IDEA GROUP PUBLISHING

701 E. Chocolate Avenue, Suite 200, Hershey PA 17033, USA
Tel: 717/533-8845; Fax 717/533-8661; URL-http://www.idea-group.com

ITP4597

Effort Estimation in Open Source Software Development: A Case Study

Stefan Koch
Department of Information Business, Vienna University of Economics and BA, Augasse 2-6, A-1090 Vienna, AUSTRIA
Telephone: ++43 1 31 336 5206, Fax: ++43 1 31 336 739, stefan.koch@wu-wien.ac.at

ABSTRACT

This paper presents first attempts at estimating the effort in open source software projects. First, the possible parties interested in the results of effort estimation, and both hindrances and advantages for effort estimation in this context are explored. Using data concerning an open source project retrieved from public data, several well-known estimation models from literature are applied, and their applicability for this type of development is discussed.

1. INTRODUCTION

Open source software development (Feller and Fitzgerald, 2002; Raymond, 1999) has generated increasing interest in the last years. This software is characterized by several rights given by the respective licence, including free redistribution, inclusion of the source code, possibility for modifications and derived works, and some others (Perens, 1999). The guiding principle for open source software development is that by sharing source code, developers cooperate under a model of rigorous peer-review and take advantage of "parallel debugging" that leads to innovation and rapid advancement in developing and evolving software products.

While the differences between the decentralized open source process and traditional software engineering practices have been debated (McConnell, 1999; Vixie, 1999), and also quantitative studies of development projects and communities have been undertaken (Dempsey et al., 2002; Ghosh and Prakash, 2000; Hermann et al., 2000; Koch and Schneider, 2002; Krishnamurthy, 2002; Mockus et al., 2002), some points remain to be explored. One of the most important questions remaining is the effort for developing open source software, which is not known even to the leaders of the respective project. As software engineering has dealt with the problem of estimating the effort for a software project for decades and has produced a multitude of methods to this end, their use seems a natural answer. Whether these models can indeed be used for open source software projects needs to be ascertained. If this were the case, the information delivered would have additional benefits even for the open source community and companies pursuing related business models.

2. CASE STUDY

For this research, data concerning a large scale open source project were needed. Therefore we chose to use data available to the public from the version control system CVS (Concurrent Versions System; Fogel, 1999) and discussion lists of the GNOME project. In particular, data concerning the participants' contributions to the project, their cooperation and the progression of the project in size and participants over time could be retrieved from these sources. For results concerning areas other than effort estimation see Koch and Schneider (2002).

The data retrieved from the CVS-repository included for every checkin programmer, file, date, LOC added and deleted, revision number and some comment. Using the conventions of CVS, a programmer is doing work on the project by submitting ("checking in") files, which is recorded with the changes in the lines-of-code and further information. The definition of this often disputed metric LOC is taken from CVS and therefore includes all types, e.g. also commentaries (Fogel, 2000). As the difference between the date of the first and the last checkin of a programmer includes all time elapsed, not necessarily only time spent working on the project, this measure is not usable for predicting output. Therefore, a programmer is defined as being active in a given period of time if he performed at least one checkin during this interval.

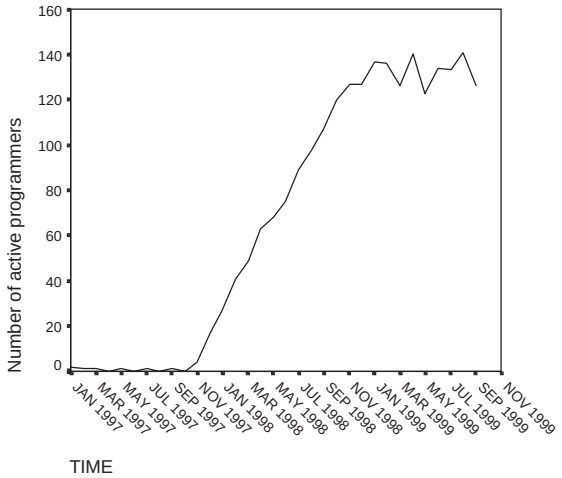
In the GNOME project, 301 programmers were identified, who differ significantly in their effort for this software project, with a majority contributing only a quite small amount to the total work done, a result also found by Dempsey et al. (2002) and Mockus et al. (2002). The total size of the GNOME project in LOC has experienced a steady increase up to the size of 1 800 000 LOC at the end of the observed time period, with 1 230 000 LOC being the size at the time it became operational (first major release in March 1999). During this time, the number of active programmers has seen a staggering rise between November 1997 and the end of 1998 (see Fig. 1). During the year 1999 this number has been roughly constant at around 130 persons. One reason for this development could be taken from Norden (1960) and Putnam (1978) who argue that only a given amount of persons can be working in a productive manner at a given time. In the light of this interpretation, the peak manning of the project has already been reached and will only see a downfall from thereafter. A correlation of 0.932 was found between total of LOC added and number of active programmers each month, which confirms the usability of this number for effort estimation. Another interesting finding is that productivity (defined as the mean number of LOC per programmer) is strongly positive correlated with number of active programmers in each month, thus violating Brooks' Law (Brooks, 1995).

