Towards Predictor Models for large Libre Software Projects

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

ABSTRACT

Libre (free/open source) software provides an ample range of publicly available data sources about its development, which can be retrieved and analyzed. Consequently, it offers a good opportunity to build predictive estimation and evolution models. The main challenge to understand libre software development is that its development nature is radically different from 'classical' in-house software development, common in industry in the last decades. Developers and other human resources are generally a mixture of a few hired developers and many volunteers whose contribution (in number of hours per week and in total time devoted to the project) is not foreseeable in advance. This paper is a first step in finding predictive models in the libre software world. We have studied three data repositories (versioning system, mailing lists and bug tracking system) of GNOME, a large libre software project with several thousand contributors and several millions of lines of code, measuring activity and participation in it during the last years. Results and correlations for these sources allow us to adventure some first estimations of how participation and activity will evolve in the future.

Categories and Subject Descriptors

K.6.3 [**Software Management**]: [software development, software maintenance]

General Terms

Management, Measurement

Keywords

libre software, predictor models, open source, data mining, software evolution

PROMISE'05, May 15, 2005, St. Louis, Missouri, USA.

1. INTRODUCTION

It has already been stated [8], that the development process in successful libre software[1] projects is radically different from the 'classical' in-house development strategies that had been used for decades. Besides using extensively distributed development environments, self-selected volunteers perform crucial tasks such as support, bug reporting, minor software enhancements and even form a part of the developing core group, which is responsible for a high amount of the total source code produced.

But although the importance of libre software has been raising during the last years (for instance, much of the Internet infrastructure relies on libre code), there is a lack of predictor models for these type of developments. This paper is a first step in finding predictive models in the libre software world. We have studied three data sources (versioning system, mailing lists and bug tracking system) from a large libre software project and measured activity and participation during the last years. Results and correlations for these sources allow us to make first estimations of how participation and activity will be in future.

The structure of this paper is as follows: the next section will illustrate related work on prediction models and other evolutionary studies on libre software. Then, we will present the research goals of this study, following with the -empirically based- methodology that we have used. Results and observations have been placed in the next section, including a discussion about some issues that may be learned from that results. Finally, some conclusions and further lines of work can be found.

2. RELATED WORK

Libre software projects offer governance structures and participation procedures that differ drastically from what has been observed in industry during the last decades. German [2] provides a detailed description of how is organized GNOME, the large libre software project that we will also target in this study. The goal of that study was to identify how the most important problems of global software development (distance, time differences and cultural differences) are solved in the libre software world. Besides these characteristics, libre software development has a strong volunteer

[1]Through 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).

nature which makes it difficult to manage and also to predict [7].

Since the source code and many of the subproducts of the development process, such as mailing list archives, bug report systems or versioning repositories are publicly available [11], there have been many empirically based studies on libre software ([8], [4]).

Other studies that may help advancing the research agenda towards predictor models in the libre software world are related to software evolution, specially if they can be linked to the ones that study the human resources as the ones presented in the previous paragraph. In this sense, Godfrey et al. [3] found that the Linux kernel had a superlinear growth, apparently breaking one of Lehman's laws on software evolution [6].

But the most important work regarding predictor models in the libre software world is the one authored by Koch et al. [5]. Classical effort estimation models that derive from Norden [9] and Putnam [10] are used to see what the development of a large libre software system may cost. The main assumption was that once a peak point is given in the software development, it is possible to apply a Norden-Rayleigh-type curve and predict the total effort (and hence the cost) of the project. Due to the public availability of some development data (for the case of this study mailing list archives and commits to the versioning repository were used) the estimation could be validated with some hints of success, but also some weak points.

3. RESEARCH GOALS

Our goal with this work is to analyze participation patterns in a large libre software project which may help in the development of a model that predicts the evolution of the contributions in such kind of projects. Estimating the contributions is a first step towards the estimation of the production of source code, as well as the amount of time that should be scheduled for this purpose. Once the effort and the time required is known, cost could be evaluated.

Our basic hypothesis is that the evolution of a project cannot be characterized only by means of the size of the source code of the project. The environment plays a very important role. The *technical sphere* is as important as *the community sphere*. Therefore, a model which pretends to predict evolution, must take into account parameters from both spheres.

