8, 7, 9], but taking into account geographical information.

7. ACKNOWLEDGMENTS

We thank the SourceForge team, and Greg Madey from the University of Notre Dame, for providing access to the SourceForge data. Also, a big thank you goes to our colleagues from GSyC/LibreSoft for their help verifying the validity of the data.

8. REFERENCES

- [1] M. Conklin, J. Howison, and K. Crowston. Collaboration using OSSmole: A repository of FLOSS data and analyses. In *Proceedings of the International Workshop on Mining Software Repositories*, pages 126-130, St. Louis, Missouri, USA, May 2005.
- [2] P. A. David, A. Waterman, and S. Arora. FLOSS-US. The Free/Libre/Open Source Software Survey for 2003. *Technical report*, Stanford Institute for Economic and Policy Research, Stanford, USA, 2003.
- [3] B. J. Dempsey, D. Weiss, P. Jones, and J. Greenberg. A quantitative profile of a community of Open Source Linux developers. *Technical report*, October 1999.
- [4] R. A. Ghosh, R. Glott, B. Krieger, and G. Robles. Survey of developers (free/libre and open source software: Survey and study). *Technical report*, International Institute of Infonomics. University of Maastricht, The Netherlands, June 2002.
- [5] K. Healy and A. Schussman. The ecology of open-source software development. *Technical report*, University of Arizona, USA, Jan. 2003.
- [6] D. Lancashire. Code, culture and cash: The fading altruism of Open Source development. *First Monday*, 6(12), 2001.
- [7] L. Lopez, J. M. Gonzalez-Barahona, and G. Robles. Applying social network analysis to the information in CVS repositories. In *Proc Intl Workshop on Mining Software Repositories*, pages 101-105, Edinburg, UK, 2004.
- [8] G. Madey, V. Freeh, and R. Tynan. The open source development phenomenon: An analysis based on social network theory. In *Americas Conf on Information Systems*, pages 1806-1813, Dallas, TX, USA, 2002.
- [9] M. Ohira, N. Ohsugi, T. Ohoka, and K.-I. Matsumoto. Accelerating cross-project knowledge collaboration using collaborative filtering and social networks. In *Proceedings Intl Workshop on Mining Software Repositories*, St. Louis, Missouri, USA, May 2005.
- [10] G. Robles and J. M. Gonzalez-Barahona. Developer identification methods for integrated data from various sources. In *Proceedings of the International Workshop on Mining Software Repositories*, pages 106-110, St. Louis, Missouri, USA, May 2005.
- [11] G. Robles, S. Koch, and J. M. Gonzalez-Barahona. Remote analysis and measurement of libre software systems by means of the CVSanaly tool. In *Proc 2nd Workshop on Remote Analysis and Measurement of Software Systems*, pages 51-56, Edinburg, UK, 2004.
- [12] G. Robles, H. Scheider, I. Tretkowski, and N. Weber. Who is doing it? A research on libre software developers. *Technical report*, Technische Universitaet Berlin, Berlin, Germany, Aug. 2001.
- [13] I. Tuomi. Evolution of the Linux Credits file: Methodological challenges and reference data for Open Source research. *First Monday*, 9(6), 2004.

Web Images More... jgbarah@gmail.com

Google

Scholar

[←](#) [Edit](#) [🗑](#) [Export ▾](#)



Jesus M. Gonzalez-Barahona

Geographic location of developers at sourceforge

[\[PDF\] from irisa.fr](#)
[Texto completo para UC3M](#)

Authors: Gregorio Robles, Jesus M Gonzalez-Barahona
Publication date: 2006/5/22
Conference: Proceedings of the 2006 international workshop on Mining software repositories
Pages: 144-150
Publisher: ACM

Description: Abstract The development of libre (free/open source) software is usually performed by geographically distributed teams. Participation in most cases is voluntary, sometimes sporadic, and often not framed by a pre-defined management structure. This means that anybody can contribute, and in principle no national origin has advantages over others, except for the differences in availability and quality of Internet connections and language. However, differences in participation across regions do exist, although there are little ...

Total citations [Cited by 32](#)



Scholar articles [Geographic location of developers at sourceforge](#)
G Robles, JM Gonzalez-Barahona - Proceedings of the 2006 international workshop on ..., 2006
[Cited by 32](#) - [Related articles](#) - [All 9 versions](#)

Dates and citation counts are estimated and are determined automatically by a computer program.

[Help](#) [Privacy](#) [Terms](#) [Provide feedback](#) [My Citations](#)

Mining Large Software Compilations over Time: Another Perspective of Software Evolution*

Gregorio Robles, Jesus M. Gonzalez-Barahona
Universidad Rey Juan Carlos
{grex,jgb}@gsyc.escet.urjc.es

Martin Michlmayr
University of Cambridge
martin@michlmayr.org

Juan Jose Amor
Universidad Rey Juan Carlos
jjamor@gsyc.escet.urjc.es

ABSTRACT

With the success of libre (free, open source) software, a new type of software compilation has become increasingly common. Such compilations, often referred to as ‘distributions’, group hundreds, if not thousands, of software applications and libraries written by independent parties into an integrated system. Software compilations raise a number of questions that have not been targeted so far by software evolution, which usually focuses on the evolution of single applications. Undoubtedly, the challenges that software compilations face differ from those found in single software applications. Nevertheless, it can be assumed that both, the evolution of applications and that of software compilations, have similarities and dependencies.

In this sense, we identify a dichotomy, common to that in economics, of software evolution in the small (micro-evolution) and in the large (macro-evolution). The goal of this paper is to study the evolution of a large software compilation, mining the publicly available repository of a well-known Linux distribution, Debian. We will therefore investigate changes related to hundreds of millions of lines of code over seven years. The aspects that will be covered in this paper are size (in terms of number of packages and of number of lines of code), use of programming languages, maintenance of packages and file sizes.

Categories and Subject Descriptors

D.2.m [Software Engineering]: Distribution, Maintenance, and Enhancement

*The work of Gregorio Robles, Jesus M. Gonzalez-Barahona and Juan Jose Amor has been funded in part by the European Commission under the CALIBRE CA, IST program, contract number 004337. The work of Martin Michlmayr has been funded in part by Google, Intel and the EPSRC. We would also like to thank the anonymous reviewers for their extensive comments.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage, and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

MSR’06, May 22–23, 2006, Shanghai, China.

Copyright 2006 ACM 1-59593-085-X/06/0005 ...\$5.00.

General Terms

Measurement, languages

Keywords

Mining software repositories, large software collections, software evolution, software integrators

1. INTRODUCTION

Large systems based on libre software¹ are developed in a manner that is quite different to traditional systems. In traditional large systems, such as operating systems, most work is done in-house, with only few pieces licensed from other sources and little work contracted to other companies. Such work is also performed in close cooperation with the organization and under tightly defined requirements. Libre software, on the other hand, is typically written by small, independent teams of volunteers, sometimes collaborating with paid staff from one or more companies. While various projects interact with each other, in particular where dependencies between the software exist, there is no central coordination between the individual projects. The main task of vendors (i.e. distributions) of libre operating systems is therefore not to write software but to group existing software, taken from several sources, together and to make that collection easy to install, configure and administer.

Since users of libre software have no incentive to download software from hundreds of sites and installing them individually, distributions play an important role by providing an integrated system that is easy to install. Unsurprisingly, a number of companies have seen this as a business opportunity and offer such distributions among with related services, such as support. There are also a number of community projects which operate on a non-profit basis like other libre software projects. Given their open way of collaboration, these are a good target for in-depth study of extremely large software compilations. While some commercial entities have recently started their own community projects in addition to their enterprise offerings, most notably Fedora (Red Hat) and OpenSUSE (Novell), we will take Debian as the source

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

| Lang. | Deb. 2.0 | Deb. 2.1 | Deb. 2.2 | Deb. 3.0 | Deb. 3.1 |
|-------|----------|----------|----------|----------|----------|
| C | 262.88 | 268.42 | 268.64 | 283.33 | 276.36 |
| C++ | 142.50 | 158.62 | 169.22 | 184.22 | 186.65 |
| Lisp | 394.82 | 393.99 | 394.19 | 383.60 | 349.56 |
| shell | 98.65 | 116.06 | 163.66 | 288.75 | 338.25 |
| Yacc | 789.43 | 743.79 | 762.24 | 619.30 | 599.23 |
| Mean | 228.49 | 229.92 | 229.46 | 243.35 | 231.6 |

Table 5: Mean file size for some programming languages.

problems for the management of the future evolution of the system, something that has probably influenced the delays in the release process of the last stable versions.

A specific problem in this realm comes from the fact that until now the mean size of packages has remained almost constant, which means that the system has more and more packages (growing linearly with the size of the system in SLOCs). Since there is a certain level of complexity related to the specifics of each package, which imposes a limit on the number of packages per developer, this means that the project would need to grow in terms of developers at the same pace. However, such a growth is not easy, and causes problems of its own, specially in the area of coordination.

With respect to the absolute figures, it can be noted that Debian 3.1 is probably one of the largest coordinated software collections in history, and almost for sure the largest one in the domain of general-purpose software for desktops and servers. This means that the human team maintaining it, which has also the peculiarity of being completely formed by volunteers, is exploring the limits of how to assemble and coordinate such a huge quantity of software. Therefore, the techniques and processes they employ to maintain a certain level of quality, a reasonable speed of updating, and a release process that delivers stable versions quite usable, are worth studying, and can for sure be of use in other domains which have to deal with large, complex collections of software.

As far as the programming languages are concerned, C is the most used language, although it is gradually losing importance. Scripting languages, C++ and Java are those that seem to have a higher growth in the newer releases, whereas the traditional compiled languages have even inferior growth rates than C. These variations also imply that the Debian team has to include developers with new skills in programming languages in order to maintain the evolving proportions. By looking at the trends in languages use within the distribution, the project could estimate how many developers fluent in a given language it will need. In addition, this evolution of the different languages can also be considered as an estimation of how libre software is evolving in terms of languages used, although some of them are for sure misrepresented (for instance, Java is underrepresented, possibly because of licensing issues).

The evolution shown in this paper should also be put in the context of the activity of the volunteers doing all the packaging work. While some work has been done in this area [9], more research needs to be performed before a link can be established between the evolution of the skills and size of the developer population, the complexity and size of the distribution, the processes and activities performed by the project, and the quality of the resulting product. Only by understanding the relationships between all these

parameters can reasonable measures be proposed to improve the quality of the software distribution, or shorten the release cycle without harming reliability and stability of the releases.

All in all, the study of distributions such as Debian can be of great interest not only for understanding their evolution, but also to be used as good case studies which can help to understand large, complex software systems which are more and more common in many domains.

6. REFERENCES

- [1] J. J. Amor, J. M. Gonzalez-Barahona, G. Robles, and I. Herraiz. Measuring libre software using Debian 3.1 (Sarge) as a case study: preliminary results. *Upgrade Magazine*, Aug. 2005.
- [2] H. Gall, M. Jazayeri, R. Klosch, and G. Trausmuth. Software evolution observations based on product release history. In *Proc Intl Conference on Software Maintenance*, pages 160-170, 1997.
- [3] 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.
- [4] J. M. Gonzalez-Barahona, M. A. Ortuno Perez, P. de las Heras, J. Centeno, and V. Matellan. Counting potatoes: the size of Debian 2.2. *Upgrade Magazine*, II(6):60-66, Dec. 2001.
- [5] M. M. Lehman and L. A. Belady, editors. Program evolution: Processes of software change. *Academic Press Professional, Inc.*, San Diego, CA, USA, 1985.
- [6] M. M. Lehman and J. F. Ramil. Implications of laws of software evolution on continuing successful use of cots software. *Technical report*, Imperial College, 1998.
- [7] M. M. Lehman and J. F. Ramil. Rules and tools for software evolution planning and management. *Annals of Software Engineering*, 11(1):15-44, 2001.
- [8] M. M. Lehman, J. F. Ramil, P. D. Wernick, D. E. Perry, and W. M. Turski. Metrics and laws of software evolution - the nineties view. In *METRICS'97: Proceedings of the 4th International Symposium on Software Metrics*, page 20, nov 1997.
- [9] M. Michlmayr and B. M. Hill. Quality and the reliance on individuals in free software projects. In *Proceedings 3rd Workshop on Open Source Software Engineering*, pages 105-109, Portland, USA, 2003.
- [10] G. Robles, J. J. Amor, J. M. Gonzalez-Barahona, and I. Herraiz. Evolution and growth in large libre software projects. In *Proceedings of the International Workshop on Principles in Software Evolution*, pages 165-174, Lisbon, Portugal, September 2005.
- [11] G. Succi, J. W. Paulson, and A. Eberlein. Preliminary results from an empirical study on the growth of open source and commercial software products. In *EDSER-3 Workshop*, Toronto, Canada, May 2001.
- [12] E. B. Swanson. The dimensions of maintenance. In *Proceedings of the 2nd International conference on Software Engineering*, pages 492-497, 1976.
- [13] W. M. Turski. Reference model for smooth growth of software systems. *IEEE Transactions on Software Engineering*, 22(8):599-600, 1996.
- [14] D. A. Wheeler. More than a gigabuck: Estimating GNU/Linux's size. *Technical report*, June 2001.



MSR 2006

Mining Software Repositories

**BEST
PAPER
AWARD**

Presented to

Gregorio Robles, Jesus M. Gonzalez-Barahona
Universidad Rey Juan Carlos

Martin Michlmayr
University of Cambridge

Juan Jose Amor
Universidad Rey Juan Carlos

for their paper entitled


**“Mining Large Software Compilations over Time:
Another Perspective of Software Evolution”**



Stephan Diehl
Universität Trier, Germany
(on behalf of all workshop co-organizers)

The International Workshop on Mining Software Repositories was held on May 22-23, 2006 in Shanghai, China. It was co-located with the IEEE International Conference on Software Engineering ICSE'06.

Web Images More... jgbarah@gmail.com



Scholar

[←](#) [Edit](#) [🗑](#) [Export ▾](#)



Jesus M. Gonzalez-Barahona

Mining large software compilations over time: another perspective of software evolution

[\[PDF\] from irisa.fr](#)
[Texto completo para UC3M](#)

Authors: Gregorio Robles, Jesus M Gonzalez-Barahona, Martin Michlmayr, Juan Jose Amor
Publication date: 2006/5/22
Conference: Proceedings of the 2006 international workshop on Mining software repositories
Pages: 3-9
Publisher: ACM

Description: Abstract With the success of libre (free, open source) software, a new type of software compilation has become increasingly common. Such compilations, often referred to as 'distributions', group hundreds, if not thousands, of software applications and libraries written by independent parties into an integrated system. Software compilations raise a number of questions that have not been targeted so far by software evolution, which usually focuses on the evolution of single applications. Undoubtedly, the challenges that software ...

Total citations [Cited by 60](#)



Scholar articles [Mining large software compilations over time: another perspective of software evolution](#)
G Robles, JM Gonzalez-Barahona, M Michlmayr... - Proceedings of the 2006 international workshop on ..., 2006
[Cited by 60](#) - [Related articles](#) - [All 24 versions](#)

Dates and citation counts are estimated and are determined automatically by a computer program.

[Help](#) [Privacy](#) [Terms](#) [Provide feedback](#) [My Citations](#)

Contributor Turnover in Libre Software Projects

Gregorio Robles and Jesus M. Gonzalez-Barahona

GSyC/LibreSoft, Universidad Rey Juan Carlos, Spain
{grex,jgb}@gsyc.escet.urjc.es

Abstract. A common problem that management faces in software companies is the high instability of their staff. In libre (free, open source) software projects, the permanence of developers is also an open issue, with the potential of causing problems amplified by the self-organizing nature that most of them exhibit. Hence, human resources in libre software projects are even more difficult to manage: developers are in most cases not bound by a contract and, in addition, there is not a real management structure concerned about this problem. This raises some interesting questions with respect to the composition of development teams in libre software projects, and how they evolve over time. There are projects lead by their original founders (some sort of “code gods”), while others are driven by several different developer groups over time (i.e. the project “regenerates” itself). In this paper, we propose a quantitative methodology, based on the analysis of the activity in the source code management repositories, to study how these processes (developers leaving, developers joining) affect libre software projects. The basis of it is the analysis of the composition of the core group, the group of developers most active in a project, for several time lapses. We will apply this methodology to several large, well-known libre software projects, and show how it can be used to characterize them. In addition, we will discuss the lessons that can be learned, and the validity of our proposal.

Keywords: open source, human resources, turnover, mining software repositories

1 Introduction

Employee turnover (the ratio of the number of workers replaced in a given period to the average number of workers), is known to be high in the (proprietary) software industry [1]. In the libre software world¹, the study of turnover has not been a research target (at least to the knowledge of the authors) profusely. Most of the attention has been focused on the organizational structure of the

¹ In this paper we will use the term “libre software” to refer to any software licensed under terms compliant with the FSF definition of “free software”, and the OSI definition of “open source software”, thus avoiding the controversy between those two terms.