3. EFFORT ESTIMATION

3.1. General Discussion

Establishing effort estimation for open source projects has two reasons, with the first being to uncover how efficient this form of development is. In the last years, this topic has been the center of much debate, major points always having been the efficiency of finding and correcting bugs relatively late in the life cycle (McDonnell, 1999; Vixie, 1999), and the high overhead

Figure 1. Number of active programmers



Copyright © 2003, Idea Group Inc. Copying or distributing in print or electronic forms without written permission of Idea Group Inc. is prohibited.

costs incurred for coordination and duplicated work. As the effort expended for developing open source software is unknown, these questions could not be answered. Therefore establishing retroactive effort estimation, thus arriving at a quantification for the effort expended for an existing software system, could help in deciding whether this development model should be pursued, abandoned or combined with traditional approaches into hybrid-models.

On the other hand, while the results of an effort estimation in commercial development are used for planning and control by management, there are also stakeholders in open source projects who could be interested in such results at early stages or during a project. These include the community itself, especially, dependent on the organizational form, the owner/maintainer, inner circle or committee (Fielding, 1999; Raymond, 1999), which need to monitor progress and plan for release dates, and programmers considering whether to join or to remain in a project. Further possible interested parties are current or prospective users, who need the functionality at a given date or with a given maturity level, especially corporations which are intending to pursue a business models based on this software, need it for their operations or plan to incorporate it in their products or services.

Several problems are associated with estimation for open source projects, in addition to the problems inherent in effort estimation. The first problem is the voluntariness of people's participation, which might also result in a high turnover of personnel and reduced productivity (Brooks, 1995). On the other hand, empirical data shows both that productivity is not necessarily declining (see above) and that the staffing for open source projects follows the postulated model for commercial software development (Norden, 1960; Putnam, 1978) closely (see below and Koch and Schneider, 2002). Vixie (1999) mentions the lack of a formal design and requirements definition as a problem, as necessary information for estimation will be missing. This information depends on the model employed and is thus discussed for each approach.

In addition, several assumptions of effort estimation models are inherently violated in open source development. For example COCOMO (Boehm, 1981) assumes a good management by both software producer and client, development following a waterfall-model and permanence of the requirements during the whole process. As there is no distinction between producer and client in open source development this seems no problem. The other two assumptions are both indeed violated, as the requirements are neither written down (Vixie, 1999) nor constant over time, and the software development follows are more spiral type of approach (Boehm, 1988), having been termed micro-spirals (Bollinger et al., 1999). COCOMO II (Boehm et al., 2000) on the other hand does not contain these assumptions but incorporates a more prototype-oriented type of development. The function point method also does not contain any assumption concerning the process model as it aims at being technology-independent and taking the user's viewpoint (Albrecht and Gaffney, 1983).

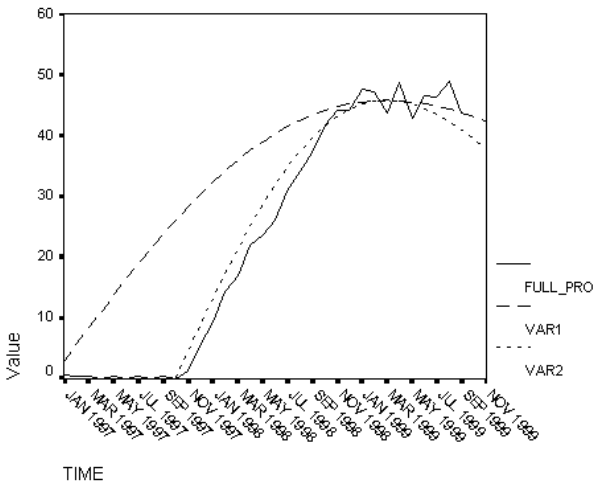
One main advantage is that all information concerning an open source project is available to the public. Therefore the data can be used by any interested party for estimation, which is not possible in commercial development.

