Exploring the Effects of Coordination and Communication Tools on the Efficiency of Open Source Projects using Data Envelopment Analysis

Stefan Koch
Vienna University of Economics and Business Administration, Institute for Information Business
Augasse 2-6, A-1090 Vienna, Austria
stefan.koch@wu-wien.ac.at
WWW home page: http://wwwai.wu-wien.ac.at/~koch/

Abstract. In this paper, we propose to explore possible benefits of communication and coordination tools in open source projects using data envelopment analysis (DEA), a general method for efficiency comparisons. DEA offers several advantages: It is a non-parametric optimization method without any need for the user to define any relations between different factors or a production function, can account for economies or diseconwhile omies of scale, and is able to deal with multi-input, multi-output systems in which the factors have different scales. Using a data set of 30 open source project retrieved from SourceForge.net, we demonstrate the application of DEA, showing that the efficiency of the projects is in general relatively high. Regarding the effects of tool employment on the efficiency of projects, the results were surprising: Most of the possible tools, and overall usage, showed a negative relationship to efficiency.

Keywords. Open Source Software Development, Efficiency, Data Envelopment Analysis, Software Repositories

1 Introduction

Considerable uncertainty has for a long time surrounded the topic of the efficiency of open source software development, and the factors influencing this efficiency. Currently, any comparison of open source software projects is very difficult. There is increased discussion on how the success of open source projects can be defined [21,22,9,10], using for example search engine results as proxies [23]. In addition, the process applied in these projects can differ significantly, and several elements of

Please use the following format when citing this chapter:

Koch, S., 2007, in IFIP International Federation for Information Processing, Volume 234, Open Source Development, Adoption and Innovation, eds. J. Feller, Fitzgerald, B., Scacchi, W., Sillitti, A., (Boston: Springer), pp. 97–108.

both process and infrastructure could have an impact. For example, [19] has used a sample of projects from SourceForge.net to uncover whether the process maturity has had any on success of open source projects. In this analysis, the notion of success was based on the downloads achieved, and a relationship to version control, mailing lists and testing strategies was found.
In this paper, we apply the method of Data Envelopment Analysis (DEA) to compare open source projects according to their efficiency in transforming inputs into outputs. For any production process, this efficiency and productivity is a key indicator in comparison to other processes. DEA is a non-parametric optimization method for efficiency comparisons without any need for the user to define any relations between different factors or a production function. In addition, DEA can account for economies or diseconomies of scale, and is able to deal with multi-input, multi-output systems in which the factors have different scales.
Efficiency and productivity in software development is most often denoted by the relation of an effort measure to an output measure, using either lines-of-code or, preferably due to independence from programming language, function points [1]. This approach can be problematic even in an environment of commercial software development due to missing components especially of the output, for example also [15] agree that productivity measures need to be based on multiple size measures. In open source development, there are additional problems which point towards DEA as an appropriate method.

In open source projects, normally the effort invested is unknown, and therefore might need to be estimated [2,16,17], and is also more diverse than in commercial projects, as it includes core team member, committers, bug reporters and several other groups with varying intensity of participation. Besides that, also the outputs can be more diverse. In the general case, the inputs of an open source project can encompass a set of metrics, especially concerned with the participants. So, in the most simple case, the number of programmers and other participants can be used. The output of a project can be measured using several software metrics, most easily the number of LOC, files, or others. This range of metrics both for inputs and outputs, and their different scales necessitates application of an appropriate method like DEA.

The main result of applying DEA for a set of projects is an efficiency score for each project. This score can serve different purposes: First, single projects can be compared accordingly, but also groups of projects, for example those following similar process models, located in different application domains or simply of different scale can be compared to determine whether any of these characteristics lead to higher efficiency.

In a prior paper, DEA has been explored in this context, but with a different dataset mostly relying on in-depth CVS analysis [18]. This has demonstrated that DEA can indeed be applied in this context, and has also shown that neither inequality in contributions, nor licensing scheme nor intended audience have a significant impact on efficiency. In this paper, we will employ the results of a DEA to investigate whether the adoption of communication and coordination tools like mailing lists or particular source code control systems have any impact on efficiency.

