Effort, co-operation and co-ordination in an open source software project: GNOME

Stefan Koch & Georg Schneider

Department of Information Business, Vienna University of Economics and Business Administration, Augasse 2–6, A-1090 Vienna, Austria, email: stefan.koch@wu-wien.ac.at

Abstract. *This paper presents results from research into open source projects from a software engineering perspective. The research methodology employed relies on public data retrieved from the CVS repository of the GNOME project and relevant discussion groups. This methodology is described, and results concerning the special characteristics of open source software development are given. These data are used for a first approach to estimating the total effort to be expended.*

Keywords: Case study, effort estimation, open source software, software development, software metrics

INTRODUCTION

Open source software development (Raymond, 1999) has generated much interest in recent years, especially since the rise of Linux. As several similar projects exist, such as the GNU project's utilities and libraries, the Perl and Tcl programming languages and the Apache web server, research from a software engineering perspective on this decentralized form of software development should be intensified. As a first step, it seems necessary to assess the differences and characteristics of such projects and their special organizational form (Vixie, 1999). Therefore, quantitative research into this form of collaborative development is necessary, which today is very scarce (Dempsey *et al*., 1999; Ghosh & Prakash, 2000; Hermann *et al*., 2000; Mockus *et al*., 2000).

Open source software is characterized by several differences from traditional software development and distribution, including the free redistribution, the inclusion of the source code, the possibility for modifications and derived works, which must be allowed to be distributed under the same terms as the original software, and some others (Perens, 1999). One example of a licence that fits these criteria is the well-known GNU general public licence (GPL). 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 (Perpich *et al*., 1997; Dempsey *et al*., 1999).

In this paper, we present results from quantitative research into an open source project using data retrieved from a publicly available CVS repository and discussion lists. We describe the methodology, how it was used in the GNOME project and results concerning the effort expended by the participants, the files constituting this development effort, usage of mechanisms for co-ordination and co-operation, together with the progression of the project over time. These results are used in a first approach to effort estimation of the project under consideration.

RESEARCH METHODOLOGY

The main idea for this research into open source software development was to use existing data on the project available to the public (Cook *et al*., 1998). Therefore, the CVS repository of the GNOME project was used for data collection. It has been demonstrated that several important aspects of a large-scale open source project could be checked using this source (Atkins *et al*., 1999; Mockus *et al*., 2000). In particular, data concerning the participants' contributions to the project, their co-operation, which could only be analysed on the basis of single files, and the progression of the project in size and participants over time were of interest.

CVS (concurrent versions system) is a version control system, which is being used extensively in the free software community (Fogel, 1999). Access is accomplished via a client, which requires a password authentification. In order to access CVS archives in a more convenient way, the Mozilla project developed Bonsai, which allows connections using a web-based interface (Fielding, 1999).

GNOME, the GNU network object model environment, is an open source software project building a desktop environment for users and an application framework for software developers. This vendor-neutral project includes a set of standard desktop tools and applications, e.g. the well-known GNU image manipulation program (GIMP), and uses the common object request broker architecture (CORBA).

Discussion lists were identified as an additional source of information besides the CVS repository. The conclusions to be drawn based on these data included the usage of this medium for co-ordination by different participants and the respective changes over time, for example during manpower build-up.

As all data retrieved needed to be managed, storage in a database was chosen. Therefore, a data model of an open source software project was developed to include all publicly available data.

The following clarifications seem to be necessary: There exist coders (or programmers) who actually work on the project by submitting ('checking in') files. On the other hand, there are posters who participate in discussions pertaining to the software. One real-world person can fulfil both roles, but the possibility exists for people only to post messages in discussion lists or for programmers who do not participate in discussions. A file, as identified by a filename and a directory path (which is necessary as some filenames are duplicates, e.g. a file named

'makefile' exists in several directories), can be checked into the CVS system by a programmer. The CVS repository then records this checkin with the changes in the lines-of-code (LOC) and further information. A posting is a message to a discussion list pertaining to the GNOME project, maybe in reply to a prior posting. A module consists of several files and constitutes a self-contained part of the GNOME project (e.g. gnome-core, gnumeric, gimp).

As a first step, the web interface of the CVS repository was used to retrieve the necessary data concerning checkins. These data included, for every checkin programmer, file, date, LOC added and deleted, revision number and some comment. This was done using a Perl script, which generated successive queries simulating a browser-based input form. Each query spanned an interval of one day in the history of the CVS archive, starting with the earliest entries at the start of the project. The requests were distributed over a 4-day period in order not to overload the server. The result of each individual query was an HTML page, which was subsequently parsed extracting the necessary attributes conforming to the data model. This information was then stored in a database (Postgresql under Linux). The necessary queries were then performed and the output analysed using a statistical package. Of course, the data concerning programmers was strictly anonymized.