3.2. Norden-Rayleigh

The first approach to estimating the effort for the GNOME project is based on Norden (1960) and Putnam (1978). A development project is modeled as an unknown but finite number of problems, which are solved by the manpower in events following a Poisson distribution. The number of people usefully employed at any given time is assumed to be approximately proportional to the number of problems ready for solution. Therefore, this number becomes smaller towards the end of a project as the problem space is exhausted. The learning rate of the team is modeled as a linear function of time which governs the application of effort. Following, the manpower function at a given time represents a Rayleigh-type curve governed by a parameter which plays an important role in the determination of the peak manpower. Using the relationship between time of peak manning and this parameter, the total manpower required can be determined once peak manning has been reached.

As the manpower distribution for the GNOME project has been retrieved from the data (see Fig. 1 above) and seems to follow a Rayleigh-type curve, this information can be used for estimating the effort. The peak manning of active programmers seems to have been reached between November 1998 and September 1999. Therefore the time elapsed between the beginning of the project (using January 1997) and the peak manning is set to 2.25 years, taking

Figure 2. Manpower function from data (FULL_PRO) and projected (VAR1 and VAR2).



the middle of this range. The peak manning is set to 131.8 persons, again using the mean, but needs to be converted to full-time employees, as assumed in the model. For this conversion, some value for the time actually invested in the project is necessary. The study of Hermann et al. (2000), which shows at several points similar characteristics of the programmers questioned to the data retrieved from the GNOME project is used which gives 13.9 hours per week spent per programmer. This results in a peak manning of 45.8 persons (see Fig. 2 for the resulting manpower function for the GNOME project depicted as variable FULL_PRO). Using these values in the model, a total effort of 169.9 person-years is obtained. The projected manpower function derived is also shown in Fig. 2 (depicted as VAR1). As the manpower distribution retrieved from the data shows a small level of activity until October 1997, a second approach was taken using this point as start of the project. The time of peak manning then becomes 1.42 years and the total effort is estimated as 107.2 person-years. The resulting manpower function is again shown in Fig. 2 (as VAR2).

As Putnam (1978) has shown, the time of peak manning is close to the time the software becomes operational, while effort thereafter is expended for modification and maintenance. The first major release of GNOME has been in March 1999, which coincides with peak manning empirically determined. The effort expended until this date is estimated as 66.8 person-years by the first approach, as 42.1 by the second. The results of the effort estimation for the total project presented above therefore include modification and maintenance. But as the requirements are not fixed in open source projects over time, but are expanded according to the requests of programmers and users, the estimation presented might not give a complete forecast. As a result, models for effort estimation would have to be extended to incorporate this generation of new functionality, maybe using a stochastic process. Besides this, the fact that the Rayleigh-curve proposed for commercial projects decades before closely fits the curve for a contemporary open source project (at least until time of operation) is astonishing and hints at the fact that a self-regulating community follows the theory for efficient manpower application as well (or maybe even better) than commercial management. In addition this model builds the foundation of several other estimation methods including COCOMO, which therefore might also be applicable in this context.

3.3. COCOMO

The original COCOMO (Boehm, 1981) is one of most widely used models but two assumptions seem problematic. These are a development following a waterfall-model and the permanence of the requirements during the process. Therefore the applicability in the context of open source projects seems questionable. To confirm this, the work of Londeix (1987) is used, who details how an estimation in COCOMO can be transferred to the model by Putnam (1978), i.e. how the corresponding Rayleigh-curve can be determined. In this case the other direction is employed to find the parameters in COCOMO correspond-

Copyright © 2003, Idea Group Inc. Copying or distributing in print or electronic forms without written permission of Idea Group Inc. is prohibited.

ing to the curve. As intermediate COCOMO has both the development mode and the values of cost drivers as parameters, there is no single solution. But nevertheless, even if organic mode, intended for small, in-house teams, is assumed, influence of the cost drivers would have to be more favorable than possible. Therefore the development of GNOME can not be modeled using original COCOMO, which states this development is more efficient than possible.

Therefore the successor COCOMO II (Boehm et al., 2000) seems to be a better choice, as it allows for both increasing and decreasing economies of scale, a prototype-oriented software process and flexibility in the requirements. When possible parameters are explored, the result is that once again this project is seen as very efficient as the cost drivers and scale factors replacing the modes of development in COCOMO II have to be rated rather favorably to obtain the estimated effort from the Rayleigh-curve, but this time the resulting combinations are within the range specified by the authors. If realistic values for the scale factors are used, the necessary value for the effect of the cost drivers is still within possible range.