Please use the following format when citing this chapter:

Robles, G., and Gonzalez-Barahona, J.M., 2006, in IFIP International Federation for Information Processing, Volume 203, Open Source Systems, eds. Damiani, E., Fitzgerald, B., Scacchi, W., Scotto, M., Succi, G., (Boston: Springer), pp. 273-286

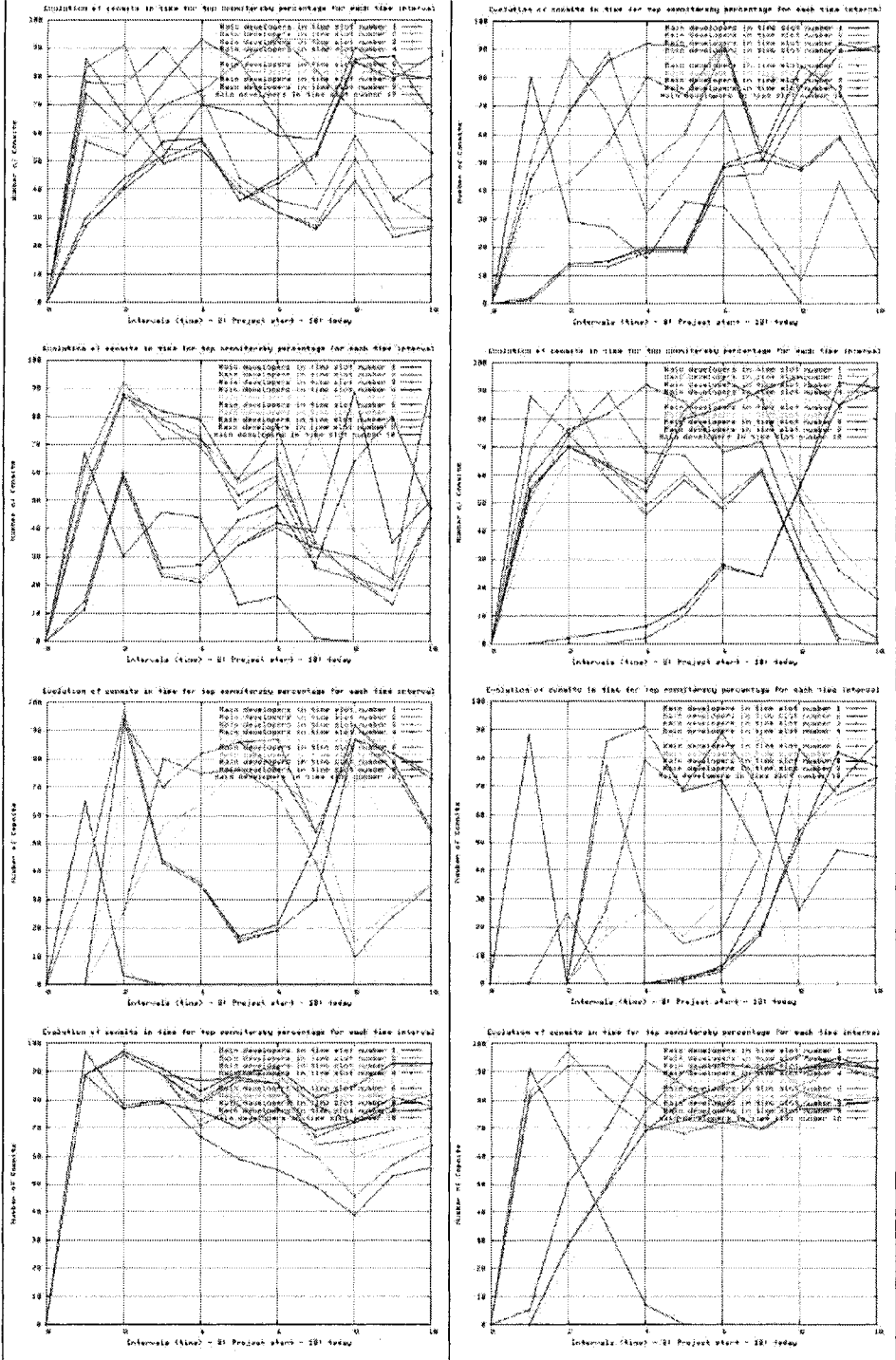


Table 6. 2x4 matrix with fractional generation plots for 8 libre software systems. Projects with heavy generational turn-over have been situated at the top. More information can be found in table 4.



Scholar



Jesus M. Gonzalez-Barahona

Contributor turnover in libre software projects

[PDF] from flosshub.org

Authors: Gregorio Robles, Jesus M Gonzalez-Barahona
 Publication date: 2006/1/1
 Book: Open Source Systems
 Pages: 273-286
 Publisher: Springer US

Description Abstract A common problem that management faces in software companies is the high instability of their staff. In libre (free, open source) software projects, the permanence of developers is also an open issue, with the potential of causing problems amplified by the self-organizing nature that most of them exhibit. Hence, human resources in libre software projects are even more difficult to manage: developers are in most cases not bound by a contract and, in addition, there is not a real management structure concerned about this ...

Total citations [Cited by 59](#)



Scholar articles [Contributor turnover in libre software projects](#)
 G Robles, JM Gonzalez-Barahona - Open Source Systems, 2006
[Cited by 59](#) - [Related articles](#) - [All 8 versions](#)

[Contributor Turnover in Libre Software Projects](#) ★
 JM Gonzalez-Barahona, G Robles - IFIP International Federation for Information Processing, 2010

[Contributor Turnover in Libre Software](#) ★
 G Robles, JM Gonzalez-Barahona

Dates and citation counts are estimated and are determined automatically by a computer program.

On the prediction of the evolution of libre software projects*

Israel Herraiz, Jesus M. Gonzalez-Barahona, Gregorio Robles
GSyC/LibreSoft - Libre Software Engineering Lab
Universidad Rey Juan Carlos (Spain)
{herraiz, jgb, grex}@gsec.es

Daniel M. German
Department of Computer Science
University of Victoria (Canada)
dmg@uvic.ca

Abstract

Libre (free / open source) software development is a complex phenomenon. Many actors (core developers, casual contributors, bug reporters, patch submitters, users, etc.), in many cases volunteers, interact in complex patterns without the constraints of formal hierarchical structures or organizational ties. Understanding this complex behavior with enough detail to build explanatory models suitable for prediction is an open challenge, and few results have been published to date in this area. Therefore statistical, non-explanatory models (such as the traditional regression model) have a clear role, and have been used in some evolution studies. Our proposal goes in this direction, but using a model that we have found more useful: time series analysis. Data available from the source code management repository is used to compute the size of the software over its past life, using this information to estimate the future evolution of the project. In this paper we present this methodology and apply it to three large projects, showing how in these cases predictions are more accurate than regression models, and precise enough to estimate with little error their near future evolutions.

1 Introduction

Libre software¹ projects are usually based on a community of many different actors, ranging from core developers to casual contributors, and even users (who may contribute for instance with bug reports). Most of them behave according to their own interests, in many cases on a volunteer basis. The community formed by those actors shows some

*This work has been funded in part by the European Commission, under the FLOSSMETRICS (FP6-IST-5-033982) and QUALOSS (FP6-IST-5-033547) projects. Israel Herraiz has been funded in part by Consejería de Educación de Comunidad de Madrid and European Social Fund, under grant number 01/FPI/0582/2005.

¹In this paper we will use the term “libre software” to refer both to “free software” (as defined by the Free Software Foundation) and “open source software” (as defined by the Open Source Initiative).

structure [8, 22] with different levels of involvement and expertise that may change over time. The management of the project is usually distributed (to some extent), and decisions are difficult to impose, since no formal organizational links or hierarchies are recognized by all the actors. Therefore, developers can not be compelled to do some tasks if they do not want to (even if those tasks are urgent): they may be involved in non fundamental activities, while not contributing to some others with higher priority for the project. There is also usually a lack of predefined requirements, undetailed designs and absence of interprocess documentation [27]. Together with volunteer contributors, some others hired by companies can also be present, in some cases with their own agenda, which usually complicates the picture even more.

In many cases, and despite the lack of apparent plans, these forces and interests result in reliable and mature software which satisfies the needs of many users. Many libre software projects grow continuously over time, and they seem to satisfy most of the requirements of their users. Well-known examples are the Apache web server, the Linux kernel or the Mozilla Firefox web browser.

The problems for forecasting the evolution of the products produced by those projects are clear. If forecasting is already a risky business in traditional software development, where the environment is more constrained, it is even more difficult in these more complex scenarios. However, having a predictive model is undoubtedly a fundamental tool for those interested in the future evolution of those products.

There are several studies proposing models for how certain aspects of libre software products evolve, in some cases including prediction capabilities [3, 10, 26]. However, they fail to provide a comprehensive view of the evolution, probably due to the inherent complexity of the phenomenon and the many different interactions among involved actors. Each individual decision, even when based on self-interest, may seem to the observer as a random event. Therefore, if we focus only on the “low-level” interactions of the community, it is difficult to find meaningful results for the global landscape.

- [4] G. Antoniol, G. Casazza, M. D. Penta, and E. Merlo. Modeling clones evolution through time series. In *Proceedings of the International Conference on Software Maintenance*, 2001.
- [5] B. B. Boehm. *Software Engineering Economics*. Prentice Hall, 1981.
- [6] F. Caprio, G. Casazza, M. D. Penta, and U. Villano. Measuring and predicting the Linux kernel evolution. In *Proceedings of the International Workshop of Empirical Studies on Software Maintenance*, Florence, Italy, 2001.
- [7] S. D. Conte. *Software Engineering Metrics and Models (Benjamin/Cummings series in software engineering)*. Benjamin-Cummings Pub Co, 1986.
- [8] K. Crowston and J. Howison. The social structure of free and open source software development. *First Monday*, 10(2), February 2005.
http://www.firstmonday.dk/issues/issue10_2/crowston/.
- [9] J.-M. Dalle, L. Daudet, and M. den Besten. Mining cvs signals. In *Proceedings of the Workshop on Public Data about Software Development*, pages 12–21, Como, Italy, 2006.
- [10] J.-M. Dalle and P. A. David. The allocation of software development resources in Open Source production mode. Technical report, SIEPR Policy paper No. 02-027, SIEPR, Stanford, USA, 2003.
<http://siepr.stanford.edu/papers/pdf/02-27.pdf>.
- [11] F.-W. Duijnhouwer and C. Widdows. Open source maturity model. Technical Report 1.5.3, Capgemini, August 2003.
- [12] A. R. Fasolino, D. Natale, A. Poli, and A. Alberigi-Quaranta. Metrics in the development and maintenance of software: an application in a large scale environment. *Journal of Software Maintenance: Research and Practice*, 12:343–355, 2000.
- [13] E. Fuentetaja and D. J. Bagert. Software Evolution from a Time-Series perspective. In *Proceedings of the International Conference on Software Maintenance*, pages 226–229, 2002.
- [14] D. M. German and A. Hindle. Visualizing the evolution of software using softChange. *International Journal of Software Engineering and Knowledge Engineering*, 16(1):5–21, 2006.
- [15] M. W. Godfrey and Q. Tu. Evolution in open source software: A case study. In *ICSM '00: Proceedings of the International Conference on Software Maintenance (ICSM'00)*, pages 131–142, Washington, DC, USA, October 2000. IEEE Computer Society.
- [16] A. E. Hassan, J. Wu, and R. C. Holt. Visualizing historical data using spectrographs. In *Proceedings of the International Software Metrics Symposium*, Como, Italy, 2005.
- [17] C. F. Kemerer and S. Slaughter. An empirical approach to studying software evolution. *IEEE Transactions on Software Engineering*, 25(4):493–509, 1999.
- [18] S. Koch. Evolution of Open Source Software systems - a large-scale investigation. In *Proceedings of the 1st International Conference on Open Source Systems*, Genova, Italy, July 2005.
- [19] M. M. Lehman and L. A. Belady, editors. *Program Evolution. Processes of Software Change*. Academic Press Inc., 1985.
- [20] M. M. Lehman, J. F. Ramil, P. D. Wernick, D. E. Perry, and W. M. Turski. Metrics and laws of software evolution - the nineties view. In *METRICS '97: Proceedings of the 4th International Symposium on Software Metrics*, page 20, nov 1997.
- [21] S. G. Makridakis, S. C. Wheelwright, and R. J. Hyndman. *Forecasting: Methods and Applications*. John Wiley & Sons, Ltd., January 1998.
- [22] 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.
- [23] Y. Peng, F. Li, and A. Mili. Modeling the evolution of operating systems: An empirical study. *The Journal of Systems and Software*, 80(1):1–15, 2007.
- [24] G. Robles, J. J. Amor, J. M. Gonzalez-Barahona, and I. Her-raiz. Evolution and growth in large libre software projects. In *Proceedings of the International Workshop on Principles in Software Evolution*, pages 165–174, Lisbon, Portugal, September 2005.
- [25] G. Robles, J. M. Gonzalez-Barahona, and J.-J. Merelo. Beyond executable source code: The importance of other source artifacts in software development (a case study). *Journal of Systems and Software*, 79(9):1233–1248, September 2006.
- [26] G. Robles, J. J. Merelo, and J. M. Gonzalez-Barahona. Self-organized development in libre software: a model based on the stigmergy concept. In *Proceedings of the 6th International Workshop on Software Process Simulation and Modeling (ProSim 2005)*, St.Louis, Missouri, USA, May 2005.
- [27] W. Scacchi. Free and Open Source development practices in the game community. *IEEE Software*, 21(1):59–66, 2004.
- [28] R. H. Shumway and D. S. Stoffer. *Time Series Analysis and Applications. With R Examples*. Springer Texts in Statistics. Springer, 2006.
- [29] W. M. Turski. Reference model for smooth growth of software systems. *IEEE Transactions on Software Engineering*, 22(8):599–600, 1996.
- [30] W. M. Turski. The reference model for smooth growth of software systems revisited. *IEEE Transactions on Software Engineering*, 28(8):814–815, 2002.
- [31] C. C. H. Yuen. An empirical approach to the study of errors in large software under maintenance. In *Proceedings of the International Conference on Software Maintenance*, 1985.
- [32] C. C. H. Yuen. A statistical rationale for evolution dynamics concepts. In *Proceedings of the International Conference on Software Maintenance*, 1987.
- [33] C. C. H. Yuen. On analyzing maintenance process data at the global and detailed levels. In *Proceedings of the International Conference on Software Maintenance*, pages 248–255, 1988.



Scholar



Jesus M. Gonzalez-Barahona

On the prediction of the evolution of libre software projects

[\[PDF\] from turingmachine.org](#)
[Texto completo para UC3M](#)

Authors: Israel Herraiz, Jesus M Gonzalez-Barahona, Gregorio Robles, Daniel M German
 Publication date: 2007/10/2
 Conference: Software Maintenance, 2007. ICSM 2007. IEEE International Conference on
 Pages: 405-414
 Publisher: IEEE

Description Abstract Libre (free/open source) software development is a complex phenomenon. Many actors (core developers, casual contributors, bug reporters, patch submitters, users, etc.), in many cases volunteers, interact in complex patterns without the constrains of formal hierarchical structures or organizational ties. Understanding this complex behavior with enough detail to build explanatory models suitable for prediction is an open challenge, and few results have been published to date in this area. Therefore statistical, non-explanatory ...

Total citations [Cited by 27](#)



Scholar articles [On the prediction of the evolution of libre software projects](#)
 I Herraiz, JM Gonzalez-Barahona, G Robles... - Software Maintenance, 2007. ICSM 2007. IEEE ..., 2007
[Cited by 27](#) - [Related articles](#) - [All 7 versions](#)

Dates and citation counts are estimated and are determined automatically by a computer program.

[Help](#) [Privacy](#) [Terms](#) [Provide feedback](#) [My Citations](#)

Adapting the “Staged Model for Software Evolution” to Free/Libre/Open Source Software

Andrea Capiluppi
Centre of Research on Open Source Software
University of Lincoln, UK
acapiluppi@lincoln.ac.uk

Jesús M. González-Barahona, Israel
Herraiz, Gregorio Robles
GsyC/LibreSoft
Universidad Rey Juan Carlos – Madrid, Spain
{jgb,herraiz,robles}@gsync.es

ABSTRACT

Research into traditional software evolution has been tackled from two broad perspectives: that focused on the *how*, which looks at the processes, methods and techniques to implement and evolve software; and that focused on the *what/why* perspective, aiming at achieving an understanding of the drivers and general characteristics of the software evolution phenomenon.

The two perspectives are related in various ways: the study of the *what/why* is for instance essential to achieve an appropriate management of software engineering activities, and to guide innovation in processes, methods and tools, that is, the *how*. The output of the *what/why* studies is exemplified by empirical hypotheses, such as the staged model of software evolution.

This paper focuses on the commonalities and differences between the evolution and patterns in the lifecycles of traditional commercial systems and free/libre/open source software (FLOSS) systems. The existing staged model for software evolution is therefore revised for its applicability on FLOSS systems.

1. INTRODUCTION

The phenomenon of software evolution has been described in the literature (e.g., [10, 11]), with several models of different nature ([1, 2, 12, 16]) being proposed to understand and explain the empirical observations. Some of these models purport to be universally applicable to all software development processes. However, the models in the literature were built mainly observing software developed using a traditional centrally-managed waterfall development process, or one of its variants [20].

Research in this area has been approached from two different perspectives. One considers the processes, methods and techniques to implement and evolve software (the *how*). The other applies systematic observation of empirical data to achieve an understanding of the causes and general characteristics of the phenomenon (the *what/why*). Both per-

spectives are related: the study of the *what/why* is important in order to achieve and appropriate plan, manage and control the various software engineering activities; and to guide the development of new processes, methods and tools, that is, to guide the *how*.

The link between the *how* and the *what/why* perspectives is illustrated in [13], where several guidelines are derived from Lehman’s laws of software evolution. The output of the *what/why* study is exemplified by empirical generalizations such as Rajlich and Bennett’s model of the lifecycle [16] and Lehman’s laws of software evolution ([11, 15]).

In this context, the present paper expands and refines the empirical hypothesis presented in the staged model of software evolution [16] so that it can be applied to FLOSS projects. For this, we compare and contrast the existing empirical knowledge (e.g. as derived from studies of proprietary systems evolved under traditional processes [9]) with the emergent FLOSS paradigm. This revision will help FLOSS developers and practitioners to characterize any FLOSS system, in terms of which phase it is currently undergoing, or which phase it will more likely move to.