2 Data Envelopment Analysis

The principle of the border production function was introduced by Farell for measuring the technical efficiency [12] and enhanced by Charnes, Cooper and Rhodes [6] into the first Data Envelopment Analysis model (the CCR model). The object of analysis the DEA considers is very generally termed Decision Making Unit (DMU). This term includes relatively flexibly each unit which is responsible for the transformation of inputs into outputs, for example hospitals, supermarkets, schools, bank branches and others.

The basic principle of DEA can be understood as a generalization of the normal efficiency evaluation by means of the relationship from an output to an input into the general case of a multi-output, multi-input system without any given conversion rates or same units for all factors. In contrast to other approaches, which require the parametric specification of a production function, DEA measures production behavior directly and uses this data for the evaluation of all DMUs. The DEA derives a production function from mean relations between inputs and outputs (whereby it is only assumed that the relation is monotonous and concave), by determining the outside cover of all production relations (see also Figure 1), while for example a regression analysis estimates a straight line through the center of all production relations. The DEA identifies "best practicing" DMUs, which lie on the production border. A DMU is understood as being efficient if none of the outputs can be increased, without either or several of the inputs increasing or other outputs being reduced, as well as vice versa.

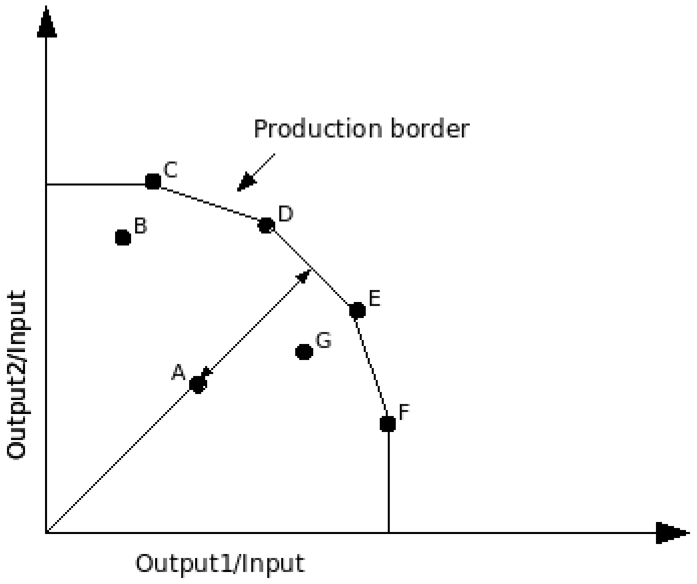

Fig. 1. Data Envelopment Analysis for the case of one input and two outputs with 7 DMUs (A – G), out of which C – F are efficient, and depicting inefficiency of A for which D and E form the reference set

For each DMU an individual weighting procedure is used over all inputs and outputs. These form a weighted positive linear combination, whereby the weights are specified in such a way that they maximize the production relationship of the examined unit, in order to let these become as efficient as possible. The efficiency of an examined unit is limited with 1. That means that no a-priori weightings are made by the user, and that the weights between the DMUs can be different. For each evaluation object the DEA supplies a solution vector of weighting factors and a DEA efficiency score. If this score is equal to 1, then the DMU is DEA efficient. In this context, DEA efficiency means that no weighting vector could be found which would have led to a higher efficiency value. DEA efficient are thus all those DMUs which are not clearly DEA inefficient compared with the others. Any inefficiency can therefore not be ruled out completely. For each inefficient DMU the DEA returns a set of efficient DMUs which exhibit a similar input/output structure and lie on the production border near to the inefficient DMU (reference set, see also Figure 1). Using this information, an idea in which direction an increase in efficiency is possible can be gained.