For an additional effort estimation, the size of the GNOME project at time of operation is used with nominal values for all parameters, resulting in 612.5 person-years, and with realistic parameters resulting in 296.8 person-years, both of which are considerably higher than the results of the Rayleigh-curve. Nevertheless, while original COCOMO must be rejected in the context of open source development, COCOMO II provides for a modern type of software development and could be applicable, although it also deems this kind of development very efficient.

3.4. Function Point

While it is difficult, especially for an outsider, to correctly quantify the function points (Albrecht and Gaffney, 1983) for an open source project at the beginning, and also the requirements even from the user perspective can change during the progress, a quantification can be arrived at using the opposite way as in converting a function point count to LOC (Albrecht and Gaffney, 1983; Boehm et al., 2000). For this conversion, the mean number of LOC necessary to implement a single function point in a given programming language is provided. In GNOME, the most employed language is C, followed by Perl and C++. Therefore the overall conversion factor is estimated by using the factors from Boehm et al. (2000) for these languages with a weight of 0.7, 0.2 and 0.1, respectively, resulting in 100.5 LOC per function point. The size of GNOME at the time of operation thus corresponds to 12 200 function points.

In order to arrive at an effort estimation based on the function point count, either this measure is converted to LOC and another model like COCOMO is employed, or a relationship between function points and effort from historic projects is used. As the first approach has already been employed above, the second is taken. Using the equation provided by Albrecht and Gaffney (1983) results in an effort of 353.8 person-years. Different equations are provided by Kemerer (1987) resulting in 336.3 person-years, and by Matson et al. (1994) with a linear model resulting in 101.9 and a logarithmic model in 82.3 person-years. It seems interesting that the newer models estimate the effort as significantly less. This might be caused by the larger database containing larger projects of Matson et al. (1994) and the date of their study which allows for newer practices to be included in their results and thus resulting in stronger similarity to open source development. Nevertheless, the results which are in all cases higher than those of the Rayleigh-curve hint at a rather efficient mode of software development.

The main advantages for using function points are also of interest to their application in open source development. They are technology-independent, the user-viewpoint is considered and there is no assumption concerning the underlying software process. Therefore this metric can be used for comparisons of productivity and efficiency. Of course, additional data for open source projects need to be available.

4. Conclusion

In this paper we have discussed why effort estimation for open source development can be of interest. On the one hand, estimation of effort for completed developments is necessary for assessing their efficiency, and estimation at earlier stages can give important information to several stakeholders. The main problems in estimating this effort have been detailed, with some of them having been mitigated. Use of several approaches has been demonstrated using empirical data gathered for the GNOME project. Results indicate that open source development at least until time of operation seems to follow the model proposed for commercial projects very closely. In addition, this model can be used for estimating the effort for an open source project, although not right at the beginning, but including maintenance and modifications. The original COCOMO was dismissed both on theoretical and empirical grounds as being incompatible with open source development, while COCOMO II seemed applicable, as were function points, especially for providing a measure for technology-independent productivity comparisons. Both the results for COCOMO II and retroactive function point estimation showed that the estimations exceeded those of the Rayleigh-curve, hinting at a very efficient way of development from the viewpoints of these models.

References

Albrecht, A.J. and Gaffney, J.E. (1983) Software Function, Source Lines of Code, and Development Effort Prediction: A Software Science Validation. *IEEE Transactions on Software Engineering*, 9, 6, 639-648.

Boehm, B.W. (1981) *Software Engineering Economics*. Prentice-Hall, Englewood Cliffs, New Jersey.

Boehm, B.W. (1988) A Spiral Model for Software Development and Enhancement. *IEEE Computer*, 21, 5, 61-72.

Boehm, B.W., Abts, C., Brown, A.W., Chulani, S., Clark, B.K., Horowitz, E., Madachy, R., Reifer, D.J. and Steece, B. (2000) Software Cost Estimation with COCOMO II. Prentice Hall, Upper Saddle River, New Jersey.

Bollinger, T., Nelson, R., Self, K.M. and Turnbull, S.J. (1999) Open-source methods: Peering through the clutter. *IEEE Software*, 16, 4, 8-11.

Brooks jr., F.P. (1995) *The Mythical Man-Month: Essays on Software Engineering.* Anniversary ed., Addison-Wesley, Reading, Massachusetts.

Dempsey, B.J., Weiss, D., Jones, P. and Greenberg, J. (2002) Who is an open source software developer? *CACM*, 45, 2, 67-72.