Also retrieved by a Perl script were the postings to the relevant discussion lists including the sender, the subject, time and complete text. For analysis of the posting behaviour of the programmers, the short name that each programmer uses for checkins had to be matched to the full name or email address used for postings. For 175 persons, this has been possible using several regular expressions with human check-up.

This approach of using existing information publicly available eliminated one of the most pressing problems in software engineering research, the lack of data, especially concerning the past history of projects. In addition, this approach does not intrude on the software project under consideration and is inexpensive (Cook *et al*., 1998; Atkins *et al*., 1999).

RESULTS

This section details the results from analysis of the CVS repository data and the discussion lists. For the most part, these results concern the programmers, e.g. the effort expended on the project, and the files composing the software development effort. In addition, the changes in the project over time are explored. Before these results are presented, the metrics used in the analysis are described.

Metrics used

The first metric used is the number of lines-of-code (LOC) added to a file. The definition of this often-disputed metric LOC (Park, 1992; Humphrey, 1995) is taken from the CVS repository and therefore includes all types of LOC, e.g. also commentaries (Fogel, 1999). In addition, any LOC changed is counted as one line-of-code added and one line-of-code deleted. The grand total of LOCs added was 6 300 000 for the whole project and all programmers

during the time under consideration from the beginning of the project. The next metric is defined analogously and pertains to the LOC deleted. For the whole project, 4 500 000 LOCs have been deleted. The difference between the LOC added and the LOC deleted therefore gives the change in the size of the software artefact under consideration in the corresponding time period. These changes can be cumulated to give the size at any moment. The metric of checkin refers to the submission of a single file by a single programmer. Overall, 220 000 checkins have been made for the project. The total time spent on the project is defined for every programmer as the difference between the date of his first and his last checkin. As this therefore includes all time elapsed, not necessarily only time spent actually working on the project, this measure can only give an upper bound for actual time spent working [no time sheet data, as used by Atkins *et al*. (1999), are available]. It will be shown in the analysis that this measure is indeed not usable for predicting the output of a given programmer. When the results from a questionnaire to Linux developers (Hermann *et al*., 2000) are taken, which give a mean of 13.9 h per week spent on open source development, the total effort computed from the time spent on the project by all programmers is about 145 000 person–hours of effort (or 954 person–months). As a further definition, a programmer is defined as being active in a given period of time if he performed at least one checkin during this interval. Therefore, programmers will not contribute to this number during large periods of inactivity. For each file, the time worked on is also defined as the time elapsed between the first and the last checkin. The same considerations of course apply as for programmers, i.e. this measure can also only be taken as an upper bound.

In addition, the postings made to mailing lists pertaining to the GNOME project were analysed. The most important metric derived is the number of postings. The grand total of postings observed was 19 909.

Data on persons involved in the project

Open source software development is assumed to be performed by more people than traditional development, as these do not spend all their time working on the project. In the GNOME project, 301 programmers were identified who currently work, or have worked upon, this software. As will be shown, these programmers differ significantly in their effort for this software project. For additional data on contributors to open source development, including their country of origin, see Dempsey *et al*. (1999) and Hermann *et al.* (2000).

A single programmer added on average 21 000 LOCs with a standard deviation of 67 000. The corresponding values for LOCs deleted were 15 000 and 48 000. The maxima were 931 000 for LOCs added and 621 000 for LOCs deleted. Figure 1 shows the corresponding histogram. As can be seen from these results, there are indeed significant differences between the programmers, with a majority contributing a quite small amount, a result also found by Dempsey *et al*. (1999). A case study of the development of the Apache web server showed that the top 15 programmers added 88% of the LOCs (Mockus *et al*., 2000). In contrast, the top 15 programmers for the GNOME project were only responsible for 48%, whereas the top 52 persons were necessary to reach 80%. A clustering of the programmers based on the

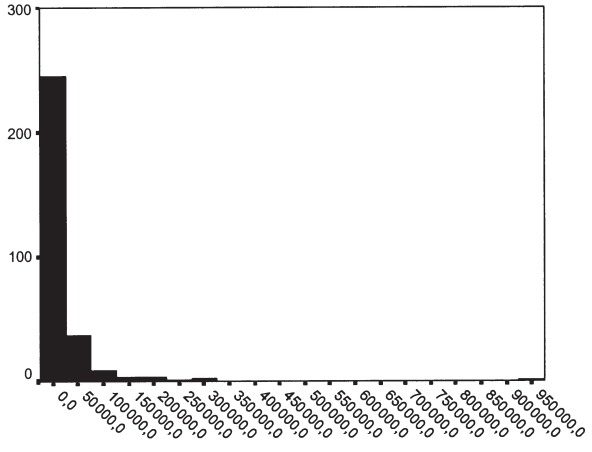

Figure 1. Histogram of LOCs added per programmer.