2. THE STAGED MODEL

The *staged model for software evolution* represents the software lifecycle as a sequence of steps [16]. Based on the traditional commercial projects, the core idea of the model is that software systems evolve through distinct stages: - Initial development, or *alpha stage*, includes all the phases (design, first coding, testing) achieved before the first running version of the system. In this stage, usually no releases are made public to the users.

- Evolutionary pressures enhance the system with new features and capabilities in the phase of the *evolution changes*: binary releases and individual patches are made available to the users, and feedback is gathered to further enhance the system.

- As long as the profitability of either new enhancements or changes to the existing code base is overcome by the costs of such modifications, the *servicing phase* is reached. The system is considered mature, changes are added to the code base, but no further enhancements (apart from patches) are provided to the end users.

- When the service is discontinued and no more code patches are released, the stage of *phase-out* is meant to declare the system’s end. This can be associated with the presence of a new enhanced system substituting the old one.

- The old system serves as a basis for the new one and then it is *closed down*.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

IWPSE’07 September 3-4, 2007, Dubrovnik, Croatia
Copyright 2007 ACM 978-1-59593-722-3/07/09 ...\$5.00.


cle may be considered different from traditional commercial system. The staged model for the software evolution, as in its original form expressed in [16] model, was discussed. A general resemblance between commercial and FLOSS evolutionary behavior was recognized: initial development tend to be superlinear or at least with sustained growth (see for instance Figure 1). A stabilization point where fewer functionalities were added has been recognized in some FLOSS evolutionary behavior (central part of Figure 1). Apart from the commonalities, three points of difference were detected for enhancing the staged model.

The first is relative to availability of releases: commercial companies make software systems available to third parties only when they are running and are tested enough. On the contrary, FLOSS systems are available in versioning system repositories well before first official release, and may be downloaded at any time. The second difference is relative to the transition between the evolution stage and the servicing stage: we encountered several cases in which a new development stage was achieved after a phase without major enhancements. The third revision made to the model is a possible transition between the phases of phase out and evolution: a case was illustrated in which a new development team took over the responsibility of a project that was declared closed (Grace). More in general, generations of developers have been identified in several FLOSS systems, where the most active developers (in terms of commits) get replaced frequently along the lifecycle of a FLOSS application. Therefore, it may be concluded that after some modifications, the original staged model for software evolution can be extended to consider the evolution of a FLOSS project.

5. REFERENCES

- [1] M. Aoyama. Metrics and analysis of software architecture evolution with discontinuity. In *IWPSE '02: Proceedings of the International Workshop on Principles of Software Evolution*, pages 103–107, New York, NY, USA, 2002. ACM Press.
- [2] A. Capiluppi. Models for the evolution of os projects. In *ICSM '03: Proceedings of the International Conference on Software Maintenance*, page 65, Washington, DC, USA, 2003. IEEE Computer Society.
- [3] A. Capiluppi, J. Fernandez-Ramil, J. Higman, H. C. Sharp, and N. Smith. An empirical study of the evolution of an agile-developed software system. In *ICSE '07: Proceedings of the 29th International Conference on Software Engineering*, pages 511–518, Washington, DC, USA, 2007. IEEE Computer Society.
- [4] A. Capiluppi and M. Michlmayr. From the cathedral to the bazaar: An empirical study of the lifecycle of volunteer community projects. In J. Feller, B. Fitzgerald, W. Scacchi, and A. Silitti, editors, *Open Source Development, Adoption and Innovation*, pages 31–44. International Federation for Information Processing, Springer, 2007.
- [5] N. Chapin, J. E. Hale, J. F., Ramil, and W.-G. Tan. Types of software evolution and software maintenance. *Journal of Software Maintenance and Evolution: Research and Practice*, 13(1):3–30, 2001.
- [6] R. English and C. Schweik. Identifying success and tragedy of floss commons: A preliminary classification of sourceforge.net projects. In *Proceedings of the 1st International Workshop on Emerging Trends in FLOSS Research and Development*, Minneapolis, MN, 2007. ICSE.
- [7] D. M. German and A. Mockus. Automating the measurement of open source projects. In *Proceedings of the 3rd Workshop on Open Source Software Engineering*, Portland, Oregon, USA, 2003.
- [8] M. W. Godfrey and Q. Tu. Evolution in open source software: A case study. In *ICSM '00: Proceedings of the International Conference on Software Maintenance (ICSM'00)*, page 131, Washington, DC, USA, 2000. IEEE Computer Society.
- [9] M. Lehman and J. Ramil. Feast: Feedback evolution and software technology.
- [10] M. M. Lehman. Programs, cities, students, limits to growth? In D. Gries, editor, *Programming Methodology*, pages 42–62. 1978.
- [11] M. M. Lehman and L. A. Belady, editors. *Program evolution: processes of software change*. Academic Press Professional, Inc., San Diego, CA, USA, 1985.
- [12] M. M. Lehman, G. Kahen, and J. F. Ramil. Behavioural modelling of long-lived evolution processes: some issues and an example. *Journal of Software Maintenance*, 14(5):335–351, 2002.
- [13] M. M. Lehman and J. F. Ramil. Rules and tools for software evolution planning and management. *Ann. Softw. Eng.*, 11(1):15–44, 2001.
- [14] M. Michlmayr. *Quality Improvement in Volunteer Free and Open Source Software Projects – Exploring the Impact of Release Management*. PhD thesis, University of Cambridge, UK, 2007.
- [15] S. L. Pfleeger. *Software Engineering: Theory and Practice*. Prentice Hall PTR, Upper Saddle River, NJ, USA, 2001.
- [16] V. T. Rajlich and K. H. Bennett. A staged model for the software life cycle. *Computer*, 33(7):66–71, 2000.
- [17] E. S. Raymond. *The cathedral and the bazaar: musings on Linux and open source by an accidental revolutionary*. O'Reilly & Associates, Inc., Sebastopol, CA, USA, 2001.
- [18] G. Robles, J. J. Amor, J. M. González-Barahona, and I. Herraiz. Evolution and growth in large libre software projects. In *IWPSE*, pages 165–174. IEEE Computer Society, 2005.
- [19] G. Robles and J. M. González-Barahona. Contributor turnover in libre software projects. In E. Damiani, B. Fitzgerald, W. Scacchi, M. Scotto, and G. Succi, editors, *OSS*, volume 203 of *IFIP*, pages 273–286. Springer, 2006.
- [20] W. W. Royce. Managing the development of large software systems: concepts and techniques. In *ICSE '87: Proceedings of the 9th international conference on Software Engineering*, pages 328–338, Los Alamitos, CA, USA, 1987. IEEE Computer Society Press.
- [21] A. Senyard and M. Michlmayr. How to have a successful free software project. In *Proceedings of the 11th Asia-Pacific Software Engineering Conference*, pages 84–91, Busan, Korea, 2004.
- [22] W. M. Turski. Reference model for smooth growth of software systems. *IEEE Trans. Softw. Eng.*, 22(8):599–600, 1996.

Web Images More... jgbarah@gmail.com



Scholar ← Edit 🗑️ Export ▾



Jesus M. Gonzalez-Barahona

Adapting the staged model for software evolution to free/libre/open source software [PDF] from open.ac.uk

Authors: Andrea Capiluppi, Jesús M González-Barahona, Israel Herraiz, Gregorio Robles
 Publication date: 2007/9/3
 Conference: Ninth international workshop on Principles of software evolution: in conjunction with the 6th ESEC/FSE joint meeting
 Pages: 79-82
 Publisher: ACM

Description: Abstract Research into traditional software evolution has been tackled from two broad perspectives: that focused on the how, which looks at the processes, methods and techniques to implement and evolve software; and that focused on the what/why perspective, aiming at achieving an understanding of the drivers and general characteristics of the software evolution phenomenon.

Total citations: [Cited by 31](#)



Scholar articles: [Adapting the staged model for software evolution to free/libre/open source software](#)
 A Capiluppi, JM González-Barahona, I Herraiz... - Ninth international workshop on Principles of software ..., 2007
[Cited by 31 - Related articles - All 8 versions](#)

[Adapting the "Staged Model for Software Evolution" to FLOSS](#) ★
 A Capiluppi, JM Gonzalez, I Herraiz, G Robles
[Related articles](#)

Dates and citation counts are estimated and are determined automatically by a computer program.

[Help](#) [Privacy](#) [Terms](#) [Provide feedback](#) [My Citations](#)

Forecasting the number of changes in Eclipse using time series analysis*

Israel Herraiz, Jesus M. Gonzalez-Barahona, Gregorio Robles
Grupo de Sistemas y Comunicaciones
Universidad Rey Juan Carlos, Spain
{herraiz, jgb, grex}@gsyc.escet.urjc.es
Libresoft team

Abstract

In order to predict the number of changes in the following months for the project Eclipse, we have applied a statistical (non-explanatory) model based on time series analysis. We have obtained the monthly number of changes in the CVS repository of Eclipse, using the CVSanaly tool. The input to our model was the filtered series of the number of changes per month, and the output was the number of changes per month for the next three months. Then we aggregated the results of the three months to obtain the total number of changes in the given period in the challenge.

1 Introduction

There have been some cases of proposals of predictive models for libre (free / open source) software projects. In our opinion, the phenomenon of libre software development is quite complex as to obtain a satisfactory explanatory model. For instance, in spite of the proposed models in the literature [9, 4, 1], little empirical validation of these models have been done, so failing on the prediction of the actual evolution of libre software projects.

Using the “low-level” approach taken by the mentioned papers is a difficult task, because the events that happen within a project are random-like. All the interactions (a change made to the source code, a new message to the mailing list, a new developer coming to the project, a developer leaving the project) that we may find in a project can not be predicted, because they involved people. However, if we look at the macroscopic level, the aggregation of all these random-like interactions is not random, and despite contain-

*This work has been funded in part by the European Commission, under the FLOSSMETRICS (FP6-IST-5-033547) and QUALOSS (FP6-IST-5-033547) projects. Israel Herraiz has been funded in part by Consejería de Educación of Comunidad de Madrid and European Social Fund, under grant number 01/FPI/0582/2005.

ing noise, can be predicted by means of statistical methods. Think for example of the stock market. It is really difficult to predict how individual actors will behave, because of the many factors that may impact their actions. But when the global stock market is considered, several statistical methods may be used to predict the near future in absence of impacting external events.

The idea of using time series analysis to predict software is not new. Already in the period from 1985 to 1988 several papers [11, 12, 13] used statistical methods, including time series analysis, to model software evolution. For instance, in [13] time series ARIMA models are used to predict the evolution in the maintenance phase of a software project, using sampling periods of about one month.

Later, Kemerer and Slaughter [6] followed this line of research proposing an ARIMA model which is able to predict the monthly number of changes of a software project. However, they did not obtain very good results, because the phenomenon studied (monthly number of changes, the same than in the challenge) is quite noisy. To avoid the problems found by Kemerer and Slaughter, we applied *kernel smoothing* in the hope of reducing the amount of noise in the source data.

There have been several other research papers using time series methods to predict the evolution of software projects. Because of the space restrictions, we just cite them here [2, 3, 5].

2 Methodology

We obtained the monthly number of changes for Eclipse, classified by plugins (using the mapping table provided for the challenge), using the tool CVSanaly [8]. From the database created by CVSanaly, we mined the revisions that were not deletions, and added up all the revisions in each one of the months, from the beginning of the history in the CVS until the last of January 2007.

Therefore, we obtained a list with the number of changes for every month since the beginning of the history of each

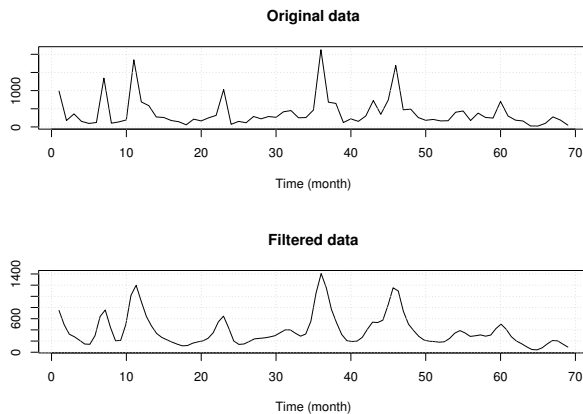


Figure 1. Original data and filtered data for the case of org.eclipse.core. Kernel smoothing with a bandwidth of 2.

plugin in the CVS. The data was actually obtained for each one of the subdirectories in the CVS. We mapped subdirectories to plugins adding up the changes for every subdirectory being part of the same plugin, in the hope of reducing some noise by aggregation. We needed to filter the data, though. We filtered this data using *kernel smoothing* with a bandwidth of 2, and the filtered series was the input for the ARIMA model. Figure 1 shows the original data and the filtered data for the plugin `org.eclipse.core`. We have used [7] as a guide. That book includes some examples and charts to select the right model based on the values of some parameters that we describe below.

The ARIMA model has three parameters:

- d , which is the number of differences needed to make the data stationary. In our case it was $d = 1$ for all the plugins.
- p , which is the *auto-regressive* part of the model. The right value is obtained by inspecting the autocorrelation coefficients and partial autocorrelation coefficients plots. In all the plugins, it was value was between 2 and 3.
- q , which is the *moving average* part of the model. Again, it is obtained by inspecting the autocorrelation coefficients and partial autocorrelation coefficients plots. This time, it was 0 for all the plugins.

We inspected the autocorrelation plots and partial autocorrelation plots for all the plugins. Then we selected the values for p and q for each case, obtained the model, obtained the predictions for the next three months.

We then added up the results for the next three months


for every plugin, and those were the results that we submitted to the challenge.

For more details on how to apply this methodology we recommend to read [7, 10].

References

- [1] I. Antoniadis, I. Samoladas, I. Stamelos, L. Aggelis, and G. L. Bleris. Dynamical simulation models of the open source development process. In S. Koch, editor, *Free/Open Source Software Development*, pages 174–202. Idea Group Publishing, Hershey, PA, 2004.
- [2] G. Antoniol, G. Casazza, M. D. Penta, and E. Merlo. Modeling clones evolution through time series. In *Proceedings of the International Conference on Software Maintenance*, 2001.
- [3] F. Caprio, G. Casazza, M. D. Penta, and U. Villano. Measuring and predicting the Linux kernel evolution. In *Proceedings of the International Workshop of Empirical Studies on Software Maintenance*, Florence, Italy, 2001.
- [4] J.-M. Dalle and P. A. David. The allocation of software development resources in Open Source production mode. Technical report, SIEPR Policy paper No. 02-027, SIEPR, Stanford, USA, 2003. <http://siepr.stanford.edu/papers/pdf/02-27.pdf>.
- [5] E. Fuentetaja and D. J. Bagert. Software Evolution from a Time-Series perspective. In *Proceedings of the International Conference on Software Maintenance*, pages 226–229, 2002.
- [6] C. F. Kemerer and S. Slaughter. An empirical approach to studying software evolution. *IEEE Transactions on Software Engineering*, 25(4):493–509, 1999.
- [7] S. G. Makridakis, S. C. Wheelwright, and R. J. Hyndman. *Forecasting: Methods and Applications*. John Wiley & Sons, Ltd., January 1998.
- [8] G. Robles, S. Koch, and J. M. González-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)*, pages 51–56, Edinburg, Scotland, UK, 2004.
- [9] G. Robles, J. J. Merelo, and J. M. Gonzalez-Barahona. Self-organized development in libre software: a model based on the stigmergy concept. In *Proceedings of the 6th International Workshop on Software Process Simulation and Modeling (ProSim 2005)*, St.Louis, Missouri, USA, May 2005.
- [10] R. H. Shumway and D. S. Stoffer. *Time Series Analysis and its Applications. With R Examples*. Springer Texts in Statistics. Springer, 2006.
- [11] C. C. H. Yuen. An empirical approach to the study of errors in large software under maintenance. In *Proceedings of the International Conference on Software Maintenance*, 1985.
- [12] C. C. H. Yuen. A statistical rationale for evolution dynamics concepts. In *Proceedings of the International Conference on Software Maintenance*, 1987.
- [13] C. C. H. Yuen. On analyzing maintenance process data at the global and detailed levels. In *Proceedings of the International Conference on Software Maintenance*, pages 248–255, 1988.

Web Images More... jgbarah@gmail.com



Scholar



Jesus M. Gonzalez-Barahona

Forecasting the number of changes in Eclipse using time series analysis

[\[PDF\]](#) from tribler.org
[Texto completo para UC3M](#)

Authors Israel Herraiz, Jesus M Gonzalez-Barahona, Gregorio Robles
Publication date 2007/5/20
Conference Proceedings of the Fourth International Workshop on Mining Software Repositories
Pages 32
Publisher IEEE Computer Society
Description Abstract In order to predict the number of changes in the following months for the project Eclipse, we have applied a statistical (non-explanatory) model based on time series analysis. We have obtained the monthly number of changes in the CVS repository of Eclipse, using the CVSanaly tool. The input to our model was the filtered series of the number of changes per month, and the output was the number of changes per month for the next three months. Then we aggregated the results of the three months to obtain the total number of ...

Total citations [Cited by 26](#)



Scholar articles [Forecasting the number of changes in Eclipse using time series analysis](#)
 I Herraiz, JM Gonzalez-Barahona, G Robles - Proceedings of the Fourth International Workshop on ..., 2007
[Cited by 26](#) - [Related articles](#) - [All 15 versions](#)

Dates and citation counts are estimated and are determined automatically by a computer program.

