Relationship between Size, Effort, Duration and Number of Contributors in Large FLOSS projects

Juan Fernandez-Ramil[1,3], Daniel Izquierdo-Cortazar[2] and Tom Mens[3]

[1] The Open University, Milton Keynes, U.K. j.f.ramil@open.ac.uk
[2] Universidad Rey Juan Carlos, Madrid, Spain dizquierdo@gsyc.urjc.es
[3] Université de Mons-Hainaut, Mons, Belgium {j.f.ramil,tom.mens}@umh.ac.be

1 Introduction

This contribution presents initial results in the study of the relationship between size, effort, duration and number of contributors in eleven evolving Free/Libre Open Source Software (FLOSS) projects, in the range from approx. 650,000 to 5,300,000 lines of code. Our initial motivation was to estimate how much effort is involved in achieving a large FLOSS system. Software cost estimation for proprietary projects has been an active area of study for many years (e.g. [1][2][3]). However, to our knowledge, no previous similar research has been conducted in FLOSS effort estimation. This research can help planning the evolution of future FLOSS projects and in comparing them with proprietary systems. Companies that are actively developing FLOSS may benefit from such estimates [4]. Such estimates may also help to identify the productivity 'baseline' for evaluating improvements in process, methods and tools for FLOSS evolution. Table 1 shows the projects that we have considered, together with the programming language(s) primarily used for each system and the time from the first known commit.

Table 1. FLOSS systems studied and some of their characteristics

name	primary language	description	time from 1st commit (years)
Blender	C/C++	cross-platform tool suite for 3D animation	6
Eclipse	Java	IDE and application framework	6.9
FPC	Pascal	Pascal compiler	3.4
GCC	C/Java/Ada	GNU Compiler Collection	19.9
GCL	C/Lisp/ASM	GNU Common Lisp	8.8
GDB	C/C++	GNU Debugger	9.5
GIMP	C	GNU Image Manipulation Program	10.9
GNUBinUtils	C/C++	collection of binary tools	9.4
NCBITools	C/C++	libraries for biology applications	15.4
WireShark	C	network traffic analyser	10
XEmacs	Lisp/C	text editor and application development system	12.2

The measurements extracted for this study are listed in Table 2. We used the SLOC-Count[4] tool to measure the size of the software in lines of code. In order to measure size in number of files, duration, effort and team size, we used CVSAnalY[5]. This tool stores information extracted from the version control log (CVS, Subversion or Git) in a MySQL database. Specifically, we indirectly used CVSAnalY by downloading data from FLOSSMetrics[6]. For mesuring $EFFORT$, possibly in a very approximate way, we added one contributor-month for each contributor making one commit or more in a given month. Then, we calculated the number of contributor-years by dividing by 12 the total contributor-month values for each project.

Table 2. Measured attributes for each system

attribute name	description	extracted using:
$KLOC$	physical lines of source code (in thousands)	SLOCCount
$netKLOC$	see explantion in text	
$FILES$	total number of files in code repository	CVSAnalY
$netFILES$	see explanation in text	
$EFFORT$	effort in contributor-years	CVSAnalY
DUR	time length of 'active' evolution in years	CVSAnalY
DEV	number of unique contributors	CVSAnalY

2 Data filtering

In most of the projects, when looking at the total number of files ($FILES$) in the repository per month, we observed a number of relatively large 'jumps'. Figure 1 (left hand side) shows this for project GCL. It is unlikely that the productivity of the team of contributors has increased so suddenly. It is more likely that each 'jump' reflects moments in which the repository receives chunks of externally generated files. It could also be that non-code files were added. We tried to filter out this phenomenon by subtracting the size increments in the 'jumps'. We also subtracted the initial size. We call the result $netFILES$. For GCL, the result is presented on the right hand side of Figure 1. Since $KLOC$ was measured only at one recent date, we also defined $netKLOC$ as $KLOC * \frac{netFILES}{FILES}$.

When measuring project life span or duration DUR since the first commit up to Oct 2008 – for Eclipse the cut off month was April 2008 –, we excluded any apparent periods of no commits or with monthly commits number much lower than during other periods. We did this for two of the projects, GCC and GCL. This can be seen on Figure 1 for GCL. The left-hand-side plot shows the total duration. The right-hand-side shows the period of active work, which determined the value of DUR (4.3 years instead of 8.8). Similarly, for GCC we assumed a value of DUR of 11.2 instead of 19.9 years.

