Effort Estimation by Characterizing Developer Activity[*]

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

Gregorio Robles
grex@gsyc.escet.urjc.es

Jesus M. Gonzalez-Barahona
jgb@gsyc.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.

1. INTRODUCTION

Libre software[1] 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 significative 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-

[1]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).

ing for activity records, even when a company is promoting the project by means of hired developers.

2. IMPROVING "CLASSICAL" EFFORT ESTIMATION MODELS

In any software development project several sources of costs can be accounted, ranging from equipment to training and travel. In general, human resources are the origin of most of the costs. That is the reason why development effort estimation is central to cost estimation in software development.

Unfortunately, the amount of human effort needed for a certain project is at the same time the most difficult parameter to estimate. One of the reasons for this difficulty is at the root of software development: the specific system to be constructed is usually unique in its kind, and has never been built before with the same (or even similar) constraints. Among these constraints, some can be highlighted: the novelty of the domain, the use of new technologies, or the high variance of the productivity of software developers. Because of these reasons, effort estimation is still one of the open challenges in software engineering.

Classical effort models have been classified in several manners [5]. Of all of them, two are specially worth mentioning: those by Boehm and by Conte.

Boehm [2] cites seven kinds of estimation methods: algorithmic cost modeling, expert judgment, analogies, Parkinson's Law, pricing to win, top-down estimations, and bottom-up estimations. Conte [4] proposed a different classification, based on the nature of the models: historic or experimental, statistical, theoretical, and compound.

COCOMO and COCOMO II [3], two of the best known classical models, estimate effort using a statistical method, based on exponential formulas adjusted using regression techniques. This technique is an algorithmic model in Boehm's classification, and compound in Conte's work. COCOMO curves feature a good fitting of the historical database in which they are based.

These proposals are all based on a product metric: software size (usually in SLOC). However, in our opinion and at least for libre software projects, these models can be improved with the use of several additional data sources, which are usually available in software development repositories (that are public, and can be therefore easily analyzed by researchers). In particular, a promising line is the use of this information to perform the characterization of developer activity, which can in turn be used to better estimate effort in a project.

3. SOURCES OF INFORMATION ABOUT DEVELOPER ACTIVITY

In the next sections, some ideas about effort estimations in libre software will be commented. Since they will be related to the characterization of developer activity, we will start introducing the sources of information about activities performed by libre software developers. Three main sources of information will be considered:

1. Source code management systems. The source code in libre software projects is customarily stored in a revision control system, usually CVS[2]. With a revision control system, developers can easily manage source code, create development branches and later merge them to the stable one, etc. The repository of the source code management system permits the retrieval of detailed information about every transaction (commit) in it: date, author, change performed, comments, and other information. This information can be analyzed for tracing activity related to source code creation and modification.

2. Mailing lists archives. In libre software projects, since most developers are geographically distributed, the communication occurs mainly through electronic mail, in most cases in the context of electronic mailing lists. The corresponding archives can be analyzed, retrieving data such as date, subject, author and contents. This information can be used to reconstruct threads of discussion and communication flows between developers.

3. Bug tracking systems. These systems store all bug reports, requests for enhancements and new features requests. Libre software users can submit reports, providing feedback after performing testing activities (in most cases by using the system in real-world environments). Reports are assigned to developers, who fix the corresponding bugs, or implement the requested features, and then mark the report as *solved*. In bug tracking systems, users may add comments and may propose source code patches. All this information can also be analyzed.

Some other sources of information about developer activity may be considered. For example, activities related to the release of a new public version of the software, or the time spent writing in the project weblogs can be traced. However, these are more sporadic and specific activities, which we assume have less impact in the total effort accounting.

It is important to highlight that all these repositories are usually publicly available for libre software projects. The access to them does not require any special arrangement or agreement and can even be performed in a non-intrusive way. The researcher can just download that information from the project Internet site.

4. DEVELOPERS AND COST / EFFORT ESTIMATION

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

[2]Most libre software projects use a version control repository, being CVS the most popular. More than 10,000 projects hosted at SourceForge use it [11], and 7 out of the 10 largest projects in Debian 3.1 [1] also use a publicly available CVS. It should be noted, that in recent times many projects are switching to the next-generation Subversion system.