[Help](#)
[Privacy](#)
[Terms](#)
[Provide feedback](#)
[My Citations](#)

Towards a theoretical model for software growth*

Israel Herraiz, Jesus M. Gonzalez-Barahona, Gregorio Robles
Grupo de Sistemas y Comunicaciones
Universidad Rey Juan Carlos, Spain
{herraiz, jgb, grex}@gsyc.escet.urjc.es

Abstract

Software growth (and more broadly, software evolution) is usually considered in terms of size or complexity of source code. However in different studies, usually different metrics are used, which make it difficult to compare approaches and results. In addition, not all metrics are equally easy to calculate for a given source code, which leads to the question of which one is the easiest to calculate without losing too much information. To address both issues, in this paper present a comprehensive study, based on the analysis of about 700,000 C source code files, calculating several size and complexity metrics for all of them. For this sample, we have found double Pareto statistical distributions for all metrics considered, and a high correlation between any two of them. This would imply that any model addressing software growth should produce this Pareto distributions, and that analysis based on any of the considered metrics should show a similar pattern, provided the sample of files considered is large enough.

1 Introduction

One of the goals of software engineering is to measure different aspects of software projects, with the aim of finding a small set of attributes that may characterize them. Among those attributes, metrics of the internal attributes of the source code are usually considered, with special attention to size and complexity.

In fact, many different metrics for size and complexity do exist, and have been successfully used in many empirical studies. However, due to this diversity in metrics, comparison of results is not always easy, and the basic question of which metrics are enough to understand a certain aspect of

the source code of a project is still largely unsolved. For obtaining some insight in both issues, we have studied a large quantity of source code (about 700,000 C files) corresponding to mature, stable software in use in a Unix-like software distribution, FreeBSD.

All the software included in the study is libre (free, open source) software, which could lead to some bias in the results, but probably they can easily be extrapolated to at least C code of any kind, since the license of the software is not likely to influence distributions of size or complexity. Probably the results can also be extended to other languages different from C, but further research is needed for that conclusion.

FreeBSD is of course not the only large collection of libre software: there are several other projects gathering software from many different libre software projects, adapting it, and producing an integrated system, ready to be used. Among them, several Linux distributions (Debian, Fedora, Ubuntu, Mandriva, etc.) are the most well known. In our case, we have selected FreeBSD because it is a good compromise between quantity of code and simplicity in packaging. In FreeBSD, source code packages (called *ports*) are easy to retrieve and handle automatically.

With the quantity of source code in FreeBSD (more than 1.7 millions of files, and 400 MSLOC in total, more than 40% of them corresponding to C code), it is possible to apply statistical methods to find out correlations and patterns in the set of analyzed data. In the case of this paper, our first motivation was to find out which independent metrics may be used to characterize size and complexity. To our surprise, we have also found that all metrics considered follow a statistical distribution (double Pareto) that has been deeply studied in other fields.

In this respect, it is also worth mentioning that some authors have proposed models to explain why these distributions appear in some of those fields. We have found that those models could be easily applied to the case of software growth, and could be used to simulate the growth of a software product and to model events in a source control management system.

*This work has been funded in part by the European Commission, under the FLOSSMETRICS (FP6-IST-5-033547) and QUALOSS (FP6-IST-5-033547) projects. Israel Herraiz has been funded in part by Consejería de Educación of Comunidad de Madrid and European Social Fund, under grant number 01/FPI/0582/2005.

code using the source packages of the system (*ports*). This meant to obtain 1.7 millions of files, with a total size of 410 MSLOC. From this set of files, 700,000 files were written in C. We measured size and complexity, using different metrics, for all these files.

We decided to focus on libre software in the study because of two main reasons: it is easily available in large quantities, and the results of the analysis can easily be checked by other research teams. The decision of considering only the C language has been also practical: most of the tools available to measure code work well with C source code.

All the metrics resulted to be highly correlated by means of power laws. Based on this, and on the fact of the easiness of calculation, we find it interesting to use SLOC to characterize both size and complexity of software products, in any kind of studies. For instance, to study the growth of a software project, in the part regarding the internal attributes of the software, it would be enough to measure only SLOC, to obtain a landscape of the evolution of the size and complexity of the project.

All the metrics were found to follow a double Pareto distribution. This distribution is formed by a lognormal distribution in the main body, and power laws distributions in the tails for high and low values of the distribution. Some composed metrics (such as number of comments per SLOC) presented also double Pareto distributions.

This kind of distributions has been found also for the size of files in a filesystem. Some theoretical models about the growth of file systems have been proposed to explain these cases, which have also been used to optimize the download of files from web and FTP servers [2].


The most interesting model we have found is the *Recursive Forest File* model, proposed by Michael Mitzenmacher [12]. It can explain how files change over time, and how they are inserted and removed from a tree of files. Because of its nature, it would be easily adaptable to the case of a source control system, being therefore able of explaining how and why software grows.

In further research, we will try to adapt this model to the case of a control version system, and to verify it against the actual history of a software project.

References

- [1] I. Antoniadis, I. Samoladas, I. Stamelos, L. Aggelis, and G. L. Bleris. Dynamical simulation models of the open source development process. In S. Koch, editor, *Free/Open Source Software Development*, pages 174–202. Idea Group Publishing, Hershey, PA, 2004.
- [2] P. Badford, A. Bestavros, A. Bradley, and M. Crovella. Changes in Web client access patterns: characteristics and caching implications. *World Wide Web*, 2(1-2):15–28, June 1999.
- [3] S. D. Conte. *Software Engineering Metrics and Models (Benjamin/Cummings series in software engineering)*. Benjamin-Cummings Pub Co, 1986.
- [4] J.-M. Dalle and P. A. David. The allocation of software development resources in Open Source production mode. Technical report, SIEPR Policy paper No. 02-027, SIEPR, Stanford, USA, 2003. <http://siepr.stanford.edu/papers/pdf/02-27.pdf>.
- [5] M. W. Godfrey and Q. Tu. Evolution in open source software: A case study. In *ICSM '00: Proceedings of the International Conference on Software Maintenance (ICSM'00)*, pages 131–142, Washington, DC, USA, October 2000. IEEE Computer Society.
- [6] I. Herraiz, G. Robles, J. M. Gonzalez-Barahona, A. Capiluppi, and J. F. Ramil. Comparison between SLOCs and number of files as size metrics for software evolution analysis. In *Proceedings of the 10th European Conference on Software Maintenance and Reengineering*, pages 203–210, Bari, Italy, 2006.
- [7] S. H. Kan. *Metrics and Models in Software Quality Engineering (2nd Edition)*. Addison-Wesley Professional, September 2002.
- [8] S. Koch. Evolution of Open Source Software systems - a large-scale investigation. In *Proceedings of the 1st International Conference on Open Source Systems*, Genova, Italy, July 2005.
- [9] M. M. Lehman, J. F. Ramil, and U. Sandler. An approach to modelling long-term growth trends in software systems. In *International Conference on Software Maintenance*, pages 219–228, Florence, Italy, November 2001.
- [10] M. M. Lehman, J. F. Ramil, P. D. Wernick, D. E. Perry, and W. M. Turski. Metrics and laws of software evolution - the nineties view. In *METRICS '97: Proceedings of the 4th International Symposium on Software Metrics*, page 20, nov 1997.
- [11] M. Mitzenmacher. A brief history of generative models for power law and lognormal distributions. *Internet Mathematics*, 1(2):226–251, 2004.
- [12] M. Mitzenmacher. Dynamic models for file sizes and double Pareto distributions. *Internet Mathematics*, 1(3):305–333, 2004.
- [13] G. Robles, J. J. Amor, J. M. Gonzalez-Barahona, and I. Herraiz. Evolution and growth in large libre software projects. In *Proceedings of the International Workshop on Principles in Software Evolution*, pages 165–174, Lisbon, Portugal, September 2005.
- [14] G. Robles, J. J. Merelo, and J. M. Gonzalez-Barahona. Self-organized development in libre software: a model based on the stigmergy concept. In *Proceedings of the 6th International Workshop on Software Process Simulation and Modeling (ProSim 2005)*, St.Louis, Missouri, USA, May 2005.
- [15] W. M. Turski. Reference model for smooth growth of software systems. *IEEE Transactions on Software Engineering*, 22(8):599–600, 1996.
- [16] W. M. Turski. The reference model for smooth growth of software systems revisited. *IEEE Transactions on Software Engineering*, 28(8):814–815, 2002.

Web Images More... jgbarah@gmail.com



Scholar ← Edit 🗑️ Export ▾



Jesus M. Gonzalez-Barahona

Towards a theoretical model for software growth

[\[PDF\] from researchgate.net](#)
[Texto completo para UC3M](#)

Authors: Israel Herraiz, Jesus M Gonzalez-Barahona, Gregorio Robles
 Publication date: 2007/5/20
 Conference: Proceedings of the Fourth International Workshop on Mining Software Repositories
 Pages: 21
 Publisher: IEEE Computer Society
 Description: Abstract Software growth (and more broadly, software evolution) is usually considered in terms of size or complexity of source code. However in different studies, usually different metrics are used, which make it difficult to compare approaches and results. In addition, not all metrics are equally easy to calculate for a given source code, which leads to the question of which one is the easiest to calculate without losing too much information. To address both issues, in this pa-per present a comprehensive study, based on the analysis of about ...

Total citations [Cited by 59](#)



Scholar articles [Towards a theoretical model for software growth](#)
 I Herraiz, JM Gonzalez-Barahona, G Robles - Proceedings of the Fourth International Workshop on ..., 2007
[Cited by 59](#) - [Related articles](#) - [All 14 versions](#)

[Towards a theoretical model for software growth *](#)
 JM Gonzalez-Barahona, G Robles - Proceedings of the 29th International Conference on ..., 2007

Dates and citation counts are estimated and are determined automatically by a computer program.

[Help](#) [Privacy](#) [Terms](#) [Provide feedback](#) [My Citations](#)

Corporate Involvement of Libre Software: Study of Presence in Debian Code over Time*

Gregorio Robles, Santiago Dueñas, Jesus M. Gonzalez-Barahona

GSyC/LibreSoft, Universidad Rey Juan Carlos (Madrid, Spain)
{greg,sduenas,jgb}@gsyc.escet.urjc.es

Abstract. Although much of the research on the libre (free, open source) phenomenon has been focused on the involvement of volunteers, the role of companies is also important in many projects. In fact, during the last years, the involvement of companies in the libre software world seems to be raising. In this paper we present an study that shows, quantitatively, how important this involvement is in the production of the largest collection of code available for Linux: the Debian GNU/Linux distribution. By studying copyright attributions in source code, we have identified those companies with more attributed code, and the trend of corporate presence in Debian from 1998 to 2004.

Keywords: open source, libre software, involvement of companies, empirical study, software business

1 Introduction

For companies producing computer programs, libre software² is not yet another competitor playing with the same rules. The production of libre software differs from *traditional* software development in many fundamental aspects, ranging from ethical and psychological motivation to new economic and marketing premises, to new practices and procedures in the development process itself.

One of the key differences is the different role of users. While in the *classical* software development environment the development team can be clearly distinguished from the users, most of the libre software projects develop around themselves a community [7]. This community is usually formed by people with many different involvements, from pure users to core developers, including many mixed roles, such as that of users contributing with patches (small modifications) to the code. Therefore, in most libre software projects we may observe a *continuum* of commitment to the project which includes a wide range of occasional contributors.

* This work has been funded in part by the European Commission, under the FLOSS-METRICS (FP6-IST-5-033547) and FLOSSWORLD (FP6-IST-015722) projects.

² Through this paper the term “libre software” will be used to refer to code that conforms either to the definition of “free software” (according to the Free Software Foundation) or of “open source software” (according to the Open Source Initiative).

Please use the following format when citing this chapter:

Robles, G., Dueñas, S., Gonzalez-Barahona, J.M., 2007, in IFIP International Federation for Information Processing, Volume 234, Open Source Development, Adoption and Innovation, eds. J. Feller, Fitzgerald, B., Scacchi, W., Sillitti, A., (Boston: Springer), pp. 121–132.

- Workshop on Mining Software Repositories*, pages 106–110, St. Louis, Missouri, USA, May 2005.
17. Gregorio Robles, Jesús M. González-Barahona, and Martin Michlmayr. Evolution of volunteer participation in libre software projects: evidence from Debian. In *Proceedings of the 1st International Conference on Open Source Systems*, pages 100–107, Genoa, Italy, July 2005.
 18. Diomidis Spinellis. *Code Reading: The Open Source Perspective*. Addison Wesley Professional, 2003.
 19. Ilkka Tuomi. Evolution of the Linux Credits file: Methodological challenges and reference data for Open Source research. *First Monday*, 9(6), June 2004.
 20. Georg von Krogh, Sebastian Spaeth, and Karim R. Lakhani. Community, joining, and specialization in Open Source Software innovation: A case study. *MIT Sloan Working Paper No. 4413-03*, 2003.
 21. David A. Wheeler. More than a gigabuck: Estimating GNU/Linux’s size, June 2001.



Scholar



Jesus M. Gonzalez-Barahona

Corporate involvement of libre software: Study of presence in Debian code over time

[PDF] from flosshub.org

Authors Gregorio Robles, Santiago Duenas, Jesus M Gonzalez-Barahona

Publication date 2007/1/1

Book Open Source Development, Adoption and Innovation

Pages 121-132

Publisher Springer US

Description Abstract Although much of the research on the libre (free, open source) phenomenon has been focused on the involvement of volunteers, the role of companies is also important in many projects. In fact, during the last years, the involvement of companies in the libre software world seems to be raising. In this paper we present an study that shows, quantitatively, how important this involvement is in the production of the largest collection of code available for Linux: the Debian GNU/Linux distribution. By studying copyright ...

Total citations Cited by 24



Scholar articles [Corporate involvement of libre software: Study of presence in Debian code over time](#)
 G Robles, S Duenas, JM Gonzalez-Barahona - Open Source Development, Adoption and Innovation, 2007
[Cited by 24](#) - [Related articles](#) - [All 8 versions](#)

[Corporate Involvement of Libre Software: Study of Presence in Debian Code over Time](#) ★
 S Dueas, G Robles, JM Gonzalez-Barahona - IFIP International Federation for Information Processing, 2010

Dates and citation counts are estimated and are determined automatically by a computer program.

[Help](#) [Privacy](#) [Terms](#) [Provide feedback](#) [My Citations](#)

Quantitative Analysis of the Wikipedia Community of Users

Felipe Ortega

Universidad Rey Juan Carlos.
Tulipan s/n.
28933, Mostoles. Madrid. SPAIN
jfelipe@gsyc.es

Jesus M. Gonzalez-Barahona

Universidad Rey Juan Carlos.
Tulipan s/n.
28933, Mostoles. Madrid. SPAIN
jgb@gsyc.es

Abstract

Many activities of editors in Wikipedia can be traced using its database dumps, which register detailed information about every single change to every article. Several researchers have used this information to gain knowledge about the production process of articles, and about activity patterns of authors. In this analysis, we have focused on one of those previous works, by Kittur et al. First, we have followed the same methodology with more recent and comprehensive data. Then, we have extended this methodology to precisely identify which fraction of authors are producing most of the changes in Wikipedia's articles, and how the behaviour of these authors evolves over time. This enabled us not only to validate some of the previous results, but also to find new interesting evidences. We have found that the analysis of sysops is not a good method for estimating different levels of contributions, since it is dependent on the policy for electing them (which changes over time and for each language). Moreover, we have found new activity patterns classifying authors by their contributions during specific periods of time, instead of using their total number of contributions over the whole life of Wikipedia. Finally, we present a tool that automates this extended methodology, implementing a quick and complete quantitative analysis of every language edition in Wikipedia.

Categories and Subject Descriptors H.3.7 [Information Storage and Retrieval]: Digital Libraries—system issues

General Terms Performance

Keywords quantitative analysis, methodology, Wikipedia, WikiXRay

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

WikiSym'07, October 21–23, 2007, Montréal, Québec, Canada.
Copyright © 2007 ACM 978-1-59593-861-9/07/0010...\$5.00

1. Introduction

Wikis present a new paradigm of website with dynamic contents created by its own users. When we refer to content development systems, *wiki* is now synonymous of *collaborative*, *agile*, *powerful* and even *easy*. They are now a core component of what is presented as the Web 2.0, providing tools to create contents through collaboration and interaction of users.

Wikipedia is, by far, the most successful example of this new paradigm of web services. With more than 200 different language editions, and one of the biggest communities of users of the Internet, Wikipedia has showed us the power of collaborative content development. As of April 26th, 2007, a total of 1,755,932 articles are already available in the English language edition, and the top 10 language editions (English, German, French, Japanese, Polish, Dutch, Italian, Portuguese, Spanish and Swedish) accumulate a total sum of 4,852,810 articles. Many scientific works are increasingly referencing Wikipedia, and many learning systems employ it as a primary source of information.

One of the most relevant contributions that Wikipedia has made to the wiki community is the MediaWiki software, a libre (free, open source) software that facilitates the task of creating, configuring, maintaining and using a wiki. MediaWiki provides some very useful tools for wiki users and administrators, for instance:

- *Easy-to-use editing interface*: A toolbar provides easy access to the most common editing functions. This reduces the learning curve of new users, who can get up to speed in editing contents fastly.
- *Content classification*: Contents can be classified attending to their topic into several categories, presented in special pages, thus helping users searching for a certain topic.
- *Discussion pages for every article*: Each article in MediaWiki is accompanied by its talk page, a special page allowing users interested in editing that article to exchange their impressions, and collaborate towards obtaining consensus about the article's contents and presentation.

instead of focusing on the most active contributors in period 40 concentrating the 10% of the total number of edits in that period, we can analyse the most popular articles in period 40, (those obtaining the highest number of contributions), that receive the 10% of the total number of edits in that period. We could then use those data to correlate them with the number of distinct authors per article, the length of those articles, etc.

We have applied our tool to the quantitative analysis of the English edition of Wikipedia, trying to validate some of the results and previous methodologies we have presented so far. We have also used it to extend those previous methodologies to other language editions of Wikipedia and to include per period parameters in the quantitative analysis process.