LOCs added hinted at the existence of a smaller group of 11 programmers within this larger group, who were still more active. It is therefore possible that such a small group numbering 10–15 persons, which still allows for easy communication and co-operation, shows up in open source projects, even if it is not responsible for 80% of the total code, as was proposed by Mockus *et al.* (2000). A similar distribution for the LOC contributed to the project was found in a community of Linux kernel developers by Hermann *et al.* (2000). This might give a hint that other results from their study could also be used in the context of the GNOME project.

The number of checkins performed by a programmer was on average 731 with a standard deviation of 1857. The time spent on the project had a mean of 246 days and a standard deviation of 213 (Figure 2). Compared with the data concerning the number of LOCs added and deleted, the differences between the programmers seem to be much smaller. This finding will be explored further later on. The LOCs added per single checkin had a mean of 28, the LOCs deleted 20.

Next, possible relations between these variables were explored. Besides a high correlation between the number of LOCs added and the number of LOCs deleted, which is not surprising as a lot of changes to the code are included, there is also a high degree of correlation to be found between the LOCs added and the number of checkins, so it can be said that programmers who add more LOCs also use more checkins. As the correlation between the total LOCs added and the LOCs added per checkin is very low, there is no relation stating that very active programmers use bigger checkins, i.e. those containing more LOCs. This could be taken as an indication that there is no significant difference in programming style between programmers.

As the scatterplot suggests (Figure 3), there is no strong relation between time spent on the project and the total number of LOCs added. Therefore, it cannot be said that programmers who stay on the project longer also contribute significantly more output. A similarly low

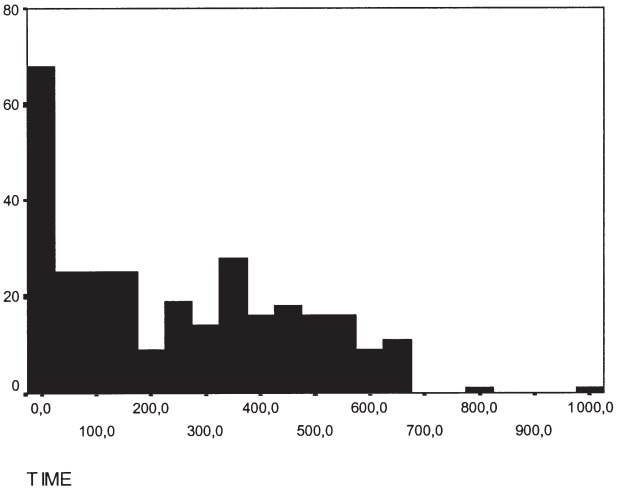

Figure 2. Histogram of time spent on the project for each programmer.

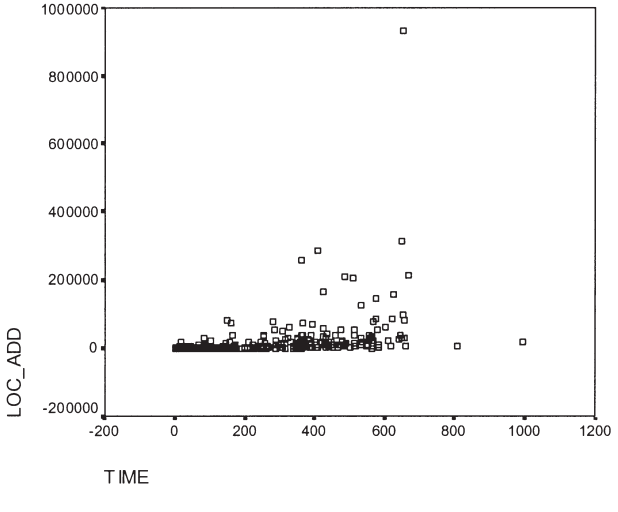

Figure 3. LOCs added against time spent on each project for each programmer.

correlation was found by Hermann *et al*. (2000). It is to be assumed that this constitutes a difference from commercial software development, where people spend all their (working) time on the project, thereby constantly generating output, whereas participants in open source development only donate some of their spare time, most often irregularly, to this effort. Mockus *et al*. (2000) have shown that several commercial projects indeed show less variation in developer contributions.