In particular, we have used several parameters related to three of the most important aspects of software development: source code production (and modification), communication, and maintenance. For source code production we have considered commits in the CVS repository, which is a good indicator of the coding activity. For communication, we have considered messages to mailing lists since those lists are the most important channel for information flow within the project. For maintenance, we have considered bug reports since again they are a good indicator of the fixes needed, and of how they are being dealt with.

Based on these parameters, we have looked for correlations and interpolations that can highlight patterns and relationships among them.

4. METHODOLOGY

The data sources that have been used for this study are publicly available on the Internet and can be retrieved in an almost automatic manner. Specifically, the ones considered for this study were the mailing lists archives[2], the bug tracking system[3] and the CVS versioning system[4] of the GNOME project, a several million lines of code desktop environment developed by some thousands of developers world-wide.

The data gathered from the mailing list archives has been parsed and dumped into a database. We have downloaded the archives for 109 mailing lists included in the GNOME and GIMP archives, and computed in total $464,953$ messages. The first message was sent on May 30th 1996 and the last one on November 16th 2004.

For the analysis of the GNOME bug tracking system we have retrieved all the bugs which are publicly available trough the Bugzilla web interface, and have obtained $123,739$ bug reports submitted by $41,835$ reporters. The first bug dates from February 1999, while the last considered for this study was reported on November 2004.

For CVS we have used the CVSAnalY tool [12], which makes a detailed analysis of the logs of the CVS repositories. In the case of GNOME we have identified $1,067$ different developers with write access to the CVS repository working on the 767 existing modules. The CVS repository was open in November 1997, and the last commits considered in this study are from April 2004.

For each source we have performed a detailed study to characterize its evolution. We will therefore handle two different sets of parameters in the following section, one linked to the volume (number) of artifacts that have been generated for each source and another one which includes the (number of) persons that have produced them:

- Regarding volume and size we have chosen the number of messages sent to mailing lists, the number of reported bugs, and the number of commits to the versioning system (CVS), computed for each month during the lifetime of the project.

- Regarding the persons involved, we have chosen the number of people who have sent at least one message for a given month (in the following, *posters*), the number of people who have reported at least one bug for a given month (in the following, *reporters*) and number of people who have committed at least one change to the versioning system for a given month (in the following, *commiters*).

5. RESULTS AND OBSERVATIONS

We have correlated the number of messages, the number of bugs and the number of commits; one of the correlations is showed in the figure 1, and the rest in the appendix (figures 6 and 7). We have also correlated the number of posters, the number of reporters, and the number of commiters. One of these correlations is showed in the figure 2, and the rest in the appendix (figures 8 and 9). With all these correlations, we can estimated any of the parameters if we know any other of them.

Furthermore, we have found that the series in time of these parameters are power laws. For example, in figure 4 and figure 5 we show the power laws for the number of

[2]http://mail.gnome.org/archives/

[3]http://bugzilla.gnome.org/

[4]http://libresoft.dat.escet.urjc.es/cvsanal/gnome2-cvs/

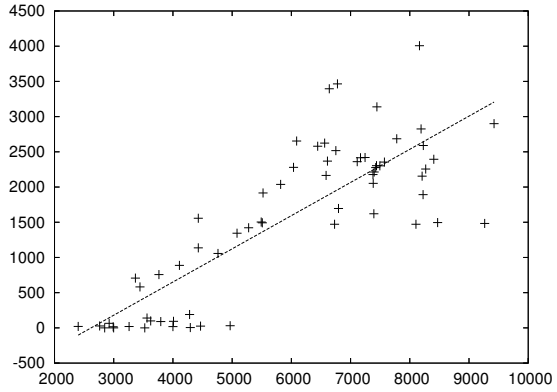

Figure 1: Correlation between the number of messages (x axis) and the number of bugs (y axis)

