A Statistical Examination of the Evolution and Properties of Libre Software

Israel Herraiz
GSyC/Libresoft
Universidad Rey Juan Carlos, Spain
herraiz@gsyc.urjc.es

Abstract

How and why does software evolve? This question has been under study since almost 40 years ago, and it is still a subject of controversy. In the seventies, Meir M. Lehman formulated the laws of software evolution*, a first attempt to characterize the dynamics of the evolution of software. With the raise of the libre (free / open source) software development phenomenon, some cases that do not fulfill those laws have appeared. Are Lehman's laws valid in the case of libre software development? Is it possible to design an universal theory for software evolution? And if it is, how? This thesis is a large-scale empirical and statistical approach to analyze the properties and evolution of libre software, using publicly available data sources, hence enabling repeatability of the results and third parties verification, fundamental aspects of any empirical study. The main results are that a small subset of basic size metrics are enough to characterize a software system, software systems are self-similar, and software evolution is a short range correlated (short memory) process.*

1. Introduction

Software evolution was born as field of research 40 years ago, and keeps being an intense matter of study. Meir M. Lehman, the pioneer in software evolution studies, coedited a book in 1985 where the *laws of software evolution* were stated [8]. That set of laws has evolved over the years to keep synchronized with newer software technologies and development methodologies, and many experiences and case studies supporting them have been reported [5].

However, these laws have faced some controversial studies that showed cases where the laws were not fulfilled [12]. The most recent of those cases are related to libre (free, open source) software [1, 14]. The case of libre software development is paradoxical if we assume the laws of software evolution to be true. Libre software is usually developed in communities distributed all over the world, communicating through the Internet. In many cases, the members of the development team have not even met in person. Those people come from different cultures, speak different languages, work in different time zones, do not write requirements but take them as implicit, and yet they manage to deliver quality products. Another difference is that the laws of software evolution assume the lifetime of a software project to be divided in two clear and different sets: development and evolution, while in libre software both development and evolution activities happen at the same time, and during the whole lifetime of the project. Another hint about the in unsuitability of the laws for the case of libre software is the recommendation of Lehman about keeping the interval between releases as long as possible so the growth rate is maintained under a safe value [7], that is in direct conflict with the *release early, release often* principle [13] massively followed by community-driven libre software projects, and proven to be successful in practice.

This conflict is not surprising if we think of the dimension of the empirical studies which the laws rely on. The first sets of laws were only based on seven case studies (more details may be found in the dissertation [2]). Still in the nineties, the lack of empirical basis for software evolution was a subject of research [3].

This thesis derives from all the above points. The original goal of Lehman was to obtain a theory for software evolution, being the laws a first approach to such a theory, that lacked the necessary empirical support to become a theory. The search for a theory is a search for conserved quantities and invariant properties. Thanks to the availability of thousands of open software repositories in the libre software community, it is possible to search for those invariants following a statistical approach and basing on a large set of case studies.

Indeed, this approach was suggested by Lehman [6], but was not possible at the time because of the difficulties to access real data about software projects. Therefore the main goal of this thesis is:

To study the evolution and properties of a sufficiently large amount of libre software projects, by means of empirical studies and using a statistical approach, to determine whether or not some commonalities do exist, that could be used to formulate a universal theory of software evolution.

2. Data Sources

In this thesis, we have used three main data sources: the set of packages of FreeBSD, change records of a large amount of projects hosted in SourceForge.net, and the evolution in size for three selected case studies.

The first dataset was obtained extracting the source code of a sample of packages of FreeBSD (*ports*, in the terminology of FreeBSD), a well known software distribution. Ports contain different setup files that are able to retrieve the source code needed to build the package directly from the original upstream authors. This makes it possible to obtain original *upstream* source code, without any patching applied by the distributors. Besides source code, ports also provide meta-information, like the domain of application, allowing for more refined analysis. This dataset contained source code belonging to $12,010$ software projects, with more than a million files. Some of the analysis were done only on C source code files. That subsample of C source code files belonged to $6,556$ software projects, with nearly half a million files.

The second dataset contained the record history of some software projects hosted in SourceForge.net (SF.net), a popular hosting service for libre software. SF.net contains hundreds of thousands of projects. However, many of them are inactive projects. We selected projects with at least three developers, one year of active history and with a CVS repository. This reduced the size of the sample to $3,821$ projects. For all those projects, we retrieved the time series of daily changes. This is, the daily number of changes for each project.

Finally, the third dataset contained only three case studies (the FreeBSD and NetBSD kernels, and PostgreSQL, a databases management system), that were used to fit some predictive models, and to compare them. The predicted parameter was size, measured in Source Lines of Code (SLOC), or more precisely, the time series of the daily size measured in SLOC. The fitting set was composed of more than 10 years of history, and the test set was the last year of history of the projects.