Data on the postings made to several discussion lists pertaining to the GNOME project shows 19 909 postings made, of which 6903 had been replies to other postings. A total of

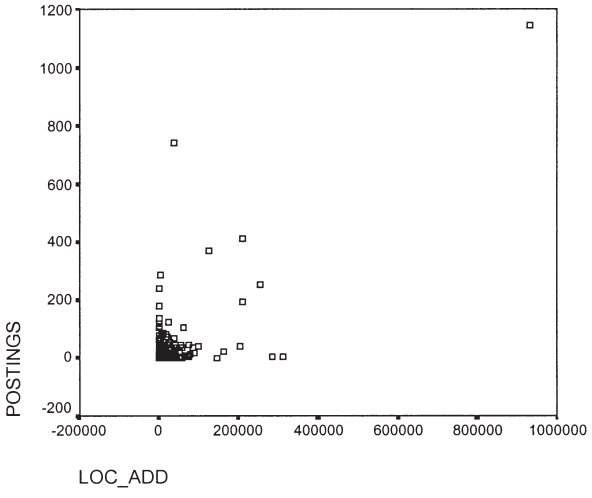

Figure 4. LOCs added against postings made for each programmer.

1881 different posters has been identified. The mean number of postings per person is therefore 11.

Data on the postings made by identified programmers were also explored. Each programmer posted a mean of 43 messages with a standard deviation of 116. The distribution is again similar to the number of LOCs added. These programmers in sum contributed 7455 messages out of the total of 19 909 messages retrieved from the discussion lists. The mean number of messages for the programmers is significantly higher than for all posters with 43 to 11. For the group of programmers, as the scatterplot indicates (Figure 4), a correlation of 0.691 could be found between number of postings and sum of LOCs added, showing that more productive programmers are also more active participants in the discussion lists.

Data on files

Since the beginning of the GNOME project, 38 634 files have been worked on. Of course, these files differ significantly in size and complexity. The total LOCs added to any file was taken as a metric for its size, which on average for a given file was 163 with a standard deviation of 1136 and a maximum of 60 000 (see Figure 5 for the histogram). The LOCs deleted were found to have a similar distribution with a mean of 117 and a standard deviation of 984.

The number of checkins for a given file had a mean of six with a standard deviation of 18. The time worked on a given file had a mean of 95 days (standard deviation 152 days).

A correlation between the LOCs added and the number of checkins exists but is not very strong (Pearson correlation of 0.341). As a stronger correlation of 0.602 can be found between the total LOCs added to a file and the LOCs added per single checkin, this seems to indicate that larger files are checked in using larger chunks. In addition, if the total number of checkins is taken as an indicator of complexity, as more complex files necessitate more changes

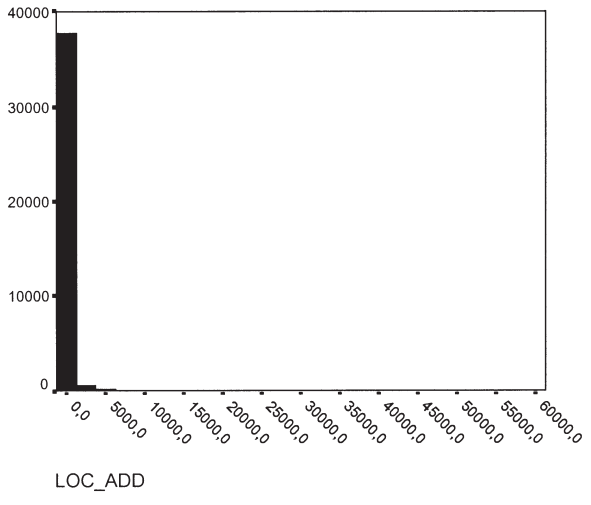

Figure 5. Histogram of LOCs added per file.

and therefore checkins, there is no strong correlation with size, as taken from the total LOCs added.

As for programmers, the relation between time and total effort (as measured in the sum of LOCs added) is not very strong, so the time spent between first and last checkin of a single file is not a good predictor of the total effort spent on this file.

If the associations between programmers and files are examined, it can be seen that there is a low level of co-operation on the basis of files. Only a small number of programmers work together on a given file, hinting at a division of labour at higher levels of abstraction. This number is higher for bigger files, although not in appropriate relation. More active programmers do a higher percentage of their total work on these larger files, and it can be assumed that, within this smaller circle of persons, co-operation is possible at this microlevel.

Data on progression of the project over time

The progression of the GNOME project over time was also explored to gain an insight into the evolution of this software (Figure 6).

As can be seen, the total size of the GNOME project has experienced a steady increase, with the cumulated LOC difference over time taken to gain an understanding of this progression. In accordance with Norden (1960), Putnam (1978) and Parr (1980), the development is assumed to become slower towards the end of the life cycle. A flattened end to the plotted curve is therefore a hint that the project is nearing completion. This point does not seem to have been reached yet for the GNOME project.