The first model of the DEA was introduced by Charnes, Cooper and Rhodes [7] and is therefore designated as CCR model. They pose four assumptions for the production possibility set, which are convexity, possibility for inefficient production, constant returns to scale and minimum extrapolation. The different basic models of

the DEA can be divided on the basis of two criteria: This is on the one hand the orientation of the model, on the other hand the underlying assumption regarding the returns to scale of the production process. With input-oriented models the reduction of the input vector maximally possible with the given manufacturing technology is determined, whereas with output-oriented models the maximally possible proportional increase of the output vector is determined. The returns to scale can be assumed either as being constant or variable. With constant returns to scale size-induced productivity differences are considered into the efficiency evaluation, with variable returns to scale the differences are neutralized by the model. The most common example of a model with variable returns to scale is an advancement to the CCR model by Banker, Charnes and Cooper, the BCC model [3]. This model includes an additional measuring variable in the fundamental equation to capture rising, falling or constant returns to scale.
In the area of software development, DEA was so far only rarely applied. Banker and Kemerer use this approach in order to prove the existence of both rising and falling returns to scale [4]. Banker and Slaughter use the DEA in the area of maintenance and enhancement projects [5]. It can be proven that rising returns to scale are present, which would have made a cost reduction of around 36 per cent possible when utilized. An investigation of Enterprise Resource Planning (ERP) projects was done by [20], using 30 SAP R/3-projects of a consulting firm for the application of the DEA. [14] gives an in-depth discussion on the application of DEA in software development.

3 Data Selection and Set

Based on the date January 8th 2007, we selected the thirty most often downloaded projects from SourceForge.net, as presented by the website based on the past 7 days. This statistic is updated daily, the current standings can be seen anytime from the respective web page[1]. This was done in order to arrive at a relatively homogeneous set of projects. Potential problems and pitfalls in using this approach are described in the following.

For each of these projects, a number of variables was retrieved from the respective homepage. We define and use the following variables in this study, with binary variables later on employed for distinguishing between groups of projects:

- Project: This simply gives the project's name.
- Donations: This binary variable shows whether the project has activated the donations feature.
- GNU-style licence: This binary variable codes whether a project is under a GNU GPL licence (true) or not (false), to give an impression of whether a strict copyleft-scheme is followed by the project.

[1]http://sourceforge.net/top/toplist.php?type=downloads_week

- Audience: Again, the intended audience of a project is coded as a binary variable, depending on whether the intended audience is developers or system administrators (true) or not (false).
- Age: The age of the project in years, which is computed based on the year in which the project was registered on SourceForge.net (using 2007 minus registration year).
- Developers: This is the number of developers as reported by the project's page on SourceForge.net.
- Downloads: This is the number of downloads of the project within the last 7 days, as given by the respective statistics page described above.
- Status: The development status from the web page. This is assigned by the project's administration, and has seven possible values, reaching from planning, pre-alpha, alpha, beta to production/stable and mature, and to inactive.
- Translations: The number of different translations available, from the project's page, with all languages counted equally as one.
- Operating Systems: As translations, but with the respective operating systems (or families thereof, e.g. all Windows versions are counted as one).
- Tracker: This binary variable codes whether the project employs the tracker service of SourceForge.net (true in that case).
- Tracker total: This is the total number of entries summed over all different active trackers of a project.
- Mailing list: This binary variable codes whether the project employs the mailing list service of SourceForge.net (true in that case).
- Mailing list total: This is the total number of postings summed over all mailing lists of a project.
- Forum: This binary variable codes whether the project employs the forum service of SourceForge.net (true in that case).
- Forum total: This is the total number of messages summed over all different active forums of a project.
- Tasks: This binary variable codes whether the project employs the task service of SourceForge.net (true in that case).
- Tasks total: This is the total number of tasks (in any status like open or closed) summed over all different subprojects of a project.
- CVS: This binary variable codes whether the project employs a CVS repository (true in that case).
- CVS commits: The total number of commits to the CVS repository as given on the project's page.
- SVN: This binary variable codes whether the project employs an SVN repository (true in that case).
- SVN commits: The total number of commits to the SVN repository as given on the project's page.
- Size: The size in byte of software offered, summing over all packages of the project in the latest release.