3. Research Questions

In this section, we show the different research questions addressed in this thesis, each one in a subsection with the following structure: research question, description of the methodology, summary of results and implications of those results.

3.1 First Question: Validity of the Laws

Are the laws of software evolution valid for the case of libre software?

In this research question, wee tried to determine whether or not the laws of software evolution are valid for the case of libre software. This question has been raised some times in the past. For instance, Godfrey and Tu [1] found that Linux grows with a superlinear pattern, and therefore it does not fulfill the laws of software evolution. Lehman argued that Godfrey and Tu's study was not comparable with other studies because the metrics used in each study were different [9]. In the study by Godfrey and Tu, size was measured in SLOC, and Lehman always used number of source code files (termed modules in his papers).

This question was addressed using the first dataset. For all the source code files, we measured several size and complexity metrics, and then correlated the measurements, to determine whether or not they were independent. The correlation was done only for C source code files.

The results of the correlation analysis show that SLOC is highly correlated with all the rest of size and complexity metrics, and therefore any study using number of files or SLOC should report the same results, regardless of the size metric used.

Therefore, all the conflicting cases that have been reported in the literature [1, 14] are comparable to the classical studies by Lehman, and therefore they also are evidences of the invalidity of the laws (at least for the case of libre software).

3.2 Second Question: Metrics to Characterize a Software Product

Which metrics should be used to characterize the evolution of a software product?

The result of the first question also answers this research question: all the size and complexity metrics considered in this thesis are highly correlated (at least, for the case of the C programming language), and thus all the metrics are providing the same information. Among all the metrics, SLOC is popular in the literature (which makes it easier to compare against previous works), and there are tools to measure it for a myriad of programming languages. Therefore, we suggest to use SLOC to characterize a software product.

3.3 Third Question: Shape of the Statistical Distribution of Size

What is the shape of the statistical distribution of software size?

After measuring the size of all the source code files of the first dataset, we estimated the density probability function of software size. We repeated that estimation at different levels of granularity: size of files in SLOC, size of software projects in number of files, and size of domains of applications in number of projects (that fall in that domain of application). We also repeated all measurements at different levels (file, project, domain) in SLOC; for instance, size of every project in number of SLOC (contrarily to number of files like in the previous measurement). With all those data, we estimated the density probability function for each level, with all the size metrics.

The estimation of the density probability function showed that the shape of that function corresponds to a double Pareto distribution. This distribution has been found before in other areas [11], and even some dynamic models have been proposed for those areas [10], which suggest that those models might be adapted for the case of software evolution.

This result has implications for a hypothetical theory of software evolution. The shape of this distribution gives information about the kind of process that generates it [10]. Therefore, any theory of software evolution should reproduce this kind of distributions. In particular, *physical* models for evolution can be proposed based on the processes that generate this kind of distributions.

3.4 Fourth Question: Self-Similarity

How does that shape change with the scale of the measurements (using different entities, metrics and so forth)?

Interestingly, and linking with the third question, the double Pareto distribution appears regardless of the metric used, and regardless of the level of granularity. Thus, the distribution of the size of all the files measured in SLOC is a double Pareto. The distribution of the size of software projects in number of files is also a double Pareto. And the size of domains of application in number of projects is also a double Pareto. This is an evidence of self-similarity in software, which may be an explanation for the fast growth of some libre software projects.

3.5 Fifth Question: Software Evolution Dynamics

What kind of dynamics drive software evolution (short range or long range correlated)?

One crucial aspect of a theory of software evolution is its dynamics. A recent work by Wu found that libre software evolution is driven by a self-organized criticality (SOC) dynamics [15, 16]. Among other implications, process that can be explained by a SOC dynamics are long range correlated.

The methodology is based on calculating the autocorrelation function plot of a time series. Depending on the shape of that plot, we can determine if the time series corresponds to a short range or a long range correlated process.

Using the sample of the daily time series of number of changes for the $3,821$ projects extracted from SF.net, we found that more than 80% of them were short range correlated process, suggesting that SOC may not be a good model for software evolution.

From a practitioner's point of view, this means that when taking decisions about the next steps of a software project, recent events have a greater impact in the future of the project, and therefore should be given more importance that events that took place time ago.

3.6 Sixth Question: Forecasting Software Evolution

What is the best method to predict software evolution?

The fact that most of the projects in the sample extracted from SF.net are short range correlated processes (or ARIMA processes in the time series terminology), can be used to fit accurate models with the purpose of forecasting software evolution.

That kind of models are statistical, and hence non-explanatory. We can not find out why software evolves, but at least can predict the evolution of some parameters (like size) that might be useful from a management point of view. Traditionally, regression models have been used for this task [1, 14, 4].