For each module, the progression over time was also analysed, i.e. viewing each one as a separate project. This analysis was undertaken to uncover whether there were differences in the projects in their progression in the life cycle. The cumulated LOC difference over time was

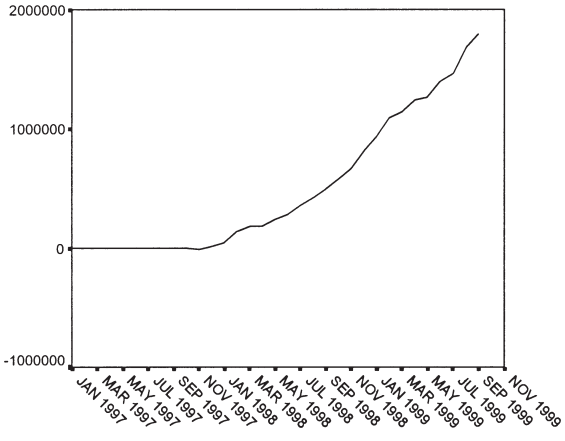

Figure 6. LOC differences in each month cumulated.

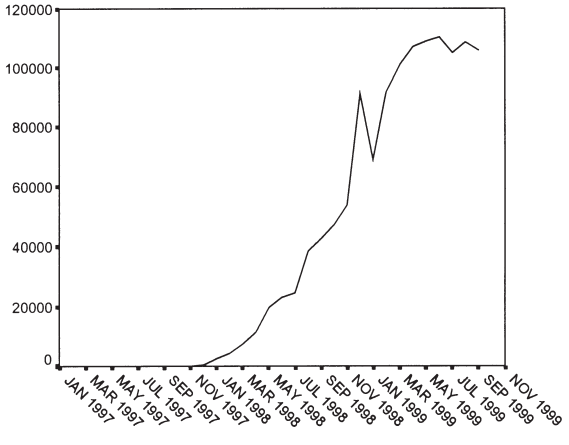

Figure 7. LOC differences in each month for gnome-core cumulated.

again taken to gain an understanding of the growth in size of any module with a flattened end to the plotted curve as a hint of completion. There are indications that several of these modules have already progressed to a later stage in the life cycle. For example, the size of the module gnome-core (Figure 7), which constitutes a basis for several others, seems to have stabilized at this time. In contrast, gimp does not yet seem to have entered this stage (Figure 8). These results provide support for the notion that the total growth of the GNOME project is in different time periods supported by different parts, i.e. that the life cycle of the GNOME project is composed of several subcycles of projects whose starting points are shifted in time and which progress at different speeds.

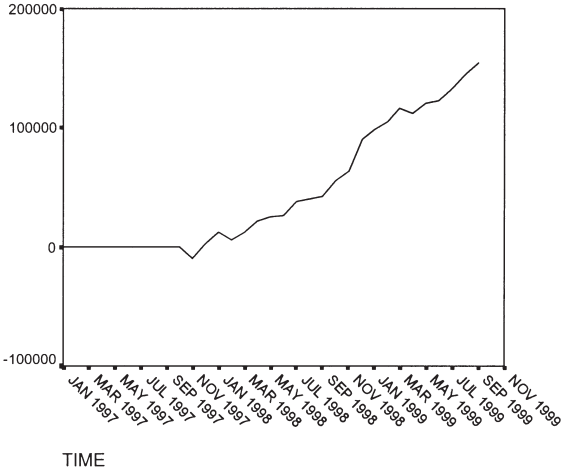

Figure 8. LOC differences in each month for gimp cumulated.

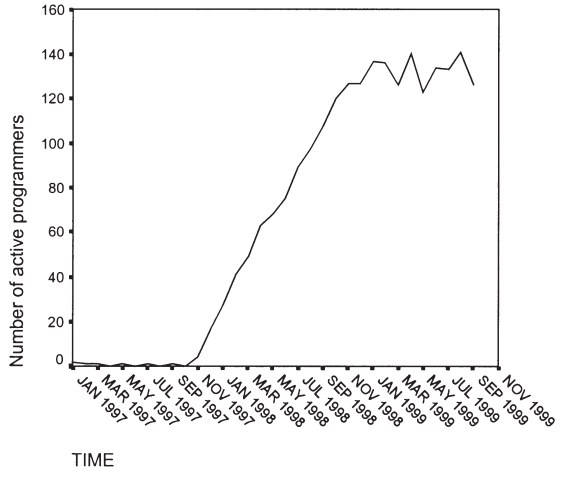

Figure 9. Number of active programmers each month.