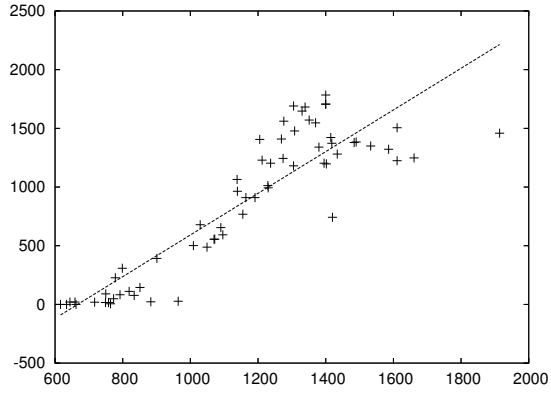

Figure 2: Correlation between the number of posters (x axis) and the number of reporters (y axis)

messages versus time, and for the number of poster versus time. In the appendix we include the rest of laws of evolution for each parameter (figures 12, 13, 14 and 15).

The calculated Pearson coefficients show correlation between all the pair of parameters. The linear correlation among the number of messages and the number of bugs throws a Pearson coefficient of 0.951. The correlation among the number of bugs and the number of commits has a coefficient of 0.818. And finally, the correlation among the number of messages and the number of commits has a Pearson coefficient of 0.971. These coefficients evidence strong correlations among the number of messages, the number of bugs and the number of commits.

We have obtained the same correlations for people who participate in the mailing lists, Bugzilla and CVS. There are also evidences of linear correlations, as the Pearson coefficients show. The correlation among the number of posters and the number of reporters has a Pearson coefficient of 0.967. The correlation between the number of reporters and the number of commiters has a coefficient 0.818. Finally, the coefficient for the correlation among commiters and posters has a value of 0.988.

As we can see, the relation among number of bugs and number of commits is the weakest, as the relation between number of reporters and number of commiters. In the other side, relation among bugs or commits and messages, and among reporters or commiters and posters, are the best, which can indicate that the activity of reporting bugs or making a commit generates more activity in the mailing lists.

Other more interesting correlations are those which correlate the number of people with the activity due to this people. We have correlated number of messages and number of people who send messages, number of bugs and number of reporters, and number of commits and number of developers who can write into the CVS. As our data evidence, these correlations are similar to Brooks' law [1], because they are not linear, but quadratic. All the relations have been fitted by a function like the following: $f(x) = ax^2 + bx + c$, except for the relation among the number of reporters and the number of bugs, which has resulted to be linear.

In the case of the figure 3, which shows the relation between the number of posters and the number of messages, we have founded a quadratic relation, with a coefficient matrix

	a	b	c
a	1.000		
b	−0.989	1.000	
c	0.950	−0.984	1.000

The relation showed in the figure 10 (included in the appendix) resulted to be linear, with a Pearson coefficient of 0.818. At this moment, we can't explain the reason behind this pattern. If the number of bugs were almost equal to the number of reporters, we can try to say that the bugs are more distributed and diffused than commits or messages, and so reporters use to have a little number (one, two, etc) of bugs. But in this case, the number of total bugs is about two times larger than the number of reporters. It is a really strange case, because it looks that, each month, people usually report two bugs, and nobody reports a large number of bugs. That is, there is not individuals who concentrate the greater part of all bugs.

In the case of the figure 11 (included in the appendix), which shows the relation between the number of commits and the number of commiters, we have again a quadratic relation, with a coefficient matrix

	a	b	c
a	1.000		
b	−0.997	1.000	
c	0.987	−0.996	1.000

The figures 3 and 11 verifies the Brooks' law, because a linear increase in the number of people does not lead to a linear increment of the activity of the project (understanding activity as number of messages or number of commits). When the number of people who participate becomes larger, the project begins to be saturated, and the growing rate decreases.

6. CONCLUSIONS AND FURTHER WORK

This paper presents a first step in the development of a framework to obtain prediction models of the evolution of large libre software projects. We have shown in a specific case study how several metrics of participation can be studied, and how they are related to each other.