5. CONCLUSIONS AND FUTURE WORK

In this research paper, we revised previous methodologies proposed for the quantitative analysis of Wikipedia. We showed that, although these methodologies reveal some interesting behavioral patterns, they can also hide another important phenomena that could lead us to acquire a more complete and detailed picture of the Wikipedia's community of users.

Later, we presented our own proposal for enhancing these previous methodologies to undertake the quantitative analysis of Wikipedia. This extended methodology can be applied to further language editions other than English, and it focuses on the study of per period parameters to precisely follow the evolution in time of the Wikipedia community of users.

We finally showed some graphs summarizing the most relevant parameters that we can study with this new methodology using *WikiXRay*, a Python software tool that automates the quantitative analysis of all Wikipedia language editions. Further development of *WikiXRay* will include extending this per period analysis method to Wikipedia articles, and reflecting possible correlations between featured parameters that will let us better explain the behavior of the communities of users in all language editions of Wikipedia.

References

- [1] D. Anthony, S. W. Smith, and T. Williamson. Explaining quality in internet collective goods: Zealots and good samaritans. the case of wikipedia. November 2005.
- [2] L. S. Buriol, C. Castillo, D. Donato, S. Leonardi, and S. Millozzi. Temporal evolution of the wikigraph. In *Proceedings of the Web Intelligence Conference*, December 2006.
- [3] C. Gini. On the measure of concentration with especial reference to income and wealth. In *Cowless Comission*, 1936.
- [4] A. Kittur, E. Chi, B. A. Pendleton, B. Suh, and T. Mytkowicz. Power of the few vs. wisdom of the crowd: Wikipedia and the rise of the bourgeoisie. In *Proceedings of the 25th Annual ACM Conference on Human Factors in Computing Systems (CHI 2007)*, April-May 2007.
- [5] F. B. Viegas, M. Wattenberg, and K. Dave. Studying cooperation and conflict between authors with history flow visualizations. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 575–582, April 2004.
- [6] F. B. Viegas, M. Wattenberg, J. Kriss, and F. van Ham. Talk before you type: Coordination in wikipedia. In *Proceedings of the 40th Annual Hawaii International Conference on System Sciences (HICSS'07)*, page 78a. Computer Society Press, January 2007.
- [7] J. Voss. Measuring wikipedia. In *Proceedings of the ISSI 2005*, July 2005.



Scholar



Jesus M. Gonzalez-Barahona

Quantitative analysis of thewikipedia community of users

[\[PDF\]](#) from smith.edu

Authors Felipe Ortega, Jesus M Gonzalez Barahona
 Publication date 2007/10/21
 Conference Proceedings of the 2007 international symposium on Wikis
 Pages 75-86
 Publisher ACM

Description Abstract Many activities of editors in Wikipedia can be traced using its database dumps, which register detailed information about every single change to every article. Several researchers have used this information to gain knowledge about the production process of articles, and about activity patterns of authors. In this analysis, we have focused on one of those previous works, by Kittur et al. First, we have followed the same methodology with more recent and comprehensive data. Then, we have extended this methodology to ...

Total citations [Cited by 84](#)



Scholar articles [Quantitative analysis of thewikipedia community of users](#)
 F Ortega, JM Gonzalez Barahona - Proceedings of the 2007 international symposium on ..., 2007
[Cited by 84](#) - [Related articles](#) - [All 9 versions](#)

Dates and citation counts are estimated and are determined automatically by a computer program.

[Help](#)
[Privacy](#)
[Terms](#)
[Provide feedback](#)
[My Citations](#)

On The Inequality of Contributions to Wikipedia

Felipe Ortega, Jesus M. Gonzalez-Barahona and Gregorio Robles
 Libresoft Group, Universidad Rey Juan Carlos
 Tulipan, s/n 28933. Mostoles.
 Madrid. SPAIN.
 Email: {jfelipe,jgb,grex}@gsyc.es

Abstract—Wikipedia is one of the most successful examples of massive collaborative content development. However, many of the mechanisms and procedures that it uses are still unknown in detail. For instance, how equal (or unequal) are the contributions to it has been discussed in the last years, with no conclusive results. In this paper, we study exactly that aspect by using Lorenz curves and Gini coefficients, very well known instruments to economists. We analyze the trends in the inequality of distributions for the ten biggest language editions of Wikipedia, and their evolution over time. As a result, we have found large differences in the number of contributions by different authors (something also observed in free, open source software development), and a trend to stable patterns of inequality in the long run.

I. INTRODUCTION

Nowadays, Wikipedia is one of the most successful examples of large-scale collaborative content development. In just some few years, it has been able to attract an impressive number of contributors who create content, update it, modify it, and try to adapt it to the emerging policies that the project is establishing. Probably one of the key factors in its success is the low entry barrier for new authors. Thanks to the easy-to-use and friendly user interface that Wikipedia provides to authors, they can start to contribute even upon arrival to a Wikipedia page for the first time.

However, the fact that anyone can contribute to the Wikipedia does not imply that every Wikipedia visitor becomes an author (actually, only a very small fraction of visitors contribute). And the fact that it is easy to contribute does not mean that all authors contribute the same way, neither with the same intensity. On the contrary, the number of contributions is known to vary much from author to author [1]. But the pattern of these differences, or, in other words, how equal or unequal the contributions from different authors are, has not yet been characterized in detail.

Indeed, in the last years there has been a great controversy about this point. For instance, on September 4th, 2006, Aaron Swartz included a quote from Wikipedia's founder, Jimmy Wales, in his blog [2] where Wales argued that the majority of the total number of contributions to the Wikipedia came from a small group of authors. Swartz used a different metric, counting the number of characters in each contribution (rather than the number of contributions), and searched for the text blocks that remained in the final version of the article. He then applied this metric to several articles picked up at random, and showed that less frequent contributors were actually providing much of the articles contents. Despite that, as articles continue

to evolve and change over time, we will inevitably miss some contribution effort if we focus only in the final revision of an article. Perhaps some contributions are removed later in the article's life not remaining in the final version, but we should take it into account if we want to have a complete picture of the mechanisms found in the Wikipedia.

To analyze what is actually happening in this respect, in this paper we show the results of calculating the level of inequality in the contributions to several language editions of the Wikipedia, using the Gini coefficient (a well-known metric to measure inequality distributions introduced by Conrado Gini). This coefficient was originally thought to analyze the distribution of wealth among a population, but is used in other domains as well to provide a quantitative measure of the inequality of a distribution under study. By comparing the Gini coefficients of the various language editions of the Wikipedia, and its evolution over time, we can learn not only about how unequal the contributions by different authors are, but also if it is converging (or not) to some value.

More in detail, we have considered the top-ten Wikipedia language editions, according to their total number of articles (which will allow comparison of results among different language editions, as well as the search for peculiarities and similarities). We calculated the sum of the number of contributions made by authors for each language edition, and then, obtained the Gini coefficient of this overall effort. This coefficient will give information about the inequality in contributions from the whole group of authors, for every language edition considered in our study. But, taking into account that the activity of authors may change over time, we have also calculated the Gini coefficient for the contributions to each edition on a monthly basis as well.

As a result, we have obtained a good picture of the evolution of inequality of contributions over time for all the studied editions. This picture shows interesting patterns, such as a tendency to converge to a stable Gini coefficient after some oscillations during the early life of the editions, despite the tremendous growth in number of authors and articles. In addition to this, we also found relevant differences among the editions under study.

In the following sections, we first revisit some previous research about the contribution of users to Wikipedia. Then, we present our methodology for measuring the inequality of contributions to the Wikipedia, for the top-ten language editions, followed by results and discussion on them. Finally,

basis.

Moreover, the level of inequality was not correlated, in many cases, with other parameters like the number of articles or the number of authors. For example, the Japanese Wikipedia, which is in the 4th place by its number of articles, and in the 5th place attending its total number of authors, presented the most equal pattern regarding the aggregate number of contributions. Other language editions, like German and especially English, with a higher number of authors, showed a higher level of inequality. This reflects that it is not the number of authors producing contents or the total number of articles being modified what determines the level of inequality in a certain community of users. Other behavioral and social issues could be affecting how equal or unequal those contributions are.

The second remarkable conclusion is that the evolution over time of this inequality level has remained almost stable over the months in every language edition. We found that the typical value of the Gini coefficient per month was situated between the 80% and 85% limits in every case, at least for the last two years period. In the ancient history of these language editions, we found a very variable behavioral pattern at the very beginning of the graphs. But later, we found a common growing trend of the inequality level in every language edition, just until it reaches the *standard* margin mentioned above. This is also very interesting, because it probably points out a stability value towards which every edition of Wikipedia tends to evolve, thus possibly giving us a hint about the level of maturity achieved by a certain language edition.


Thus, we have found not only that the inequality of the aggregated number of contributions is very high, but also that this level of inequality has remained somewhat constant in the recent history of every language edition. We should remark that we need both data to sustain these hypotheses, because as we have shown previously, the Gini coefficient of the aggregate number of contributions is merely a useful, but not definitive, indicator of the real behavior of the communities of users we analyzed.

Our future lines of research include considering the size of the contributions made by each author in this analysis, (possibly in number of characters and in number of words), thus reflecting the real amount of data added or deleted from the Wikipedia by an author. Hence, we would be able to offer another interesting point of view to get insight into the inner behavioral patterns of the Wikipedia.

REFERENCES

- [1] A. Kittur, E. Chi, B. A. Pendleton, B. Suh, and T. Mytkowicz, "Power of the few vs. wisdom of the crowd: Wikipedia and the rise of the bourgeoisie," in *Proceedings of the 25th Annual ACM Conference on Human Factors in Computing Systems (CHI 2007)*, Apr./May 2007.
- [2] A. Swartz. (2006, Sep.) Who writes wikipedia. [Online]. Available: <http://www.aaronsw.com/weblog/whowriteswikipedia>
- [3] C. Gini, "On the measure of concentration with especial reference to income and wealth," in *Cowless Comission*, 1936.
- [4] R. Dorfman, "A formula for the gini coefficient," *The Review of Economics and Statistics*, no. 61, pp. 146–149, Mar. 1979.
- [5] J. A. Mills *et al.*, "Statistical inference via bootstrapping for measures of inequality," *Epidemiological Bulletin*, vol. 22, no. 12, Mar./Apr. 1997.
- [6] C. Castillo-Salgado *et al.*, "Measuring health inequalities: Gini coefficient and concentration index," *Epidemiological Bulletin*, vol. 22, no. 1, Mar. 2001.
- [7] A. Wagstaff *et al.*, "On the measurements of inequalities in health," *Soc. Sci. Med.*, vol. 33, no. 5, pp. 545–577, 1991.
- [8] G. von Krogh, S. Spaeth, and K. R. Lakhani, "Community, joining, and specialization in Open Source Software innovation: A case study," *MIT Sloan Working Paper No. 4413-03*, 2003.
- [9] 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*, vol. 11, no. 3, pp. 309–346, 2002.
- [10] S. Koch and G. Schneider, "Effort, cooperation and coordination in an open source software project: GNOME," *Information Systems Journal*, vol. 12, no. 1, pp. 27–42, 2002.
- [11] J. Voss, "Measuring wikipedia," in *Proceedings of the ISSI 2005*, Jul. 2005.
- [12] F. B. Viegas, M. Wattenberg, and K. Dave, "Studying cooperation and conflict between authors with history flow visualizations," in *Proceedings of the SIGCHI conference on Human factors in computing systems*, Apr. 2004, pp. 575–582.
- [13] F. B. Viegas, M. Wattenberg, J. Kriss, and F. van Ham, "Talk before you type: Coordination in wikipedia," in *Proceedings of the 40th Annual Hawaii International Conference on System Sciences (HICSS'07)*. Computer Society Press, Jan. 2007, p. p. 78a.
- [14] B. Stvilia, M. Twidale, L. Gasser, and L. Smith, "Information quality discussions in wikipedia," UIUCLIS, Technical Report ISRN 2005/2+CSCW, 2005. [Online]. Available: <http://mailer.fsu.edu/bstvilia/papers/qualWiki.pdf>
- [15] —, "Information quality in a community-based encyclopedia," in *Knowledge Management: Nurturing Culture, Innovation, and Technology - Proceedings of the 2005 International Conference on Knowledge Management*, S. Hawamdeh, Ed., Charlotte, NC: World Scientific Publishing Company, 2005, pp. 101–113.
- [16] B. Stvilia, M. Twidale, L. Smith, and L. Gasser, "Assessing information quality of a community-based encyclopedia," in *Proceedings of the International Conference on Information Quality - ICIQ 2005*, Cambridge, MA, USA, 2005, pp. 442–454.
- [17] —. (2006) Information quality work organization in wikipedia. (Under review). [Online]. Available: http://mailer.fsu.edu/bstvilia/papers/stvilia_wikipedia_infoWork_p.pdf
- [18] D. M. Wilkinson and B. A. Huberman, "Assessing the value of cooperation in wikipedia," 2007. [Online]. Available: <http://www.citebase.org/abstract?id=oai:arXiv.org:cs/0702140>
- [19] M. Völkel, M. Kröttsch, D. Vrandečić, H. Haller, and R. Studer, "Semantic wikipedia," in *WWW '06: Proceedings of the 15th international conference on World Wide Web*. New York, NY, USA: ACM Press, 2006, pp. 585–594.
- [20] M. Kröttsch, D. Vrandečić, and M. Volkel, "Wikipedia and the semantic web - the missing links," 2005. [Online]. Available: citeseer.ist.psu.edu/krotzsch05wikipedia.html
- [21] M. Strube and S. P. Ponzetto, "Wikirelate! computing semantic relatedness using wikipedia," in *Proceedings of the 21th National Conference on Artificial Intelligence (AAAI06)*, Boston, Massachusetts, USA, Jul. 16–20, 2006.
- [22] T. Holloway, M. Bozicevic, and K. Borner, "Analyzing and visualizing the semantic coverage of wikipedia and its authors," 2005. [Online]. Available: <http://www.citebase.org/abstract?id=oai:arXiv.org:cs/0512085>
- [23] D. Anthony, S. W. Smith, and T. Williamson, "Explaining quality in internet collective goods: Zealots and good samaritans. the case of wikipedia," Nov. 2005.
- [24] R. A. Ghosh and V. V. Prakash, "The orbiten free software survey," *First Monday*, vol. 5, no. 7, May 2000, http://www.firstmonday.dk/issues/issue5_7/ghosh/.
- [25] G. Robles, J. M. Gonzalez-Barahona, and J.-J. Merelo, "Beyond executable source code: The importance of other source artifacts in software development (a case study)," *Journal of Systems and Software*, vol. 79, no. 9, pp. 1233–1248, September 2006.

Web Images More... jgbarah@gmail.com



Scholar



Jesus M. Gonzalez-Barahona

On the inequality of contributions to Wikipedia

[\[PDF\] from urjc.es](#)

Authors: Felipe Ortega, Jesus M Gonzalez-Barahona, Gregorio Robles
 Publication date: 2008/1/7
 Conference: Hawaii International Conference on System Sciences, Proceedings of the 41st Annual
 Pages: 304-304
 Publisher: IEEE
 Description: Abstract—Wikipedia is one of the most successful examples of massive collaborative content development. However, many of the mechanisms and procedures that it uses are still unknown in detail. For instance, how equal (or unequal) are the contributions to it has been discussed in the last years, with no conclusive results. In this paper, we study exactly that aspect by using Lorenz curves and Gini coefficients, very well known instruments to economists. We analyze the trends in the inequality of distributions for the ten biggest ...

Total citations [Cited by 104](#)



Scholar articles: [On the inequality of contributions to Wikipedia](#)
 F Ortega, JM Gonzalez-Barahona, G Robles - Hawaii International Conference on System Sciences, ..., 2008
[Cited by 104](#) - [Related articles](#) - [All 7 versions](#)

Dates and citation counts are estimated and are determined automatically by a computer program.

[Help](#)
[Privacy](#)
[Terms](#)
[Provide feedback](#)
[My Citations](#)

Determinism and Evolution

Israel Herraiz
Universidad Rey Juan Carlos
Madrid, Spain
herraiz@gysc.es

Jesus M. Gonzalez-
Barahona
Universidad Rey Juan Carlos
Madrid, Spain
jgb@gysc.es

Gregorio Robles
Universidad Rey Juan Carlos
Madrid, Spain
grex@gysc.es

ABSTRACT

It has been proposed that software evolution follows a Self-Organized Criticality (SOC) dynamics. This fact is supported by the presence of long range correlations in the time series of the number of changes made to the source code over time. Those long range correlations imply that the current state of the project was determined time ago. In other words, the evolution of the software project is governed by a sort of determinism. But this idea seems to contradict intuition. To explore this apparent contradiction, we have performed an empirical study on a sample of 3,821 libre (free, open source) software projects, finding that their evolution projects is short range correlated. This suggests that the dynamics of software evolution may not be SOC, and therefore that the past of a project does not determine its future except for relatively short periods of time, at least for libre software.

Categories and Subject Descriptors

D.2.7 [Software Engineering]: Distribution, Maintenance, and Enhancement—*Reverse engineering, Version control*;
D.2.9 [Software Engineering]: Management—*Life cycle, Software configuration management, Time estimation*

General Terms

Theory

Keywords

software evolution, time series analysis, self-organized criticality, long term process, short term process

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

MSR'08, May 10-11, 2008, Leipzig, Germany.

Copyright 2008 ACM 978-1-60558-024-1/08/05 ...\$5.00.

1. INTRODUCTION

Libre¹ software development has been traditionally a source of strange cases of software evolution. The first to report one of those were Godfrey and Tu [8, 9]. Their findings suggested that the classical Lehman's *laws of software evolution* [15] were not fulfilled in the case of Linux, because it was evolving at a growing rate (which in fact was still growing 5 years later [21]).