The most important aspect in open source development projects is the participation of programmers, as a higher number of contributors will also probably lead to more output. As can be seen (Figure 9), the number of active programmers, defined as being active within the time frame of 1 month, has seen a staggering rise between November 1997 and the end of 1998. During 1999, this number was roughly constant at around 130 persons. The factors resulting in this development cannot be seen from the data retrieved for this research, but it can be assumed that marketing-like instruments, such as coverage in the open source community, have played an important role. Also, many developers will be more likely to join an open source project at the beginning (or during take-off) than at the end, as development will be more highly

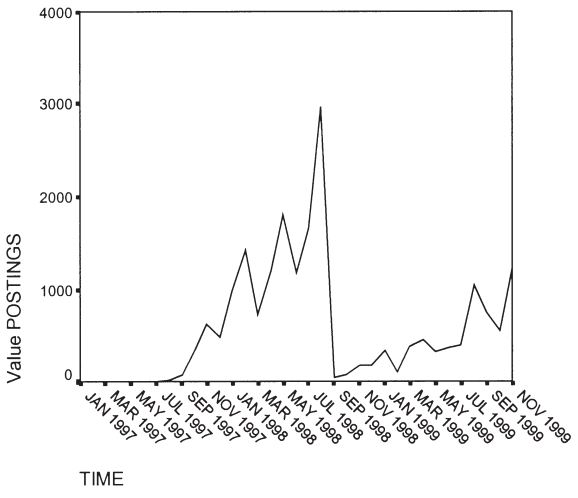

Figure 10. Total of postings made in each month.

regarded than maintenance and personal influence on the whole project could be greater (Raymond, 1999). It can be seen that members of the more active group of programmers did indeed join the GNOME project earlier. Another reason for this development in active programmers can be seen from the research of Norden (1960), Putnam (1978) and Parr (1980), who argue that only a given number of persons can be working on a project in a productive manner at a given time (in relation to the problems ready for solution at this point). In view of this interpretation, the peak manning of the project has already been reached and will only see a downfall from now on. It is also possible that the organization of more programmers than the number active in 1999 is not feasible, given the structures in place at the moment (e.g. the CVS system), i.e. that this boundary is not inherent in the problem worked on. This would lead to the conclusion that more effective organizational structures would allow more programmers to work on the project and thus lead to higher output. More work should then be invested in the design of such structures.

A correlation of 0.932 was found between the total LOCs added and the number of active programmers each month. This confirms the intuitive relationship between these two variables.

The number of new postings made each month was also explored to gain an understanding of the development of activity in the discussion lists over time. As can be seen, the activity was strongest in 1998, which coincided with the build-up in active programmers (Figure 10). A possible explanation would be that more communication and co-ordination using this medium is necessary to accommodate new programmers joining the project (Abdel-Hamid & Madnick, 1991; Brooks, 1995). After they have joined, less interaction is necessary between them. Therefore, the correlation between the total postings for each month and the difference in active programmers was also explored, which yielded a result of 0.366.

The correlation of the number of postings as a measure of activity in the discussion lists with the total LOCs added in each month as a measure of programming effort was

also explored. A correlation of 0.227 was found, not supporting a relationship between activity in the discussion lists and programming effort expended on the GNOME project overall.

EFFORT ESTIMATION

In this section, a first approach to estimating the effort on the GNOME project is detailed based on Norden (1960) and Putnam (1978). According to this approach, a development project is modelled as a series of problem-solving efforts by the manpower involved to reach a set of objectives constituting technological progress. The number of problems is assumed to be unknown but finite. Each solving of a problem removes one element from the list of unsolved problems. The occurrence of such an event is random and independent and is assumed to follow 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 at that time. Therefore, the manpower usefully employed towards the end of a project becomes smaller as the problem space is exhausted. The learning rate of the team is modelled as a linear function of time, which governs the application of effort. Therefore, the cumulative manpower effort is null at the start of the project and grows monotonically towards the total effort. Accordingly, the manpower function at a given time represents a Rayleigh-type curve governed by a parameter that plays an important role in the determination of peak manpower. By deriving the manpower function relative to the time and finding the zero value, the relationship between time of peak manning and this parameter can be found. Furthermore, the value of peak manning can be obtained by substituting this value in the manpower function. Using this relationship, 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 the preceding section and Figure 9) and seems to follow a Rayleigh-type curve, this information can be used to estimate the total 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 peak manning is set at 2.25 years, taking the middle of this range. The peak manning is set to 131.8 persons, the mean staffing in these months. The next necessary step is to convert the peak manning to full-time employees, as this type is assumed in the model used. For this conversion, some value for the time actually invested in the project is necessary. As has been shown, the study by Hermann *et al*. (2000) showed at several points similar characteristics of the programmers questioned to the data retrieved from the GNOME project. Therefore, it is possible to use the resulting value of 13.9 h per week spent on the project. This value is only applied to the active programmers, i.e. those who have shown activity within the chosen time frame of 1 month. This results in a conversion to a peak manning of 45.8 persons (see Figure 11 for the resulting manpower function for the GNOME project depicted as variable FULL_PRO). Using these values in the model of Norden (1960) and Putnam (1978), a total effort of 169.9 person–years