Using the FreeBSD and NetBSD kernels, and the PostgreSQL database management system, we predicted the last year of history (the daily size) of those projects. The training set was composed of all the history of the projects, but the last year. That last year was forecasted with regression and ARIMA (time series analysis based) models, then the results were compared against the actual data. The mean squared relative error of the regression models was from 7% to 17%. For the ARIMA models, from 1.5% to 4%. Therefore, ARIMA models are much more accurate for forecasting software evolution than regression models.

Again, from a practitioner's point of view, this suggests that regression is not always the best technique for software evolution forecasting or modeling, and that more sophisticated techniques like time series analysis should be used instead.

4. Conclusions and Further Work

This thesis is a large empirical study of the evolution and properties of libre software. All the data sources used for this study are public. Moreover, all the tools, databases and scripts used to make all the calculations have been made available, so this study can be repeated and verified with new case studies[1]. The goal of this thesis is to find commonalities within that large set of software, so they can be used as a basis for an universal theory of software evolution. This has been a long pursued objective in the field of software evolution during decades. Actually, this statistical approach to this problem was proposed as early as 1974 by Lehman [6]. Thanks to the availability of thousands of open software repositories in the libre software world, it is now possible for the first time in history to perform such a large scale study, and more importantly, to make it repeatable and verifiable by third parties. These are crucial requirements in the quest for a theory of software evolution.

The results of this thesis have deep implications for such a theory. First, they show that the laws of software evolution have been invalidated in the case of libre software development. Second, among the results there are some findings that hold for a large amount of case studies (like the correlations among metrics, or the self-similarity), which suggest that those are universal properties of software.

From the practitioner's point of view, this thesis shows that, because of the statistical properties of the dynamics of software evolution, the best methods to forecast the evolution of software projects is time series analysis, and that the recent history of projects have a much deeper influence in the next steps than remote events.

However, much further work is still needed before the theory can be articulated. That software size is distributed like a double Pareto needs a more careful study, and explanatory (*physical*) models should be proposed based on that fact. Other findings, like self-similarity, should also serve as *check points* for a theory, because any explanatory model should reproduce those properties.

5. Acknowledgements

This work was done with the support of the funding provided by the Comunidad de Madrid (the regional government of Madrid) and the European Social Fund, under grant number 01/FPI/0582/2005, and the FP6 funded FLOSS-Metrics (contract IST-5-033982) and QUALOSS (contract FP6-IST-5-033547) projects.

[1] See http://purl.org/net/who/iht/phd

References

[1] 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.

[2] I. Herraiz. *A statistical examination of the evolution and properties of libre software*. PhD thesis, Universidad Rey Juan Carlos, 2008. http://purl.org/net/who/iht/phd.

[3] C. F. Kemerer and S. Slaughter. An empirical approach to studying software evolution. *IEEE Transactions on Software Engineering*, 25(4):493–509, 1999.

[4] S. Koch. Evolution of Open Source Software systems - a large-scale investigation. In *Proceedings of the International Conference on Open Source Systems*, Genova, Italy, July 2005.

[5] M. Lehman and J. Ramil. An overview of some lessons learnt in FEAST. In *Proceedings of the Workshop on Empirical Studies of Software Maintenance*, 2002.

[6] M. M. Lehman. Programs, Cities, Students: Limits to Growth?, 1974. Inaugural lecture, Imperial College of Science and Technology, University of London.

[7] M. M. Lehman. Laws of Program Evolution-Rules and Tools for Programming Management. In *Proceedings of Infotech State of the Art Conference, Why Software Projects Fail*, 1978.

[8] M. M. Lehman and L. A. Belady, editors. *Program evolution. Processes of software change*. Academic Press Professional, Inc., San Diego, CA, USA, 1985.

[9] M. M. Lehman, J. F. Ramil, and U. Sandler. An approach to modelling long-term growth trends in software systems. In *Internation Conference on Software Maintenance*, pages 219–228, Florence, Italy, 2001. IEEE Computer Society.

[10] M. Mitzenmacher. Dynamic models for file sizes and double Pareto distributions. *Internet Mathematics*, 1(3):305–333, 2004.

[11] M. Mitzenmacher and B. Tworetzky. New models and methods for file size distributions. In *Proceedings of the Annual Allerton Conference on Communication Control and Computing*, pages 603–612, 2003.

[12] S. S. Pirzada. *A statistical examination of the evolution of the UNIX system*. PhD thesis, Imperial College. University of London., 1988.

[13] E. S. Raymond. The cathedral and the bazaar. *First Monday*, 3(3), March 1998. http://www.firstmonday.dk/issues/issue3_3/raymond/.

[14] 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.

[15] J. Wu. *Open Source Software evolution and its dynamics*. PhD thesis, University of Waterloo, 2006.

[16] J. Wu, R. Holt, and A. E. Hassan. Empirical evidence for SOC dynamics in software evolution. In *Proceedings of the International Conference on Software Maintenance*, pages 244–254. IEEE Computer Society, 2007.