Those cases raised the question of whether libre software evolves differently than proprietary software, and whether the *laws of software evolution* are a valid approach for an universal theory of software evolution.

Although these questions have been addressed many times [14], most of the findings and models exposed on those works have failed to provide the theoretical background needed for a proper and universal theory of software evolution. One study that addressed the problem was Wu, in his PhD thesis [26], who among other interesting findings proposed that the evolution of libre software was governed by a Self Organized Criticality (SOC) dynamics.

This conclusion was supported by the presence of long range correlated time series in a set of 11 projects. Regardless the suitability of the selected projects for this kind of study, the limited amount of cases studies, or even the methodology used, we find the idea of long range correlated processes in software evolution as contrary to common intuition. Long range correlation would mean that the current state of the project is determined (or at least, heavily influenced) by events that took place long time ago. In other words, the evolution of libre software is governed by a sort of determinism.

In order to explore if this kind of dynamics is a property of libre software, we have selected a large (3,821) sample of projects, performing an analysis similar to the one by Wu. We have studied the daily time series of changes, focusing on deciding whether their profile were short or long range correlated.

The projects were obtained out of the whole population of projects stored in SourceForge.net, a well known hosting service for libre software projects, that provides a web-based integrated development environment. The data was obtained using the *CVSAnalY SourceForge dataset*², maintained by our research group.

¹In this paper we will use the term “libre software” to refer both to “free software”, as defined by the Free Software Foundation, and “open source software”, as defined by the Open Source Initiative.

²http://libresoft.es/Results/CVSAAnalY_SF

International ERCIM Symposium on Software Evolution. ERCIM, 2007.

- [4] A. Capiluppi and M. Michlmayr. *Open Source development, adoption and innovation*, chapter From the Cathedral to the Bazaar: An Empirical Study of the Lifecycle of Volunteer Community Projects, pages 31–44. IFIP: International Federation for Information Processing. Springer Boston, 2007.
- [5] F. Caprio, G. Casazza, M. D. Penta, and U. Villano. Measuring and predicting the Linux kernel evolution. In *Proceedings of the International Workshop of Empirical Studies on Software Maintenance*, Florence, Italy, 2001.
- [6] J.-M. Dalle and P. A. David. The allocation of software development resources in Open Source production mode. Technical report, SIEPR Policy paper No. 02-027, SIEPR, Stanford, USA, 2003. <http://siepr.stanford.edu/papers/pdf/02-27.pdf>.
- [7] A. R. Fasolino, D. Natale, A. Poli, and A. Alberigi-Quaranta. Metrics in the development and maintenance of software: an application in a large scale environment. *Journal of Software Maintenance: Research and Practice*, 12:343–355, 2000.
- [8] M. 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.
- [9] M. Godfrey and Q. Tu. Growth, evolution, and structural change in open source software. In *International Workshop on Principles of Software Evolution*, Vienna, Austria, September 2001.
- [10] I. Herraiz, J. M. Gonzalez-Barahona, and G. Robles. Forecasting the number of changes in Eclipse using time series analysis. In *International Workshop on Mining Software Repositories*. IEEE Computer Society, 2007.
- [11] I. Herraiz, J. M. Gonzalez-Barahona, G. Robles, and D. M. German. On the prediction of the evolution of libre software projects. In *IEEE International Conference on Software Maintenance*, pages 405–414. IEEE Computer Society, 2007.
- [12] J. Howison, M. Conklin, and K. Crowston. FLOSSMole: a collaborative repository for FLOSS research data and analyses. *International Journal of Information Technology and Web Engineering*, 1(3):17–26, July–September 2006.
- [13] C. F. Kemerer and S. Slaughter. An empirical approach to studying software evolution. *IEEE Transactions on Software Engineering*, 25(4):493–509, 1999.
- [14] S. Koch. Evolution of Open Source Software systems - a large-scale investigation. In *Proceedings of the 1st International Conference on Open Source Systems*, Genova, Italy, July 2005.
- [15] M. M. Lehman and L. A. Belady, editors. *Program Evolution. Processes of Software Change*. Academic Press Inc., 1985.
- [16] M. M. Lehman, J. F. Ramil, and U. Sandler. An approach to modelling long-term growth trends in software systems. In *International Conference on Software Maintenance*, pages 219–228, Florence, Italy, November 2001.
- [17] M. M. Lehman, J. F. Ramil, P. D. Wernick, D. E. Perry, and W. M. Turski. Metrics and laws of software evolution - the nineties view. In *METRICS '97: Proceedings of the 4th International Symposium on Software Metrics*, page 20, nov 1997.
- [18] N. H. Madhavji, J. Fernandez-Ramil, and D. E. Perry, editors. *Software Evolution and Feedback. Theory and Practice*. Wiley, 2006.
- [19] Y. Peng, F. Li, and A. Mili. Modeling the evolution of operating systems: An empirical study. *The Journal of Systems and Software*, 80(1):1–15, 2007.
- [20] E. S. Raymond. The cathedral and the bazaar. *First Monday*, 3(3), March 1998. <http://www.firstmonday.dk/issues/issue3.3/raymond/>.
- [21] G. Robles, J. J. Amor, J. M. Gonzalez-Barahona, and I. Herraiz. Evolution and growth in large libre software projects. In *Proceedings of the International Workshop on Principles in Software Evolution*, pages 165–174, Lisbon, Portugal, September 2005.
- [22] G. Robles, J. J. Merelo, and J. M. Gonzalez-Barahona. Self-organized development in libre software: a model based on the stigmergy concept. In *Proceedings of the 6th International Workshop on Software Process Simulation and Modeling (ProSim 2005)*, St.Louis, Missouri, USA, May 2005.
- [23] R. H. Shumway and D. S. Stoffer. *Time Series Analysis and Applications. With R Examples*. Springer Texts in Statistics. Springer, 2006.
- [24] W. M. Turski. Reference model for smooth growth of software systems. *IEEE Transactions on Software Engineering*, 22(8):599–600, 1996.
- [25] W. M. Turski. The reference model for smooth growth of software systems revisited. *IEEE Transactions on Software Engineering*, 28(8):814–815, 2002.
- [26] J. Wu. *Open Source Software evolution and its dynamics*. PhD thesis, University of Waterloo, 2006.
- [27] J. Wu, R. Holt, and A. E. Hassan. Empirical evidence for SOC dynamics in software evolution. In *IEEE International Conference on Software Maintenance*, pages 244–254. IEEE Computer Society, 2007.
- [28] C. C. H. Yuen. An empirical approach to the study of errors in large software under maintenance. In *Proceedings of the International Conference on Software Maintenance*, 1985.
- [29] C. C. H. Yuen. A statistical rationale for evolution dynamics concepts. In *Proceedings of the International Conference on Software Maintenance*, 1987.
- [30] C. C. H. Yuen. On analyzing maintenance process data at the global and detailed levels. In *Proceedings of the International Conference on Software Maintenance*, pages 248–255, 1988.



Scholar



Jesus M. Gonzalez-Barahona

Determinism and evolution

[\[PDF\] from researchgate.net](#)

Authors Israel Herraiz, Jesus M Gonzalez-Barahona, Gregorio Robles
Publication date 2008/5/10
Conference Proceedings of the 2008 international working conference on Mining software repositories
Pages 1-10
Publisher ACM

Description Abstract It has been proposed that software evolution follows a Self-Organized Criticality (SOC) dynamics. This fact is supported by the presence of long range correlations in the time series of the number of changes made to the source code over time. Those long range correlations imply that the current state of the project was determined time ago. In other words, the evolution of the software project is governed by a sort of determinism. But this idea seems to contradict intuition. To explore this apparent contradiction, we have ...

Total citations [Cited by 25](#)



Scholar articles [Determinism and evolution](#)
 I Herraiz, JM Gonzalez-Barahona, G Robles - Proceedings of the 2008 international working ..., 2008
[Cited by 25](#) - [Related articles](#) - [All 7 versions](#)

Dates and citation counts are estimated and are determined automatically by a computer program.

[Help](#)
[Privacy](#)
[Terms](#)
[Provide feedback](#)
[My Citations](#)



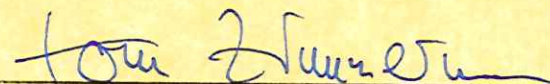
Mining Challenge Award

Israel Herraiz, Jesus M. Gonzalez-Barahona, and Gregorio Robles
(Team Libresoft)

have won the “Change Prediction” category of the
Mining Challenge 2007 with the submission

Forecasting the number of changes in Eclipse using time series analysis

Minneapolis, May 20, 2007



Tom Zimmermann, Mining Challenge Chair

Towards a Simplification of the Bug Report form in Eclipse

Israel Herraiz
Universidad Rey Juan Carlos
Madrid, Spain
herraiz@gsync.es

Jesus M.
Gonzalez-Barahona
Universidad Rey Juan Carlos
Madrid, Spain
jgb@gsync.es

Daniel M. German
University of Victoria
Canada
dmg@uvic.ca

Gregorio Robles
Universidad Rey Juan Carlos
Madrid, Spain
grex@gsync.es

ABSTRACT

We believe that the bug report form of Eclipse contains too many fields, and that for some fields, there are too many options. In this MSR challenge report, we focus in the case of the severity field. That field contains seven different levels of severity. Some of them seem very similar, and it is hard to distinguish among them. Users assign severity, and developers give priority to the reports depending on their severity. However, if users can not distinguish well among the various severity options, they will probably assign different priorities to bugs that require the same priority. We study the mean time to close bugs reported in Eclipse, and how the severity assigned by users affects this time. The results shows that classifying by time to close, there are less clusters of bugs than levels of severity. We therefore conclude that there is a need to make a simpler bug report form.

Categories and Subject Descriptors

D.2.7 [Software Engineering]: Distribution, Maintenance, and Enhancement; K.6.3 [Software management]: Software maintenance

General Terms

Management

Keywords

bug report, bug tracking system, Eclipse, MSR Challenge

1. INTRODUCTION

Bug reporting is an important task for the sustainability of a libre software project [3]. Libre software projects rely on their users for testing and verification: “With enough eyeballs any bug is shallow” (also known as Linus Law) [2]. Bug

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

MSR’08, May 10-11, 2008, Leipzig, Germany.

Copyright 2008 ACM 978-1-60558-024-1/08/05 ...\$5.00.

| Severity | Description |
|-------------|--|
| Blocker | Blocks development and/or testing work. |
| Critical | Crashes, loss of data, severe memory leak. |
| Major | Major loss of function. |
| Normal | (no description given) |
| Minor | Minor loss of function, or other problem where easy workaround is present. |
| Trivial | Cosmetic problem like misspelled words or misaligned text |
| Enhancement | Request for enhancement |

Table 1: Types of severity for bugs. Source: <https://bugs.eclipse.org/bugs/page.cgi?id=fields.html>

reporters (those submitting bugs) are important members of the communities behind libre projects. Hence it is important to constantly evaluate if the defect tracking system is satisfying the needs of both the users and the developers. In particular, we would like to know if it is possible and necessary to simplify the defect reporting form used in Eclipse without losing its effectiveness.

Software developers are usually unable to cope with all the bugs that are notified as they are submitted. They need to prioritize them. They need to determine, for any particular bug, how important it is, and allocate their time and resources according to such prioritization. Bug tracking systems allow reporters to select a severity for the bug they have found. This information is expected to be useful to developers, who do not have to sort through every single report to attend those with a greater severity first and those lesser one.

Bugzilla has become one the most pervasive bug tracking system in the libre software world. It was originally developed by the Mozilla project to help them manage and track defects. In addition to Eclipse, Bugzilla is used by other well-known projects such as GNOME and KDE. In Bugzilla every bug is labeled with a *severity* attribute which can take one of the seven types described in table 1. Those definitions are part of the default installation of Bugzilla, and are included in the bug reporting guidelines of Eclipse.

One of the major problems when assigning severity to a bug is that a reporter might not be the best qualified to de-

Summarizing, some of the requirements of the method are not completely fulfilled for the raw data used in this report. This supposes some threats to the validity of the study. We have tested some of the threats, and have found that the results are similar that using the raw data. In any case, the study should use a treatment for the data, in order to obtain more coherent populations. Furthermore, we have used aggregated data. The study should be repeated using a breakdown process, hence obtaining more coherent groups of bugs (for instance, studying each component separately).

We plan to do so and overcome the possible threats to the validity of this study in a further work.

6. CONCLUSIONS AND FURTHER WORK

We believe that the report forms of Bugzilla (the bug tracking system used by Eclipse) are too complex. There are too many fields, and for each field, too many options, and that some of these options can be removed without affecting the way bugs are handled.

In this study, we have focused in the fields for severity and priority. We demonstrate that the severity of defects can be reduced from seven options to three, and that priority can be reduced from five options to three.

When properly used the severity field provides valuable information to the developers. Unfortunately not every bug reporter is capable of using the current classification. Perhaps a solution is to leave this responsibility to a bug master: the report classifies the bug based on three categories (important, non-important, and request for enhancement), and the bug master further classifies the bug using the current seven.

While developers are using only three priorities, not all developers are using them in a consistent manner. Some never use priority 1 and 2, and some never use 4 and 5. This can lead to confusion. We strongly recommend that the number of priorities is reduced to 3: high, medium and low.

We have also observed that there are two different patterns of use of the priority field: few developers use them to subclassify the severity field (they will classify each of the critical types with P1, P2... etc; effectively ranking how each of the bugs in each severity should be handled); while others use them independently of the severity field (P1 or P2 will always be used for highly critical bugs and P4 or P5 will always be used for those with a very low severity). This lack of uniformity might be confusing if one defect has to be passed from one developer to another, and should be addressed.

There is at least another practical implication. Bugzilla designers tried to make Bugzilla universal. In order to fulfill this goal, they created many different levels of severity, priority, etc, and created many possible fields for the report forms. But people in Eclipse are not using them. Bugzilla designers might not be getting any feedback on how other projects use their system. For instance, if Bugzilla developers would find that most of the bugs severity may be labeled only with three categories, they would have changed it time ago (or at least make it customizable).

It is also interesting to mention that the defect tracking system of FreeBSD has only three levels for severity and three for priority. We would like to get access to the Bugzilla databases of several Mozilla Foundation projects (the original intended users of Bugzilla), and any other project using it as its defect tracking system to evaluate and compare how they use the bug severity and priority fields. However, bug tracking databases are not usually made available to researchers and third parties. This is unfortunate. Making those databases available would help understand how they are used, and, as this paper does, suggest improvements.

7. REFERENCES

- [1] J. Maindonald and J. Braun. *Data Analysis and Graphics using R*. Cambridge Series in Statistical and Probabilistic Mathematics. Cambridge University Press, 2006.
- [2] E. Raymond. *The Cathedral & the Bazaar*. O'Reilly, 1999.
- [3] L. Villa. How GNOME learned to stop worrying and love the bug. In *Talk at the Ottawa Linux Symposium*, Ottawa, July 2003.
<http://tieguy.org/talks/OLS-2003-html/>.
- [4] L. Villa. Why everyone needs a bugmaster. In *Talk at linux.conf.au*, Canberra, April 2005.
<http://tieguy.org/talks/LCA-2005-paper-html/>.

Web Images More... jgbarah@gmail.com

Google

Scholar ← Edit 🗑 Export ▾



Jesus M. Gonzalez-Barahona

Towards a simplification of the bug report form in eclipse

[\[PDF\] from herraiz.org](#)

Authors: Israel Herraiz, Daniel M German, Jesus M Gonzalez-Barahona, Gregorio Robles
Publication date: 2008/5/10
Conference: Proceedings of the 2008 international working conference on Mining software repositories
Pages: 145-148
Publisher: ACM

Description: Abstract We believe that the bug report form of Eclipse contains too many fields, and that for some fields, there are too many options. In this MSR challenge report, we focus in the case of the severity field. That field contains seven different levels of severity. Some of them seem very similar, and it is hard to distinguish among them. Users assign severity, and developers give priority to the reports depending on their severity. However, if users can not distinguish well among the various severity options, they will probably assign different priorities to ...

Total citations [Cited by 46](#)



Scholar articles [Towards a simplification of the bug report form in eclipse](#)
I Herraiz, DM German, JM Gonzalez-Barahona... - Proceedings of the 2008 international working ..., 2008
[Cited by 46](#) - [Related articles](#) - [All 3 versions](#)

Dates and citation counts are estimated and are determined automatically by a computer program.

[Help](#) [Privacy](#) [Terms](#) [Provide feedback](#) [My Citations](#)

Using Social Network Analysis Techniques to Study Collaboration between a FLOSS Community and a Company

Juan Martinez-Romo¹, Gregorio Robles², Jesus M. Gonzalez-Barahona²,
and Miguel Ortuño-Perez²

¹ Dpto. Lenguajes y Sistemas Informaticos, E.T.S.I. Informatica (UNED)
juaner@lsi.uned.es

² Grupo de Sistemas y Comunicaciones, Universidad Rey Juan Carlos
{grex,jgb,mortuno}@gsyc.esct.urjc.es

Abstract. Because of the sheer volume of information available in FLOSS repositories, simple analysis have to face the problems of filtering the relevant information. Hence, it is essential to apply methodologies that highlight that information for a given aspect of the project. In this paper, some techniques from the social sciences have been used on data from version control systems to extract information about the development process of FLOSS projects with the aim of highlighting several processes that occur in FLOSS projects and that are difficult to obtain by other means. In particular, the collaboration between the FLOSS community and a company has been studied by selecting two projects as case studies. The results highlight aspects such as efficiency in the development process, release management and leadership turnover.