This specific study on GNOME shows strong evidence on

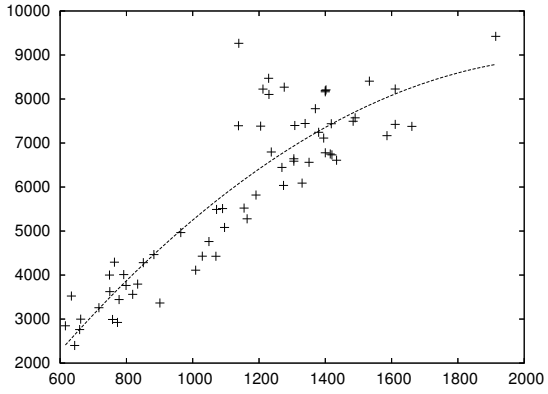

Figure 3: Correlation between the number of posters (x axis) and the number of messages (y axis)

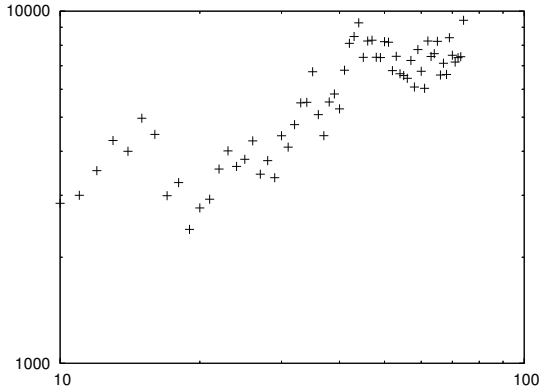

Figure 4: Number of messages versus time (as month index from the beginning of the project), in log scale.

hight correlations between the number of bugs, messages and commits, measured each month during the lifetime of the project. We have also found similar relationships between the number of posters, reporters and commiters. Furthermore, there are some hints of correlation between the number of posters and the number of messages, the number of reporters and the number of bugs, and the number of commiters and the number of commits. These correlations are quadratic, so they conform with Brooks' law that states that communication grows in a quadratic fashion.

Moreover, we have found that the evolution of these parameters in time follows a power law. Therefore, with the found correlations, and the laws of evolution of each parameter, we could predict any of the parameters if we knew the evolution of any other in the past.

We consider that these evidences are enough to continue exploring more case studies, in order to find a predictive model for large libre software projects. Our ultimate goal is to describe the evolution of the size of the project, and the cost associated to the development, using publicly available data.

The next step in our research agenda is to improve the model by studying many other libre software projects, and to extend it to consider other parameters, such as the number

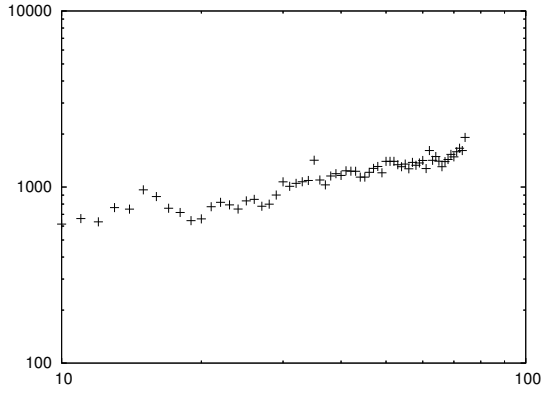

Figure 5: Number of posters versus time (as month index from the beginning of the project), in log scale.

of lines of code.

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. We also thank the PROMISE reviewers for their comments.

8. REFERENCES

[1] F. P. Brooks, Jr. *The Mythical Man-Month*. Addison Wesley, anniversary edition, 1995.

[2] D. Germn. The GNOME project: a case study of open source, global software development. *Journal of Software Process: Improvement and Practice*, pages 201–215, Aug. 2004.

[3] 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.

[4] 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.

[5] S. Koch and G. Schneider. Effort, cooperation and coordination in an open source software project: GNOME. *Information Systems Journal*, 12(1):27–42, 2002.

[6] 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.

[7] 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.

[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] P. Norden. Useful tools for project management. *Operations Research in Research and Development*, 1963.

[10] L. Putnam. A general empirical solution to the macro software sizing and estimating problem. *IEEE Transactions on Software Engineering, 4: 345-361*, 1978.

[11] G. Robles, J. M. Gonzalez-Barahona, J. Centeno-Gonzalez, V. Matellan-Olivera, and L. Rodero-Merino. Studying the evolution of libre software projects using publicly available data. In *Proceedings of the 3rd Workshop on Open Source Software Engineering*, pages 111–115, Portland, Oregon, 2003.