Feller, J. and Fitzgerald, B. (2002) *Understanding Open Source Software Development*. Addison-Wesley, London.

Fielding, R.T. (1999) Shared Leadership in the Apache Project. *CACM*, 42, 4, 42-43.

Fogel, K. (1999) *Open Source Development with CVS*. CoriolisOpen Press, Scottsdale, Arizona.

Ghosh, R. and Prakash, V.V. (2000) The Orbiten Free Software Survey. *First Monday*, 5, 7.

Hermann, S., Hertel, G. and Niedner, S. (2000) Linux Study Homepage. avaible online: http://www.psychologie.uni-kiel.de/linux-study/.

Kemerer, C.F. (1987) An Empirical Validation of Software Cost Estimation Models. *CACM*, 30, 5, 416-429.

Koch, S. and Schneider, G. (2002) Effort, Cooperation and Coordination in an Open Source Software Project: GNOME. *Information Systems Journal*, 12, 1, 27-42.

Krishnamurthy, S. (2002) Cave or community? an empirical investigation of 100 mature Open Source projects. *First Monday*, 7, 6.

Londeix, B. (1987) *Cost Estimation for Software Development*. Addison-Wesley, Wokingham, UK.

Matson, J.E., Barrett, B.E. and Mellichamp, J.M. (1994) Software Development Cost Estimation Using Function Points. *IEEE Transactions on Software Engineering*, 20, 4, 275-287.

McConnell, S. (1999) Open-source methodology: Ready for prime time? *IEEE Software*, 16, 4, 6-8.

Mockus, A., Fielding, R. and Herbsleb, J. (2002) Two case studies of Open Source software development: Apache and Mozilla. *ACM Transactions on Software Engineering and Methodology*, 11, 3, 309-346.

Norden, P.V. (1960) On the anatomy of development projects. *IRE Transactions on Engineering Management*, 7, 1, 34-42.

Perens, B. (1999) The Open Source Definition. In *Open Sources: Voices from the Open Source Revolution*, DiBona, C. et al. (eds.), O'Reilly, Cambridge.

Putnam, L.H. (1978) A general empirical solution to the macro software sizing and estimating problem. *IEEE Transactions on Software Engineering*, 4, 4, 345-361.

Raymond, E.S. (1999) *The Cathedral and the Bazaar.* O'Reilly, Cambridge.

Vixie, P. (1999) Software Engineering. In *Open Sources: Voices from the Open Source Revolution*, DiBona, C. et al. (eds.), O'Reilly, Cambridge.

Copyright © 2003, Idea Group Inc. Copying or distributing in print or electronic forms without written permission of Idea Group Inc. is prohibited.

0 more pages are available in the full version of this document, which may be purchased using the "Add to Cart" button on the publisher's webpage:
www.igi-global.com/proceeding-paper/effort-estimation-open-source-software/32164

Related Content

Towards Modelling Effective Educational Games Using Multi-Domain Framework
Mifrah Ahmad, Lukman Ab Rahim, Kamisah Osmanand Noreen Izza Arshad (2018). *Encyclopedia of Information Science and Technology, Fourth Edition (pp. 3337-3347).*
www.irma-international.org/chapter/towards-modelling-effective-educational-games-using-multi-domain-framework/184045

Computer Agent Technologies in Collaborative Learning and Assessment
Yigal Rosen (2018). *Encyclopedia of Information Science and Technology, Fourth Edition (pp. 2402-2410).*
www.irma-international.org/chapter/computer-agent-technologies-in-collaborative-learning-and-assessment/183953

Object-Oriented Approaches to Causal Mapping: A Proposal
Robert F. Otondo (2005). *Causal Mapping for Research in Information Technology (pp. 343-367).*
www.irma-international.org/chapter/object-oriented-approaches-causal-mapping/6525

The Impact of Artificial Intelligence Technology for Human-Computer Interactive Industrial Robots on Labor Employment
Shuwen Jiaand Xiaoxin Chen (2025). *International Journal of Information Technologies and Systems Approach (pp. 1-16).*
www.irma-international.org/article/the-impact-of-artificial-intelligence-technology-for-human-computer-interactive-industrial-robots-on-labor-employment/395360

The Impact of Interdisciplinary Teaching Behavior and Deep Learning: A Case Study of a Private University
Lingjun Duand Jing Li (2026). *International Journal of Information Technologies and Systems Approach (pp. 1-16).*
www.irma-international.org/article/the-impact-of-interdisciplinary-teaching-behavior-and-deep-learning/404002