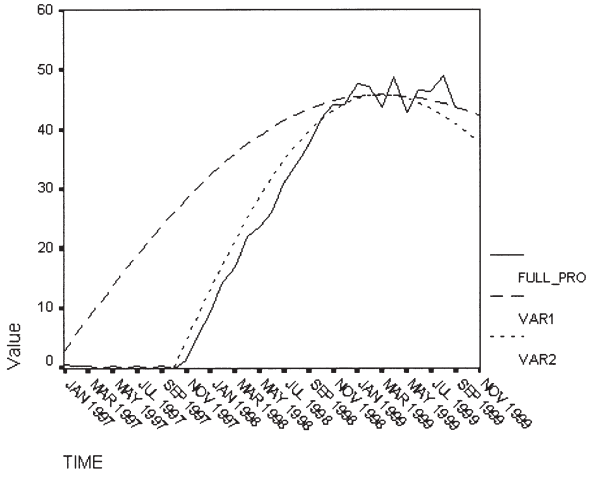

Figure 11. Manpower function from data (FULL_PRO) and projected (VAR1 and VAR2).

is obtained. The projected manpower function derived is also shown in Figure 11 (depicted as VAR1).

As the manpower distribution retrieved from the data shows a very small level of activity until October 1997, a second approach was taken using this point in time as the start of the project. The time of peak manning then becomes 1.42 years, and the total effort to be expended is estimated as 107.2 person–years. The resulting manpower function is again shown in Figure 11 for comparison (as VAR2).

As Putnam (1978) has shown, the time of peak manning is very close to the time when the software becomes operational, whereas effort thereafter is expended for modification and maintenance. The first major release of GNOME was in March 1999, which also coincided with the peak manning determined empirically. The results of the effort estimation presented above therefore reflect this assumption. On the other hand, a major difference from commercial software development is that, in open source projects, the requirements are not fixed over the life time of the software. According to the requests of programmers and especially users, new functionality is added. This violates the assumptions of most traditional models for software development effort estimation. The estimation presented also might therefore not give a complete forecast of future effort. This will have to be checked at a later date. As a result, models of effort estimation would have to be extended to incorporate this generation of new functionality during the course of an open source project, maybe using a stochastic process, whose type and parameters will have to be determined using empirical data.

CONCLUSION

Open source software development has become very important to the software industry. Therefore, research into this field from the perspective of software engineering also gains in

importance, especially quantitative data on this collaborative form of development are needed (Dempsey *et al*., 1999). This paper has presented the usage of a methodology that could be applied to several other open source software development projects. The results obtained in this study and other similar work (Mockus *et al*., 2000) have shown that insights into this kind of development can indeed be gained.

In open source development, more people are involved than in traditional organizational forms, but the data show the existence of a relatively small 'inner circle' of programmers responsible for most of the output. Those programmers are also the more active participants in the discussions pertaining to the project, although all programmers show a higher than average activity in the mailing lists compared with other participants. The size of this group does not seem to be fixed. Regarding the development of the Apache web server, it numbered 15 people (Mockus *et al*., 2000), within the GNOME project 52 persons (if contribution of 80% of the total code is used as the boundary). But evidence for a still smaller group of 11 people also exists in this case, even if they were not responsible for a similar proportion of the code. It might be assumed that a group of this size constitutes itself because of less effort in co-ordination and communication, even if the contributions are more dispersed as in the GNOME project. There is no relation to be seen between the time between first and last activity for the project and the output produced. This seems to be another striking difference from traditional software development. There is only a small number of programmers working together on a file, indicating a high degree of division of labour.

It has been shown that the project under consideration has seen a steady increase in size, as measured in LOCs over the time inspected. There is no indication that the end of the life cycle has yet been reached. Analysis of the subprojects has given support to the theory that some of these have already progressed to a later stage in their evolution, whereas others lag behind. Therefore, the growth of the GNOME project is in different time periods supported by different projects, i.e. the life cycle of the GNOME project is composed of several subcycles of projects whose starting points are shifted in time.

The number of active programmers has seen a staggering rise during a prolonged time period, but has been relatively stable for the last year. The reasons for this might be inherent in the problem worked on or may indicate deficiencies in the form of co-operation used. Also, more psychologically motivated factors might have contributed to this effect. It seems interesting that the highest amount of activity in the discussion groups has been seen during the time of the manpower build-up, hinting at a more pressing need for communication and co-ordination. The intuitive relationship between the number of active programmers and the output produced for the project was confirmed. The attraction of participants is therefore identified as one of the most important aspects of open source development projects.

A first attempt at estimating the effort for this open source project has been presented. The results seemed to give realistic numbers and therefore indicated that the theory of Norden (1960) and Putnam (1978) is applicable to this type of software development, although some conversions are necessary. Since in open source software projects the requirements are not fixed but, instead, expanded continuously, the results might not give a complete forecast of future effort, which will have to be checked at a later date. This generation of new function-

ality might need to be incorporated in future models of effort estimation in this context, possibly using some form of stochastic process.