[12] 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. ADDITIONAL FIGURES

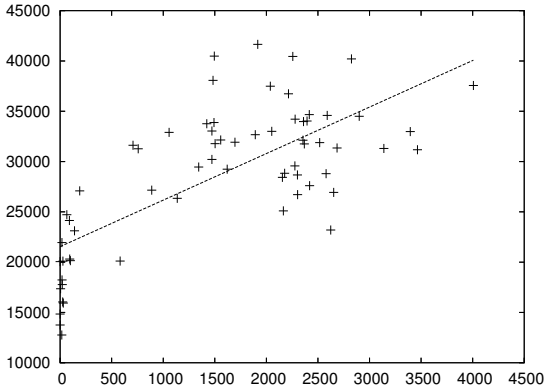

Figure 6: Correlation between the number of bugs (x axis) and the number of commits (y axis)

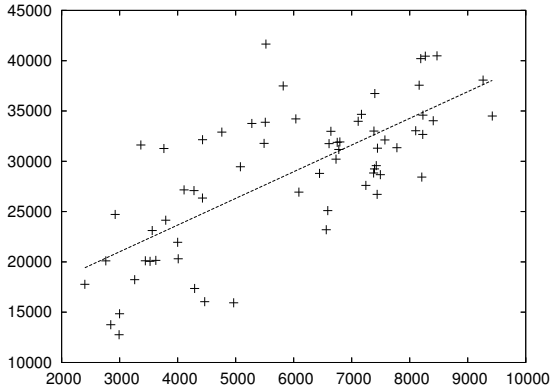

Figure 7: Correlation between the number of messages (x axis) and the number of commits(y axis)

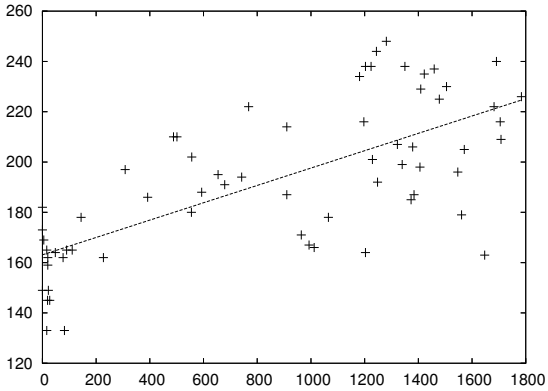

Figure 8: Correlation between the number of reporters (x axis) and the number of commiters (y axis)

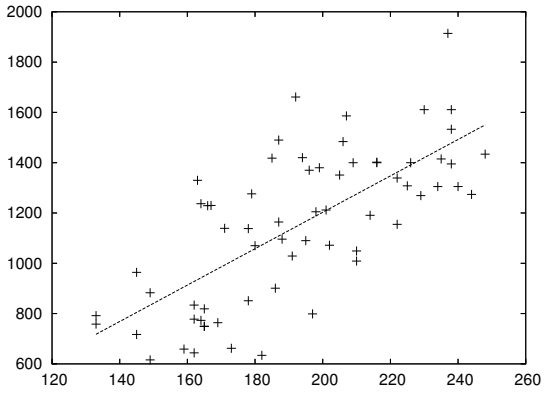

Figure 9: Correlation between the number of commiters (x axis) and the number of posters (y axis)

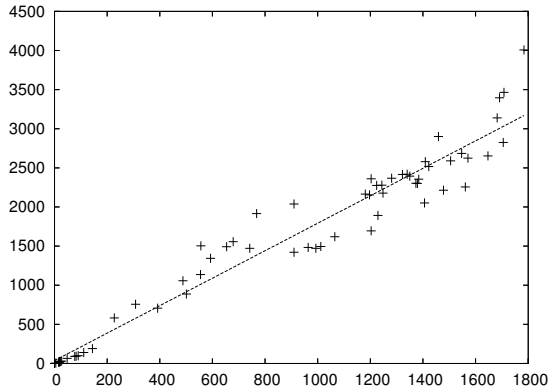

Figure 10: Correlation between the number of reporters (x axis) and the number of bugs (y axis)