A good starting point for this line of research is the tracking of activity traces of developers in the aforementioned public sources. With these data, our position is that productivity parameters and software production efforts can be calculated and estimated.

Our position is based on the following assumptions: first, the human effort in a libre software project can be considered as the sum of individual efforts. But, from the point of view of those incurring in costs because of the development effort (usually the companies involved), not all efforts are equal. If the effort is external (e.g., not hired by those companies), while it is still important for the project, it means no cost for the companies. However, it is a cost, and should be considered as such in some circumstances (for instance, when evaluating the total resources put into the project). In the following formulae, "internal" cost are those costs and efforts considered subject to likely being assigned to a company participating in the project, while "external" refers to the effort by third parties, usually volunteers.

$$cost_{\ internal} = \sum cost_{\ internal}$$

$$cost_{\ external} = \sum cost_{\ external\_developers}$$

$$cost = cost_{\ internal} + cost_{\ external}$$

Each cost is function of required effort, that is:

$$cost = f(effort)$$

Also, developer effort is function of developer activity:

$$effort = g(activity) \Rightarrow cost = f(g(activity))$$

So, there are three different problems. The first being how to identify and then measure all sources of activity in libre software development. Secondly, how to estimate the corresponding cost, that is, the functions f and g. And the third, the need to validate those estimations.

In libre software projects we can identify and measure the most important developer activities. For example, if the project only consists of source code production and maintenance (which can be measured through revision control data, available in versioning systems like CVS), communication activity (for example, measuring the participation in mailing lists, ML) and bug tracking activity (time spent in managing and correcting bugs in a bug tracking system, BTS), a function like this can be used:

$$effort = g(activity) = a{\cdot}h(CVS) + b{\cdot}j(ML) + c{\cdot}k(BTS) + d$$

The problem of defining functions h, j and k, and the rest of the variables, may be addressed by comparing the traced activity with information about time spent by developers doing that work. For example, data can be obtained from surveys about actual time spent by developers in different activities, and compared with the traced activity for each developer [7].

However, some automatic activity tracing tools are being tested and taken into consideration, tools that must be installed and run by each selected developer.

Further validation can be resolved by comparing estimations obtained from several libre software projects with real time spent, also through surveys or tracing tools.

5. OUR RESEARCH LINES

Our research group at the Universidad Rey Juan Carlos has been analyzing libre software from an empirical point of view for several years now. Due to the large amount of available data and the huge amount of projects we are analyzing, we depend on automating most processes. Therefore, we have built some tools that allow us to obtain some preliminary results, and to come to some conclusions. They could be useful for working in the challenges presented in this paper.

Below, a list of tools that being currently used is offered, along with some of the obtained results so far:

- *CVSAnalY*[3] [19] is a CVS and Subversion repository analyzer that extracts information from a repository. CVSAnalY has a web interface[4] that allows browsing through results and figures. We have used this to investigate the evolution of *core* groups in time in libre software projects [10] as well as the (social and technical) structure of some libre software projects in time, as it is the case of Apache [9], GNOME and KDE [13].

  CVSAnalY is now being extended for our work in characterizing developer activity, such as identifying and classifying *transactions* [6] [15].

- *MailingListStats*[5] is a tool for analyzing activity in mailing lists. The tool itself extracts information about authors, dates, subjects and other fields related to email, and stores it in a database. We have used this to investigate steps in comprehending the software artifacts by libre software developers [12].

- *GlueTheos*[6] [17] is a tool that retrieves periodical snapshots from versioning systems, calls several external tools (including some mentioned in this list) to perform several quantitative analysis, stores the results into a database and proceeds with some higher level analysis.

- *DrJones* can be used to perform empirical analysis of software archaeology for software archived in a versioned repository. With it we have started to study how old is the current code base in some very well-known libre software applications [18].

- *CODD*[7] [16] (and its newer version, *pyTernity*) is an authorship extraction tool that looks for copyright statements, e-mail addresses or RCS login traces in source code files.

[3]http://cvsanaly.tigris.org/
[4]http://libresoft.urjc.es/cvsanal/
[5]http://libresoft.urjc.es/index_html?menu=Tools&Tools=MLStats
[6]http://libresoft.urjc.es/index.php?menu=Tools&Tools=GlueTheos
[7]http://codd.berlios.de

- *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.

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