Further research is to be undertaken using the data collected to this point, and further sources of data still need to be explored. For example, content analysis of the postings retrieved can yield further important information concerning interactions between software developers (Brooks, 1995; Seaman & Basili, 1997), the diffusion of information and several other communication metrics (Dutoit & Bruegge, 1998). The bug-tracking archive can supply data on software quality. In addition, the source code for each file could be retrieved and analysed using measures such as cyclomatic complexity (McCabe, 1976). These results could be correlated with, e.g. LOC or number of checkins, to gain insights into the relationships between size, complexity and maintenance effort (Banker *et al*., 1993).

In a next step, the methodology described needs to be applied to other open source software development efforts to allow for comparison between projects and the discovery of common features. This profile can then be compared with data from commercial software development projects. To ensure the validity of this comparison, the necessary metrics of course need to be available for several of these traditional software projects. As some sort of versioning control system will almost certainly also be used in this organizational form of software development, the same methodology can again be applied to some extent. Therefore, understanding of both the new form of open source software development and the more traditional organizational approaches could be enhanced.

REFERENCES

Abdel-Hamid, T. & Madnick, S.E. (1991) *Software Project Dynamics: an Integrated Approach*. Prentice Hall, Englewood Cliffs, NJ.

Atkins, D., Ball, T., Graves, T. & Mockus, A. (1999) Using version control data to evaluate the impact of software tools. In: *Proceedings of the 21st International Conference on Software Engineering*, pp. 324–333.

Banker, R.D., Datar, S.M., Kemerer, C.F. & Zweig, D. (1993) Software complexity and maintenance costs. *Communications of the ACM*, **36** (11), 81–94.

Brooks, F.P., Jr (1995) *The Mythical Man-Month. Essays on Software Engineering.* Anniversary edn. Addison-Wesley, Reading, MA.

Cook, J.E., Votta, L.G. & Wolf, A.L. (1998) Cost-effective analysis of in-place software processes. *IEEE Transactions on Software Engineering*, **24**, 650–663.

Dempsey, B.J., Weiss, D., Jones, P. & Greenberg, J. (1999) *A Quantitative Profile of a Community of Open Source Linux Developers.* Technical Report TR-1999–05. School of Information and Library Science, University of North Carolina at Chapel Hill.

Dutoit, A.H. & Bruegge, B. (1998) Communication metrics for software development. *IEEE Transactions on Software Engineering*, **24**, 615–628.

Fielding, R.T. (1999) Shared leadership in the Apache project. *Communications of the ACM*, **42** (4), 42–43.

Fogel, K. (1999) *Open Source Development with CVS*. Coriolis Open Press, Scottsdale, AZ.

Ghosh, R. & Prakash, V.V. (2000) The Orbiten free software survey. *First Monday*, **5**, 7.

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

Humphrey, W.S. (1995) *A Discipline for Software Engineering*. Addison-Wesley, Reading, MA.

McCabe, T.J. (1976) A complexity measure. *IEEE Transactions on Software Engineering*, **2**, 308–320.

Mockus, A., Fielding, R. & Herbsleb, J. (2000) A case study of open source software development: the Apache server. In: *Proceedings of the 22nd International Conference on Software Engineering*, pp. 263–272.

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

Park, R.E. (1992) *Software Size Measurement: A Framework for Counting Source Statements.* Technical Report CMU/SEI-92-TR-20. Software Engineering Institute, Carnegie Mellon University, Pittsburgh.

Parr, F.N. (1980) An alternative to the Rayleigh curve model for software development effort. *IEEE Transactions on Software Engineering*, **6**, 291–296.

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

Perpich, J.M., Perry, D.E., Porter, A.A., Votta, L.G. & Wade, M.W. (1997) Anywhere, anytime code inspections: using the web to remove inspection bottlenecks in large-scale software development. In: *Proceedings of the 19th International Conference on Software Engineering*, pp. 14–21.

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

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

Seaman, C.B. & Basili, V.R. (1997) An empirical study of communication in code inspection. In: *Proceedings of the 19th International Conference on Software Engineering*, pp. 96–106.

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

Biographies

Stefan Koch is Assistant Professor of Information Business at the Vienna University of Economics and Business Administration. He received an MBA in Management Information Systems from Vienna University and Vienna Technical University, and a PhD from Vienna University of Economics and Business Administration. Currently, he is involved in the undergraduate and graduate teaching programme, especially in software project management and ERP packages. His research interests include cost estimation for software projects, the open source development model, software process improvement, the evaluation of benefits from information systems and ERP systems.

Georg Schneider received a Master's degree from the University of Economics and Business Administration in Vienna. He worked at the Institute of Information Business specializing in digital libraries and is now a researcher in bioinformatics at the Institute of Molecular Pathology, Vienna.