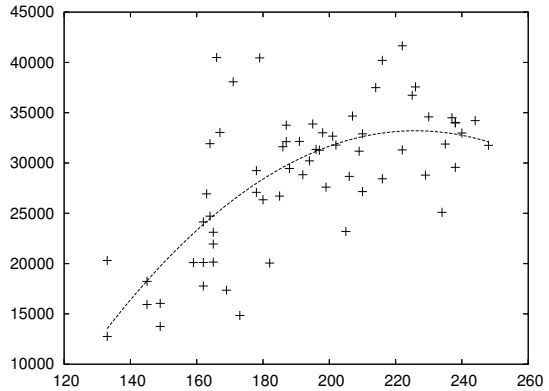

Figure 11: Correlation between the number of commiters (x axis) and the number of commits (y axis)

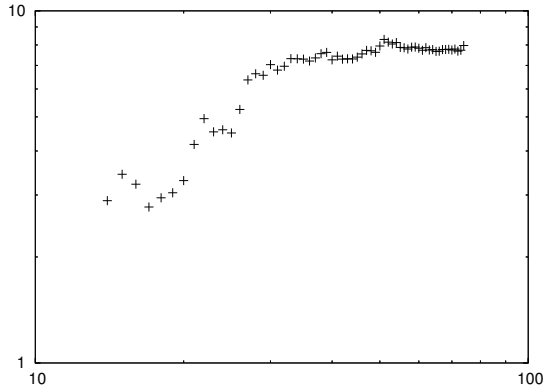

Figure 12: Number of bugs versus time (as month index from the beginning of the project), in log scale.

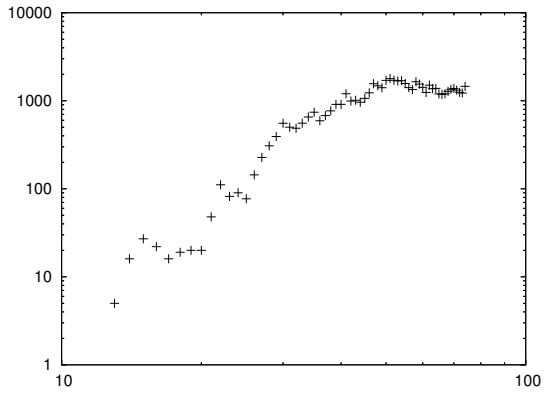

Figure 13: Number of reporters versus time (as month index from the beginning of the project), in log scale.

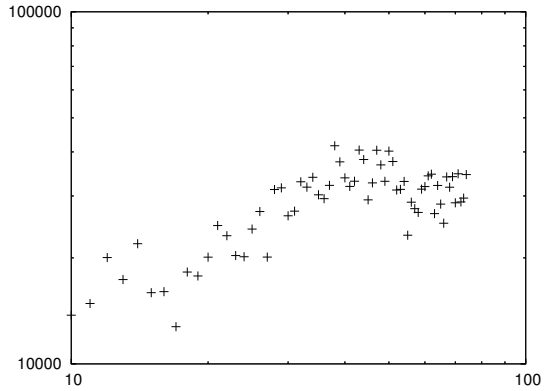

Figure 14: Number of commits versus time (as month index from the beginning of the project), in log scale.

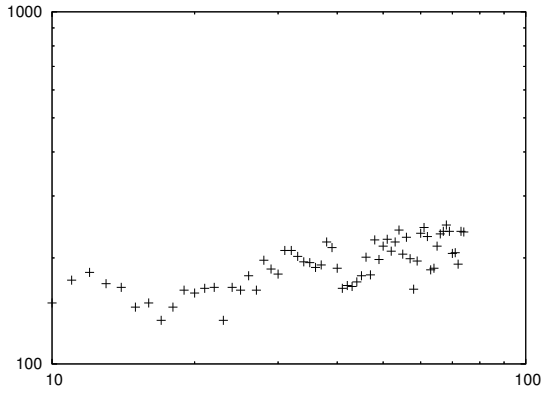

Figure 15: Number of commiters versus time (as month index from the beginning of the project), in log scale.