1 Introduction

Software projects are usually the collective work of many developers. In most cases, and especially in the case of large projects, those developers are formally organised in a well defined (usually hierarchical) structure, with clear guidelines about how to interact with each other, and the procedures and channels to use. Each team of developers is assigned to certain modules of the system, and only in rare cases they work outside their *territory*. However, this is usually not the case in FLOSS projects, where only loose (if any) formal structures can be recognised. On the contrary, FLOSS developers usually have access to any part of the software, and even in the case of large projects they can *move* more or less freely from one module to another with only some restrictions imposed by the common uses in the project, and the rules on which developers themselves have agreed. A large amount of spontaneous interaction structures arise, evolve and disappear without the intervention of a central control, yielding complex networks.


Among complex networks, social network analysis (SNA) appear as a method for analysing the structure and interactions of people and groups of

Please use the following format when citing this chapter:

Martinez-Romo, J., Robles, G., Gonzalez-Barahona, M. and Ortuño-Perez, M., 2008, in IFIP International Federation for Information Processing, Volume 275; *Open Source Development, Communities and Quality*; Barbara Russo, Ernesto Damiani, Scott Hissam, Björn Lundell, Giancarlo Succi; (Boston: Springer), pp. 171–186.

19. Audris Mockus and Lawrence G. Votta. Identifying reasons for software changes using historic databases. In *Proceedings of the International Conference on Software Maintenance*, pages 120–130, October 2000.
20. Mark E. J. Newman. Scientific collaboration networks: I. network construction and fundamental results. *Phys. Rev. E* 64, 016131, 2001.
21. Gregorio Robles. *Empirical Software Engineering Research on Libre Software: Data Sources, Methodologies and Results*. PhD thesis, Escuela Superior de Ciencias Experimentales y Tecnología, Universidad Rey Juan Carlos, 2006.
22. Gregorio Robles, Jesus M. Gonzalez-Barahona, and Martin Michlmayr. Evolution of volunteer participation in libre software projects: evidence from Debian. In *Proceedings of the 1st International Conference on Open Source Systems*, pages 100–107, Genoa, Italy, July 2005.
23. Gregorio Robles, Stefan Koch, and Jesus M. Gonzalez-Barahona. Remote analysis and measurement of libre software systems by means of the CVSAAnaly tool. In *Proceedings of the 2nd ICSE Workshop on Remote Analysis and Measurement of Software Systems (RAMSS)*, pages 51–56, Edinburgh, Scotland, UK, 2004.
24. Gregorio Robles, Juan Julian Merelo, and Jesus M. Gonzalez-Barahona. Self-organized development in libre software: a model based on the stigmergy concept. In *Proceedings of the 6th International Workshop on Software Process Simulation and Modeling (ProSim 2005)*, St.Louis, Missouri, USA, May 2005.
25. Gert Sabidussi. The centrality index of a graph. *Psychometrika* 31, 581-606, 1996.
26. Yuwan Ye, Kumiyo Nakakoji, Yasuhiro Yamamoto, and Kouichi Kishida. The co-evolution of systems and communities in Free and Open Source software development. In Stefan Koch and Stefan Koch, editors, *Free/Open Source Software Development*, pages 59–82. Idea Group Publishing, Hershey, Pennsylvania, USA, 2004.
27. Thomas Zimmermann, Peter Weissgerber, Stephan Diehl, and Andreas Zeller. Mining version histories to guide software changes. *IEEE Transactions on Software Engineering*, 31(6):429–445, June 2005.

Web Images More... jgbarah@gmail.com



Scholar



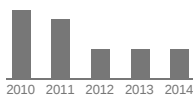
Jesus M. Gonzalez-Barahona

Using social network analysis techniques to study collaboration between a floss community and a company [\[PDF\] from uned.es](#)

Authors Juan Martinez-Romo, Gregorio Robles, Jesus M Gonzalez-Barahona, Miguel Ortuño-Perez
Publication date 2008/1/1
Book Open Source Development, Communities and Quality
Pages 171-186
Publisher Springer US

Description Abstract Because of the sheer volume of information available in FLOSS repositories, simple analysis have to face the problems of filtering the relevant information. Hence, it is essential to apply methodologies that highlight that information for a given aspect of the project. In this paper, some techniques from the social sciences have been used on data from version control systems to extract information about the development process of FLOSS projects with the aim of highlighting several processes that occur in FLOSS projects and that are difficult ...

Total citations [Cited by 22](#)



Scholar articles [Using social network analysis techniques to study collaboration between a floss community and a company](#)
 J Martinez-Romo, G Robles, JM Gonzalez-Barahona... - Open Source Development, Communities and Quality, 2008
[Cited by 22](#) - [Related articles](#) - [All 12 versions](#)
[Using Social Network Analysis Techniques to Study Collaboration between a FLOSS Community and a Company](#) ★
 G Robles, JM Gonzalez-Barahona, J Martinez-Romo - IFIP International Federation for Information Processing, 2010

Dates and citation counts are estimated and are determined automatically by a computer program.

[Help](#)
[Privacy](#)
[Terms](#)
[Provide feedback](#)
[My Citations](#)

On the analysis of contributions from privileged users in virtual open communities

Felipe Ortega, Daniel Izquierdo-Cortazar, Jesus M. Gonzalez-Barahona and Gregorio Robles
 GSyC/LibreSoft
 Universidad Rey Juan Carlos
 Tulipan sn, 28933, Mostoles, Madrid. (SPAIN)
 {jfelipe,dizquierdo,jgb,grex}@gsyc.es

Abstract

Collaborative projects built around virtual communities on the Internet have gained momentum over the last decade. Nevertheless, their rapid growth rate rises some questions: which is the most effective approach to manage and organize their content creation process? Can these communities scale, controlling their projects as their size continues to grow over time? To answer these questions, we undertake a quantitative analysis of privileged users in FLOSS development projects and in Wikipedia. From our results, we conclude that the inequality level of user contributions in both types of initiatives is remarkably distinct, even though both communities present almost identical patterns regarding the number of distinct contributors per file (in FLOSS projects) or per article (in Wikipedia). As a result, totally open projects like Wikipedia can effectively deal with faster growing rates, while FLOSS projects may be affected by bottlenecks on committers who play critical roles.

1. Introduction

Over the last decade, we have witnessed the advent of radically new trends regarding the way virtual communities start up and evolve on the Internet. The so-called Web 2.0 technologies [27] have definitely changed the way we conceive and implement collaborative projects. Forums, blogs, RSS channels, *mashups*, wikis, etc. have moved the focus of the content creation process from webmasters and service providers to final users, unleashing the real power of a worldwide network connecting a global village.

Collaborative organization has been a key feature of many of the most important collective initiatives and projects along the Internet history. Almost all of them, though, have been focused on the development of *Free, Libre and Open Source Software* (FLOSS): GNU/FSF, Linux,

Apache, Mozilla, etc. are prominent examples of these open, collaborative initiatives, whose working philosophy was described in detail in a seminal work by Raymond [30], and further explored by Coffin [9]. However, Web 2.0 technologies expanded the range of these collective initiatives to include virtually any kind of *intangible content* that we can represent in the digital world: images, sounds and music clips, video content, information repositories, help forums, personal blogs, and even universal encyclopaedias like Wikipedia (<http://www.wikipedia.org>).

Some authors [14], [4] defend the thesis that this innovative approach in collaborative virtual communities will continue to increase its influence in due course. James Surowiecki explores in [37] the implications of collaborative content creation initiatives in the way we understand collective intelligence, and how it shapes and affects many aspects of our new networked reality. For sure, large-scale projects such as Wikipedia, involving millions of users contributing in many different languages, compiling more than 8 million articles in its top-ten language editions, challenge any limits previously foreseen by the original creator of *the wiki concept* [22].

Presently, open content initiatives face some of the same evolution paths already followed by FLOSS development projects some years ago. In this context, some natural questions show up: can we find any similarities between FLOSS development projects and collaborative, open content creation initiatives? Does the non-technical nature of open content cause the attraction of a higher number of contributors? Does it affect the frequency and distribution of contributions from project participants? Can these projects scale effectively, in order to avoid becoming victims of their own success?

In this paper, we tackle these questions analyzing contributions from privileged users in FLOSS development projects, as well as in Wikipedia, the epitome of open content creation initiatives at present time. With *privileged*

- [7] C. Castillo-Salgado et al. Measuring health inequalities: Gini coefficient and concentration index. *Epidemiological Bulletin*, 22(1), Mar. 2001.
- [8] A. Ciffolilli. Phantom authority, selfselective recruitment and retention of members in virtual communities: The case of wikipedia. *First Monday*, 8(12), December 2003.
- [9] J. Coffin. Analysis of open source principles in diverse collaborative communities. *First Monday*, 11(6), 2006.
- [10] M. J. Crawley. *The R Book*. Wiley, Chichester, June 2007.
- [11] K. Crowston and J. Howison. The social structure of open source software development teams. In *Proceedings of the International Conference on Information Systems*, Seattle, WA, USA, 2003.
- [12] K. Crowston, B. Scozzi, and S. Buonocore. An explorative study of open source software development structure. In *Proceedings of the ECIS*, Naples, Italy, 2003.
- [13] T. DeMarco and T. Lister. *Peopleware : Productive Projects and Teams, 2nd Ed.* Dorset House Publishing Company, Incorporated, 1999.
- [14] C. Dibona, M. Stone, and D. Cooper. *Open Sources 2.0 : The Continuing Evolution*. O'Reilly Media, Inc., October 2005.
- [15] R. Dorfman. A formula for the gini coefficient. *The Review of Economics and Statistics*, (61):146–149, Mar. 1979.
- [16] K. Fogel. *Producing Open Source Software : How to Run a Successful Free Software Project*. O'Reilly Media, Inc., 2005.
- [17] D. M. German. The GNOME project: a case study of open source, global software development. *Journal of Software Process: Improvement and Practice*, 8(4):201–215, 2004.
- [18] D. M. German. Using software trails to reconstruct the evolution of software. *Journal of Software Maintenance and Evolution: Research and Practice*, 16(6):367–384, 2004.
- [19] C. Gini. On the measure of concentration with especial reference to income and wealth. In *Cowless Commission*, 1936.
- [20] P. Giuri, M. Ploner, F. Rullani, and S. Torrisi. Skills, division of labor and performance in collective inventions. evidence from the open source software. LEM Papers Series 2004/19, Laboratory of Economics and Management (LEM), Sant'Anna School of Advanced Studies, Pisa, Italy, Oct 2004.
- [21] A. Kittur, B. A. Pendleton, B. Suh, and T. Mytkowicz. Power of the few vs. wisdom of the crowd: Wikipedia and the rise of the bourgeoisie. In *Proceedings of the 25th Annual ACM Conference on Human Factors in Computing Systems (CHI 2007)*. ACM, April 2007.
- [22] B. Leuf and W. Cunningham. *The Wiki Way: Collaboration and Sharing on the Internet*. Addison-Wesley Professional, April 2001.
- [23] L. Lopez, G. Robles, J. M. G. Barahona, and I. Herraiz. Applying social network analysis techniques to community-driven libre software projects. *International Journal of Information Technology and Web Engineering*, 1(3):27–48, July-September 2006.
- [24] A. J. Lotka. The frequency distribution of scientific productivity. *Journal of the Washington Academy of Sciences*, 16(12):317–324, 1926.
- [25] M. Michlmayr. Managing volunteer activity in free software projects. In *Proceedings of the USENIX 2004 Annual Technical Conference, FREENIX Track*, pages 93–102, Boston, USA, 2004.
- [26] J. A. Mills et al. Statistical inference via bootstrapping for measures of inequality. *Epidemiological Bulletin*, 22(12), Mar./Apr. 1997.
- [27] T. O'Reilly. O'reilly – what is web 2.0.
- [28] F. Ortega and J. M. Gonzalez-Barahona. Quantitative analysis of the wikipedia community of users. In *WikiSym '07: Proceedings of the 2007 international symposium on Wikis*, pages 75–86, New York, NY, USA, 2007. ACM.
- [29] R Development Core Team. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria, 2008. ISBN 3-900051-07-0.
- [30] E. S. Raymond. *The Cathedral & the Bazaar*. O'Reilly, January 2001.
- [31] B. R.B., H. S.S., T. J.E., and P. D.A. Comparison of skewness coefficient, coefficient of variation, and gini coefficient as inequality measures within populations. *Oecologia*, 78(3):394–400, Mar. 2004.
- [32] J. M. Reagle. Do as i do: authorial leadership in wikipedia. In *WikiSym '07: Proceedings of the 2007 international symposium on Wikis*, pages 143–156, New York, NY, USA, 2007. ACM.
- [33] G. Robles and J. M. Gonzalez-Barahona. Contributor turnover in libre software projects. In *Proceedings of the 2nd International Conference on Open Source Systems*, Como, Italy, July 2006.
- [34] F. Rullani. Dragging developers towards the core. how the free/libre/open source software community enhances developers' contribution. LEM Papers Series 2006/22, Laboratory of Economics and Management (LEM), Sant'Anna School of Advanced Studies, Pisa, Italy, Sep 2006.
- [35] J. Sandred. *Managing Open Source Projects*. Wiley Computer Publishing, 2001.
- [36] S. Spek, E. Postma, and J. Herik. Wikipedia: organisation from a bottom-up approach, Nov 2006.
- [37] J. Surowiecki. *The Wisdom of Crowds*. Anchor, August 2005.
- [38] V. Thomas, Y. Wang, and X. Fan. Measuring education inequality - gini coefficients of education. Policy Research Working Paper Series 2525, The World Bank, Jan. 2001.
- [39] W. N. Venables and B. D. Ripley. *Modern Applied Statistics with S*. Springer, September 2003.
- [40] J. Voss. Measuring wikipedia. In *International Conference of the International Society for Scientometrics and Informetrics : 10th*. ISSI, July 2005.
- [41] A. Wagstaff et al. On the measurements of inequalities in health. *Soc. Sci. Med.*, 33(5):545–577, 1991.
- [42] M. Wattenberg, F. Viégas, and K. Hollenbach. Visualizing activity on wikipedia with chromograms. pages 272–287. 2007.

Using Software Archaeology To Measure Knowledge Loss in Software Projects Due To Developer Turnover *

Daniel Izquierdo-Cortazar, Gregorio Robles, Felipe Ortega and Jesus M. Gonzalez-Barahona
 GSyC/LibreSoft
 Universidad Rey Juan Carlos (Madrid, Spain)
 {dizquierdo, grex, jfelipe, jgb}@gsyc.escet.urjc.es

Abstract

Developer turnover can result in a major problem when developing software. When senior developers abandon a software project, they leave a knowledge gap that has to be managed. In addition, new (junior) developers require some time in order to achieve the desired level of productivity. In this paper, we present a methodology to measure the effect of knowledge loss due to developer turnover in software projects. For a given software project, we measure the quantity of code that has been authored by developers that do not belong to the current development team, which we define as orphaned code. Besides, we study how orphaned code is managed by the project. Our methodology is based on the concept of software archaeology, a derivation of software evolution. As case studies we have selected four FLOSS (free, libre, open source software) projects, from purely driven by volunteers to company-supported. The application of our methodology to these case studies will give insight into the turnover that these projects suffer and how they have managed it and shows that this methodology is worth being augmented in future research.

Keywords: *developer turnover, orphaning, software risk management, software archaeology*

1. Introduction

Software development is an activity intense in human resources. The work of many developers is required to create almost any non-trivial piece of code.

*This work has been funded in part by the European Commission, under the FLOSSMETRICS (FP6-IST-5-033547), QUALOSS (FP6-IST-5-033547) and QUALIPSO (FP6-IST-034763) projects, and by the Spanish CICYT, project SobreSalto (TIN2007-66172).

The lifespan of a software system can range from several years to decades. In such scenarios, the development team in charge of the software may suffer from turnover: old developers leave while new developers join the project. With the abandonment of old, senior developers, projects lose human resources experienced both with the details of the software system and with the organizational and cultural circumstances of the project. New developers will need some time to become familiar with both issues. In this regard, the time for volunteers to become core contributors to FLOSS ¹ projects has been measured to be 30 months in mean, since their first contribution [11].

Although maintaining the current development team could be thought as a plausible solution to mitigate this problem, turnover is usually unavoidable. Being a highly intellectual work, developers have a tendency to lose the original motivation on the software system as time passes by, and they have the natural desire to search for new objectives. In this sense, high turnovers have been observed in most large FLOSS projects, where several *generations* of successive development teams have been identified [21]. Although these environments are partially, if not mostly, driven by volunteers, turnover in industrial environments is also high.

In this paper, we present a methodology to measure the effect of developer turnover in software projects. It is based on quantifying the knowledge that developers contribute to a project based on the number of lines of code written for it. In case developers leave, their lines become *orphaned*. The amount of *orphaned* lines can be considered as a measure of the knowledge that the

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


common situation for many managers that use, but do not participate in FLOSS projects. Insiders could use this information to avoid risks (e.g. modules plenty of *orphaned* lines where the current *core* team has never worked on) or to strengthen the maintenance activities of a project (e.g. detecting those "dead" modules).

All in all, the presented methodology, based on the idea of *software archaeology*, provides useful information to address the issue of developer turnover in software projects, even if some limitations, both technical (*gate-keeper* effect, *granularity*, etc.) and conceptual (*old code vs. young code*, the effect of the quality, *full memory*), have still to be dealt with in the future.