[4] `www.dwheeler.com/sloccount/`

[5] `svn.forge.morfeo-project.org/svn/libresoft-tools/cvsanaly`

[6] `data.flossmetrics.org`

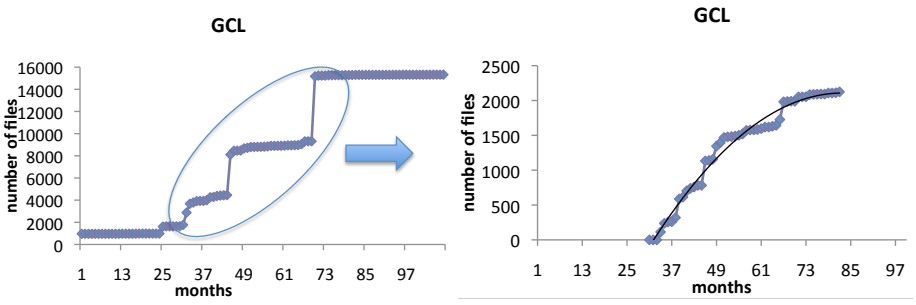

Fig. 1. Trend in $FILES$ (left) and $netFILES$ (right) per month for GCL.

For DEV we counted the number of people making at least one commit since the start of the repository. We computed two variants, one measuring all the contributors, $DEV[100]$, and one measuring the number of developers which provided 80 percent or more of the effort in contributor-months, or $DEV[80]$, in order to exclude contributors who may have made only a few commits.

3 Preliminary results

Initially, we studied the $netFILES$ trends over months. We found a very good fit to linear, quadratic or exponential models, with coefficient of determination[7] (R^2) values ranging between 0.98 to 0.99. Five of the systems had trends that follow linear models, two follow sublinear (quadratic) and three superlinear (2 quadratic and one exponential). The high goodness of fit suggests that size increase in these FLOSS systems take place at a predictable rate. This mixture of different trends is in agreement with previous empirical studies of FLOSS growth trends [5].

As it is common practice in effort estimation models (e.g., COCOMO [2]), we studied the linear correlation between size (measured in $KLOC$ and $FILES$) and $EFFORT$, DUR and DEV. Table 3 shows the parameters of linear models of the form $y = (a * size) + b$ and the corresponding a, b and R^2 values, for the eleven systems and for ten systems (excluding Eclipse) The exclusion of the data of the largest system, Eclipse, following its identification as a possible outlier in some of the scatter plots that we studied, is indicated by an '*' in the attribute's name (e.g. $KLOC^*$). Parameters a and b are not reported when R^2 is less than 0.1 because they are not meaningful. The best models, indicated in bold, are obtained when Eclipse is excluded from the dataset.

Contrary to what was expected, the sole removal of unusual 'jumps' in the monthly growth of the studied systems did not lead to visible improvements in the regression models. It is the exclusion of Eclipse that leads to improvement in the correlation in 13 out of 16 correlations studied. All the best regression results correspond to net

[7] This is a measure of goodness of fit, with values from 0, for a very poor fit, and 1 for a very good fit.

Table 3. Linear regression results - parameters a, b and R^2 values

size in:	$EFFORT$ (a, b, R^2)	DUR (a, b, R^2)	$DEV[100]$ (a, b, R^2)	$DEV[80]$ (a, b, R^2)
$KLOC$	(0.0858, 29.909, 0.339)	(-, - , 0.012)	(0.027, 88.538, 0.101)	(0.0095, 29.57, 0.1331)
$FILES$	(0.0039, 119.58, 0.390)	(-, -, 0.001)	(0.0017, 103.97, 0.219)	(0.0006, 35.142, 0.2845)
$netKLOC$	(0.076, 110.39, 0.326)	(- , - , 0.048)	(0.0287, 106.56, 0.139)	(0.0105, 35.384, 0.196)
$netFILES$	(0.0035, 156.61, 0.313)	(- , - , 8.8E-05)	(0.0016, 119.2, 0.185)	(0.006, 40.066, 0.258)
$KLOC^*$	(0.1327, -53.323, 0.387)	(0.0015, 6.1233, 0.153)	(- , - , 0.088)	(- , - , 0.07)
$FILES^*$	(0.0093, 3123,0.66)	(- , - , 0.06)	(0.004, 62.831, 0.391)	(0.0012, 24.787, 0.367)
$netKLOC^*$	(0.1699, 14.247, 0.499)	(0.0032, 5.4474, **0.525)**	(0.0626, 71.879, 0.196)	(0.0183, 27.401, 0.185)
$netFILES^*$	(0.0139, 51.455, **0.797)**	(- , - , 0.094)	(0.0143, -19.009, **0.565)**	(0.002, 25.998, **0.506)**

size and with Eclipse excluded. The best regression model obtained is the one involving $EFFORT$ as a linear function of $netFILES'$ (R^2 value of 0.797), that is, $EFFORT = 0.0139 * netFILES^* + 51.455$. According to this model, reaching say, 8,000 files would require about $(0.0139 * 8000) + 51.455$ or 163 contributor-years [6]. The six worst regression models correspond to DUR vs size, indicating that FLOSS increase in size at different rates across projects.

With regards to external validity (generalisation), eleven arbitrarily chosen FLOSS constitute a small sample to be able to generalise our conclusions. Regarding threats to internal validity, $EFFORT$ is likely to be an over-estimation of the actual effort. This is so because many FLOSS contributors are volunteers who work part-time rather than full time on the project. One way to improve measuring $EFFORT$ would be to conduct a survey of FLOSS contributors to know better their work patterns and use this knowledge to adjust our measurements. Surveys, however, may require considerable research effort and the willingness of FLOSS contributors to participate. Our $EFFORT$ measurement may be more accurate for FLOSS projects like Eclipse, where a portion of contributors are full-time employees of a sponsoring company (in this case, IBM). Despite all our care, the tools used to extract and analyse the data may contain defects that may affect the results.

4 Conclusion and further work

This paper reports on work in progress in the study of the relationship between size, effort and other attributes for large FLOSS, an issue that does not seem to have been empirically studied. The preliminary results are encouraging and better models may be obtained through refinement and improvement of the measurement methodology. If this were succesful, the models could provide a baseline to study, for example, the possible impact of new or improved processes, methods and tools. The models may also help FLOSS communities in systematic planning of the future evolution of their systems. The best model excludes data from Eclipse. This suggests that independent models for very large systems (greater than 5 million LOC) or for systems based on different technologies (Eclipse was the only Java based system in the sample) are worth exploring.

We have started to compare how well existing cost estimation models based on proprietary projects (in particular, COCOMO [2]) correspond to our FLOSS data [6]. In

order to increase the external validity of the results, we would need to study an additional number of FLOSS projects. The regression models we used are simple. Better results may be achievable by using *robust regression* [7]. Other suitable modelling techniques different than regression [8] could be evaluated against the data. Accuracy may be improved by including other attributes that may impact productivity and duration. One of these is the so-called number of *orphaned lines of code* [9], the portion of the code left behind by contributors who have left a project. Currently, our $FILES$ measurement considers all files in the repository (i.e. code, configuration, data, web pages). It is likely that better results will be achieved by considering code files only. We would also like to exclude any automatically generated files since they will bias the results (e.g., programming productivity may appear higher than it is). We also plan to examine different approaches to extract the outliers from monthly growth trends. In Figure 1 (right) one can identify 'jumps' that were not apparent when looking at the unfiltered data (left). One question is how to define formally what is a 'jump'. In particular, one needs to check manually the version repositories to confirm whether the 'jumps' are actually corresponding to the inclusion of externally generated code or libraries.

Acknowledgements

This contribution has been prepared during the research visits of two of the authors (DI and JFR) to UMH. One of the authors (JFR) acknowledges the Belgian F.R.S.-F.N.R.S. for funding through postdoctoral scholarship 2.4519.05. One of the authors (DI) has been funded in part by the European Commission, under the FLOSSMETRICS (FP6-IST-5-033547) and QUALOSS (FP6-IST-5-033547) projects, and by the Spanish CICyT, project SobreSalto (TIN2007-66172). TM acknowledges partial funding by the *Actions de Recherche Concertées – Ministère de la Communauté française - Direction générale de l'Enseignement non obligatoire et de la Recherche scientifique*.

References

1. Wolverton, R.W.: The cost of developing large-scale software. IEEE Trans. Computers **C-23**(6) (June 1974) 615 – 636
2. Boehm, B.: Software Engineering Economics. Prentice Hall (1981)
3. Jones, C.: Estimating Software Costs - Bringing Realism to Estimating. McGraw Hill (April 2007)
4. Amor, J.J., Robles, G., Gonzalez-Barahona, J.M.: Effort estimation by characterizing developer activity. In: EDSER '06: Proceedings of the 2006 international workshop on economics driven software engineering research, New York, NY, USA, ACM (2006) 3–6
5. Herraiz, I., Robles, G., Gonzalez-Barahona, J.M., Capiluppi, A., Ramil, J.F.: Comparison between SLOCs and number of files as size metrics for software evolution analysis. In: Proc. European Conf. Software Maintenance and Reengineering (CSMR), Bari, Italy (March 2006)
6. Fernandez-Ramil, J., Izquierdo-Cortazar, D., Mens, T.: How much does it take to achieve one megaloc in open source? Submitted for publication (November 2008)
7. Lawrence, K.D., Arthur, J.L.: Robust Regression: Analysis and Applications. CRC Press (1990)
8. MacDonell, S.G., Gray, A.R.: Alternatives to regression models for estimating software projects. In: Proceedings of the IFPUG Fall Conference, Dallas TX, IFPUG. (1996)
9. Izquierdo-Cortazar, D., Robles, G., Ortega, F., Gonzalez-Barahona, J.: Using software archaelogy to measure knowledge loss in software projects due to developer turnover. In: Proceedings of the Hawaii International Conference on System Sciences (HICSS-42), Hawaii, USA, forthcoming (January 2009)