References

- [1] B. W. Boehm. Software risk management: Principles and practices. *IEEE Softw.*, 8(1):32–41, 1991.
- [2] B. W. Boehm and T. DeMarco. Guest editors' introduction: Software risk management. *IEEE Software*, 14(3):17–19, 1997.
- [3] S. A. Conger. *The New Software Engineering*. International Thomson Publishing, 1994.
- [4] T. De Marco and T. Lister. *Peopleware : Productive Projects and Teams, 2nd Ed.* Dorset House Publishing Company, Incorporated, 1999.
- [5] S. G. Eick, T. L. Graves, A. F. Karr, J. S. Marron, and A. Mockus. Does code decay? Assessing the evidence from change management data. *IEEE Transactions on Software Engineering*, 27(1):1–12, 2001.
- [6] K. Fogel. *Producing Open Source Software : How to Run a Successful Free Software Project*. O'Reilly Media, Inc., 2005.
- [7] X. Ge, Y. Dong, and K. Huang. Shared knowledge construction process in an open-source software development community: an investigation of the gallery community. In *ICLS '06: Proceedings of the 7th international conference on Learning sciences*, pages 189–195. International Society of the Learning Sciences, 2006.
- [8] D. M. German. The GNOME project: a case study of open source, global software development. *Journal of Software Process: Improvement and Practice*, 8(4):201–215, 2004.
- [9] D. M. German. Using software trails to reconstruct the evolution of software. *Journal of Software Maintenance and Evolution: Research and Practice*, 16(6):367–384, 2004.
- [10] T. Gırba, A. Kuhn, M. Seeberger, and S. Ducasse. How developers drive software evolution. In *Proceedings of the International Workshop on Principles in Software Evolution*, pages 113–122, Lisboa, Portugal, September 2005.
- [11] I. Herraiz, G. Robles, J. J. Amor, T. Romera, and J. M. González-Barahona. The processes of joining in global distributed software projects. In *GSD '06: Proceedings of the 2006 international workshop on Global software development for the practitioner*, pages 27–33, New York, NY, USA, 2006. ACM Press.
- [12] A. Hunt and D. Thomas. Software Archaeology. *IEEE Software*, 19(2):20–22, March/April 2002.
- [13] C. Hutchison. Personal knowledge, team knowledge, real knowledge. *EUROCON'2001, Trends in Communications, International Conference on.*, 1:247–250 vol.1, 2001.
- [14] M. Michlmayr and B. M. Hill. Quality and the reliance on individuals in free software projects. In *Proceedings of the 3rd Workshop on Open Source Software Engineering*, pages 105–109, Portland, USA, 2003.
- [15] M. Michlmayr, G. Robles, and J. M. Gonzalez-Barahona. Volunteers in large libre software projects: A quantitative analysis over time. In S. K. Sowe, I. G. Stamelos, and I. Samoladas, editors, *Emerging Free and Open Source Software Practices*, pages 1–24. Idea Group Publishing, Hershey, Pennsylvania, USA, 2007.
- [16] 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.
- [17] P. Oman and J. Hagemeister. Metrics for assessing a software system's maintainability. *Software Maintenance, 1992. Proceedings., Conference on*, pages 337–344, Nov 1992.
- [18] T. Otte, R. Moreton, and H. D. Knoell. Applied quality assurance methods under the open source development model. In *COMPSAC*, pages 1247–1252, 2008.
- [19] D. L. Parnas. Software aging. In *Proceedings of the International Conference on Software Engineering*, pages 279–287, Sorrento, Italy, May 1994.
- [20] E. S. Raymond. The cathedral and the bazaar. *First Monday*, 3(3), March 1998. http://www.firstmonday.dk/issues/issue3_3/raymond/.
- [21] G. Robles. Contributor turnover in libre software projects. In *Proceedings of the Second International Conference on Open Source Systems*, 2006.
- [22] 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*, pages 100–107, Genoa, Italy, July 2005.
- [23] F. V. Rysseberghe and S. Demeyer. Studying software evolution information by visualizing the change history. In *International Conference on Software Maintenance*, pages 328–337, 2004.
- [24] J. Sandred. *Managing Open Source Projects*. Wiley Computer Publishing, 2001.
- [25] L. Yu. Indirectly predicting the maintenance effort of open-source software: Research articles. *J. Softw. Maint. Evol.*, 18(5):311–332, 2006.

Web Images More... jgbarah@gmail.com



Scholar

[←](#) [Edit](#) [🗑](#) [Export ▾](#)



Jesus M. Gonzalez-Barahona

Using software archaeology to measure knowledge loss in software projects due to developer turnover [\[PDF\]](#) from ifipwg213.org

Authors Daniel Izquierdo-Cortazar, Gregorio Robles, Felipe Ortega, Jesus M Gonzalez-Barahona

Publication date 2009/1/5

Conference System Sciences, 2009. HICSS'09. 42nd Hawaii International Conference on

Pages 1-10

Publisher IEEE

Description Abstract Developer turnover can result in a major problem when developing software. When senior developers abandon a software project, they leave a knowledge gap that has to be managed. In addition, new (junior) developers require some time in order to achieve the desired level of productivity. In this paper, we present a methodology to measure the effect of knowledge loss due to developer turnover in software projects. For a given software project, we measure the quantity of code that has been authored by developers that do not belong ...

Total citations [Cited by 17](#)



Scholar articles [Using software archaeology to measure knowledge loss in software projects due to developer turnover](#)
D Izquierdo-Cortazar, G Robles, F Ortega... - System Sciences, 2009. HICSS'09. 42nd Hawaii ..., 2009
[Cited by 17](#) - [Related articles](#) - [All 10 versions](#)

Dates and citation counts are estimated and are determined automatically by a computer program.

[Help](#) [Privacy](#) [Terms](#) [Provide feedback](#) [My Citations](#)

Evolution of the core team of developers in libre software projects

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

Abstract

In many libre (free, open source) software projects, most of the development is performed by a relatively small number of persons, the “core team”. The stability and permanence of this group of most active developers is of great importance for the evolution and sustainability of the project. In this position paper we propose a quantitative methodology to study the evolution of core teams by analyzing information from source code management repositories. The most active developers in different periods are identified, and their activity is calculated over time, looking for core team evolution patterns.

1. Introduction

Employee turnover is known to be high in the traditional software industry since many years ago [1]. However, in libre software¹ projects the study of developer turnover has not been an active research topic. Most of the attention in this area has been focused on the organizational structure of the projects [2], with little attention to the dynamics of the developers.

A noteworthy contribution in this sense, although it does not address the evolution of developer communities, is the *onion model* [3], which shows how developers and users are positioned in communities. In this model, it is possible to differentiate among core developers (those who have a high involvement in the project), codevelopers (with specific but frequent contributions), active users (contributing only occasionally) and passive users [4, 5].

This position paper shows how to better understand the evolution of the most active group of developers contributing to a libre software project. A specific methodology has been designed to quantitatively characterize a project in the spectrum between these two scenarios, and to visualize more in detail the evolution of the core team. The first steps of this methodology, applied to a few projects, are explained

¹In this paper we will use the term “libre software” to refer both to “free software” and “open source software”.

in [6]. An extension and refinement of the methodology is presented here.

2. Methodology

The methodology used in this study is based on retrieving data about the activity of developers from source code management repositories, which are mined using CVS-Analy [7]. This tool retrieves information about every commit to the repository, and inserts it into a database where it can be conveniently analyzed.

To characterize the evolution of the core team, first the life of the project is split in periods of equal duration. Then for every period i , the most active developers are identified as $CoreTeam_i$. This is done by calculating the number of commits during that period for the most active developers. For each $CoreTeam_i$, its activity is tracked for the rest of the life of the project (before and after period i). Hence, for each period j , the number of commits is calculated for all the developers in $CoreTeam_i$. Finally, the resulting data (that represents the activity of the each $CoreTeam_i$ for all periods) is plotted in several formats, and collapsed into some indexes that allow comparison and classification.

Because of several trade-offs, we have not considered a single time span for periods. For the purposes of the study, usually the most significant results are obtained by dividing the history of the project into 10 or 20 periods.

After considering several alternatives, we have found that fractions of 0.1 and 0.2 (that is, the top 10% and 20%) are large enough to capture developers producing most of the activity (usually more than 50%, reaching in many cases as much as 90% or 95% of the total number of commits).

3. Outputs of the methodology

Our methodology provides both some graphs that help to visualize the results and some data (in the form of arrays and indexes).

The main output of the methodology is the *AbsoluteMatrix*: a squared two dimensional array, with the number of periods as range. Values for each

the corresponding repository. Fortunately, this is the case for a large fraction of libre software projects, including the most relevant ones.

The methodology can be used to rank projects according to their distance to the two extreme cases of “code gods” and “series of generations”, using the produced indexes. But it provides also a lot of insight on the evolution of the core teams, by showing visually (both in graphs and maps) the activity patterns of the developers forming the core team in each period of the life of a project. This information can be used to identify levels of smoothness in transitions, to detect break points in the evolution of the core team, to understand the differences in activity of the core team in different periods, or to estimate unevenness in the contributions of the most active developers when compared to the rest of them.

We have applied the methodology to a relevant case study, using a well-known libre software project. Some factors not specifically discussed in this paper could influence the appropriateness of the methodology. Among them, the relevance of using the number of commits as a proxy for the activity and importance of developers. For validating it, we have studied some other parameters, such as the number of changed lines, without finding meaningful differences. However, an important problem remains open: to which extent other, non-coding activities (such as discussion, writing of documentation, or even mediation between developers) should be considered to better identify the core team of developers. This should be the focus of further research.

Another open field for research is the use of the methodology in classical (non-libre) software projects. The fact that many developers in libre software projects are volunteers can provide very interesting information about the natural behavior of programmers, as these developers are self-selected (i.e., there is no traditional, mandatory task assignment as it can be found in the commercial world). In this regard, one of the findings that should be further researched is the amount of time for turnover. From our limited set of projects we have seen that, for those projects with several generations, the time span for a generation ranges from three to five years. This could be indicative for a programmers moving to a different project to keep his motivation and interest on his work high. Having developers enrolled in companies (such as the cases of Mozilla and Evolution) and volunteer developers in these projects could give further insight to this question in subsequent research.

In any case, from our work we can conclude that the study of the behavior of human resources in libre software projects and in software engineering in general, and the relationship between its join/leave patterns and the evolution of the project, is a field worth to explore. This paper tries to be a first step in this direction, focused on studying its dy-

namics, and on finding how projects cope with the changes caused by it.


6. Acknowledgments

This work has been funded in part by the European Commission, through projects FLOSSMetrics, FP6-IST-5-033982, QUALOSS, FP6-IST-5-033547, and Qualipso, FP6-IST-034763, and by the Spanish CICYT, project Sobre-Salto (TIN2007-66172)

References

- [1] B. W. Boehm, Ed., *Software risk management*. Piscataway, NJ, USA: IEEE Press, 1989.
- [2] D. M. Germn, “The GNOME project: a case study of open source, global software development,” *Journal of Software Process: Improvement and Practice*, vol. 8, no. 4, pp. 201–215, 2004.
- [3] K. Crowston and J. Howison, “The social structure of free and open source software development,” *First Monday*, vol. 10, no. 2, February 2005.
- [4] 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*, vol. 11, no. 3, pp. 309–346, 2002.
- [5] T. Dinh-Trong and J. M. Bieman, “Open source software development: A case study of freebsd,” in *Proceedings of the 10th International Software Metrics Symposium*, Chicago, IL, USA, 2004.
- [6] G. Robles and J. M. González-Barahona, “Contributor turnover in libre software projects,” in *Open Source Systems Conference, June 8-10, 2006, Como, Italy, 2006*, pp. 273–286.
- [7] 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)*, Edinburgh, Scotland, UK, 2004, pp. 51–56.

Web Images More... jgbarah@gmail.com



Scholar ← Edit 🗑 Export ▾



Jesus M. Gonzalez-Barahona

Evolution of the core team of developers in libre software projects

[\[PDF\] from urjc.es](#)

Authors: Gregorio Robles, Jesus M Gonzalez-Barahona, Israel Herraiz
 Publication date: 2009/5/16
 Conference: Mining Software Repositories, 2009. MSR'09. 6th IEEE International Working Conference on
 Pages: 167-170
 Publisher: IEEE

Description: Abstract In many libre (free, open source) software projects, most of the development is performed by a relatively small number of persons, the "core team". The stability and permanence of this group of most active developers is of great importance for the evolution and sustainability of the project. In this position paper we propose a quantitative methodology to study the evolution of core teams by analyzing information from source code management repositories. The most active developers in different periods are identified, ...

Total citations [Cited by 34](#)



Scholar articles [Evolution of the core team of developers in libre software projects](#)
 G Robles, JM Gonzalez-Barahona, I Herraiz - Mining Software Repositories, 2009. MSR'09. 6th IEEE ..., 2009
[Cited by 34](#) - [Related articles](#) - [All 16 versions](#)

Dates and citation counts are estimated and are determined automatically by a computer program.

[Help](#) [Privacy](#) [Terms](#) [Provide feedback](#) [My Citations](#)

A Comprehensive Study of Software Forks: Dates, Reasons and Outcomes

Gregorio Robles and Jesús M. González-Barahona

GSyC/Libresoft, Universidad Rey Juan Carlos
{gregorio.robles,jesus.gonzalez.barahona}@urjc.es

Summary. In general it is assumed that a software product evolves within the authoring company or group of developers that develop the project. However, in some cases different groups of developers make the software evolve in different directions, a situation which is commonly known as a fork. In the case of free software, although forking is a practice that is considered as a last resort, it is inherent to the four freedoms. This paper tries to shed some light on the practice of forking. Therefore, we have identified significant forks, several hundreds in total, and have studied them in depth. Among the issues that have been analyzed for each fork is the date when the forking occurred, the reason of the fork, and the outcome of the fork, i.e., if the original or the forking project are still developed. Our investigation shows, among other results, that forks occur in every software domain, that they have become more frequent in recent years, and that very few forks merge with the original project.

Keywords: free software, open source, forks, forking, social, legal, sustainability, software evolution.

1 Introduction

Issues related to the sustainability of software projects have historically been studied in software engineering in the field of software evolution. However, research on software evolution has always implicitly assumed that development and maintenance of a software is performed by the same organization or group of developers. It is a task of the creators of the software to make it evolve [13].

But in some cases a software project evolves in parallel, lead by different development teams. This is known as "forking". The term fork is derived from the POSIX standard for operating systems: the system call used so that a process generates a copy of itself is called `fork()`. As a consequence, there exist two copies of the process that run independently and may perform different tasks. In analogy to this situation, a software fork happens when there exist two independent software projects, deriving both from the same software source code base.

Forking may happen in proprietary environments, but it is *natural* in free software as the freedom to modify a software and redistribute modifications is part

16. Neville-Neil, G.V.: Think before you fork. *Commun. ACM* 54, 34–35 (2011)
17. Raymond, E.S.: *The New Hacker's Dictionary*, 3rd edn. MIT Press, Cambridge (1996)
18. Raymond, E.S.: *The Cathedral and the Bazaar*. In: Raymond, E.S. (ed.) *Musings on Linux and Open Source by an Accidental Revolutionary* pp. 79–135. O'Reilly (1999)
19. Scacchi, W.: Computer game mods, modders, modding, and the mod scene. *First Monday* 15(5) (2010)
20. Schweik, C.M., English, R., Paienjtton, Q., Haire, S.: Success and Abandonment in Open Source Commons: Selected Findings from an Empirical Study of Sourceforge.net Projects. In: *Proceedings of the Sixth International Conference on Open Source Systems 2010*, pp. 91–101 (2010)
21. Weber, S.: *The Success of Open Source*. Harvard University Press (April 2004)
22. Xie, G., Chen, J., Neamtiu, I.: Towards a better understanding of software evolution: An empirical study on open source software. In: *Proceedings of the International Conference on Software Maintenance*, pp. 51–60. IEEE (2009)
23. Yamamoto, T., Matsushita, M., Kamiya, T., Inoue, K.: Measuring Similarity of Large Software Systems Based on Source Code Correspondence. In: Bomarius, F., Komi-Sirviö, S. (eds.) *PROFES 2005*. LNCS, vol. 3547, pp. 530–544. Springer, Heidelberg (2005)
24. Yu, L., Schach, S.R., Chen, K., Heller, G.Z., Offutt, A.J.: Maintainability of the kernels of open-source operating systems: A comparison of Linux with FreeBSD, NetBSD, and OpenBSD. *Journal of Systems and Software* 79(6), 807–815 (2006)



The Eighth International Conference on Open Source Systems



BEST PAPER AWARD

Presented to

Gregorio Robles and Jesus M. Gonzalez-Barahona

“A Comprehensive Study of Software Forks: Dates, Reasons and Outcomes”

11th September 2012

Walt Scacchi

General co-chair

Tommi Mikkonen

General co-chair


Imed Hammouda

Program co-chair

Björn Lundell

Program co-chair

Web Images More... jgbarah@gmail.com



Scholar



Jesus M. Gonzalez-Barahona

A comprehensive study of software forks: Dates, reasons and outcomes

[TXT] from msidllc.com

Authors: Gregorio Robles, Jesús M González-Barahona

Publication date: 2012/1/1

Book: Open Source Systems: Long-Term Sustainability

Pages: 1-14

Publisher: Springer Berlin Heidelberg

Description: Summary In general it is assumed that a software product evolves within the authoring company or group of developers that develop the project. However, in some cases different groups of developers make the software evolve in different directions, a situation which is commonly known as a fork. In the case of free software, although forking is a practice that is considered as a last resort, it is inherent to the four freedoms. This paper tries to shed some light on the practice of forking. Therefore, we have identified significant forks, several ...

Total citations: [Cited by 16](#)



Scholar articles: [A comprehensive study of software forks: Dates, reasons and outcomes](#)
 G Robles, JM González-Barahona - Open Source Systems: Long-Term Sustainability, 2012
[Cited by 16](#) - [Related articles](#) - [All 7 versions](#)

Dates and citation counts are estimated and are determined automatically by a computer program.

[Help](#)
[Privacy](#)
[Terms](#)
[Provide feedback](#)
[My Citations](#)