The first, and a major source of possible threats is construct validity. Several measures used for conceptualizing different aspects for the following analyses might be problematic and need to be discussed in this context. First is the notion of developer, which is taken directly from the web page. In some projects, people could be contributing code without relevant account, which sometimes is only granted to long-time participants, by sending it to one of those persons who then does the actual commit. Therefore, the number of developers might actually be higher than the number reported here. This fact is very problematic to check. In a case study of the OpenACS project under participation of project insiders and using the strict standards for CVS comments, [11] have found that only 1.6% of revisions pertained to code committed for someone without CVS privilege. Other metrics suffer from similar possible problems, for example a project might have existed before it was registered on SourceForge.net, and also the size might be affected by several factors like different compression algorithms employed. In addition, several parts of the coordination and communication tools might be in use, but not opened to the public, and thus disregarded in this context, or tools completely distinct from the platform might be employed. This is for example true for the source code versioning systems in our dataset. Also [13] give an overview of problems associated with mining data from Sourceforge.net. Lastly, the external validity of the results depends on the selection of an appropriate dataset. In our case, the approach is still mostly exploratory, but using the definition above, a coherent dataset was aimed at. For, example, this shows in the intended audience, which in no case is developers or system administration alone, or the fact that all projects save one use a GPL-licence. Table 1 gives descriptive statistics for some relevant metrics.

Table 1. Descriptive Statistics of Dataset (N=30)

	Median	Mean	Std.Dev.	Min.	Max.
Downloads	97,682.00	285,507.53	538,548.91	63,122	2,457,185
Developers	9.00	10.23	9.53	1	39
Status	5.00	n/a	n/a	4	6
Age	5.00	4.87	1.76	2	8
Translations	1.50	7.90	11.19	1	35
Size	21,220K	106,780K	232,072K	2,832K	998,118K
Tracker total	175.00	1,313.73	4,095.57	0	22,527
Mailing list total	29.00	10,070.13	31,792.62	0	169,574
Forum total	0.00	3,659.40	9,520.53	0	44,272
Tasks total	0.00	2.33	9.83	0	52

4 Analysis and Results

Based on the data set and variables as described above, we set up an DEA with the following parameters, using the program accompanying [8]: The first choices to be taken concern the definition of input and output factors, as well as the model to be applied. Based on the literature on DEA in the context of IT-projects [4,5,14,20], variable returns to scale are selected. Regarding the orientation of the model, an output-orientation might seem more appropriate. Given a certain input which can be acquired, i.e. participants attracted, the output is to be maximized. According to this reasoning, the BCC-O model is applied.

Regarding the definition which factors are to be used as inputs and outputs, it is to be considered that with an increase in the number of factors more DMUs are estimated to be efficient. Also the availability of factors in the data set limits the possibilities. In this case, we selected to use the number of developers and years of existence as inputs, downloads, size, status and translations as outputs. Naturally, this selection is based on the available data, and could be changed.

For an overview of the results, see Table 1. In this table, statistics on the efficiency scores in the total population are given. Overall, 11 different projects have been classified as DEA efficient, the mean efficiency score with 0.922 seems relatively high. For each efficient project, the number of times it appears in the reference sets of non-efficient projects is also given. This can be used as an indicator of the relative importance of this project in determining efficiency scores.

Table 2. Results of Applying DEA to the dataset

No. of DMUs	30
Average	0.922
Std. Dev.	0.096
Median	0.953
Minimum	0.706
Maximum	1.000
Number of DEA-efficient DMUs	11
Frequency in Reference Set	
Peer set	**Frequency**
eMule	8
Ares Galaxy	0
Azureus	3
GTK+ and The GIMP installers for Windows	10
eMule Plus	0
emule Xtreme Mod	15
Portable Apps	6
CDex	16
Gaim	2
MediaCoder	0
WinSCP	0

As one of the results is an efficiency score for each project, we can now use this score for analysing potential effects on this efficiency. As explanatory variables, information concerning the communication and coordination tools employed by the projects is used. As a start, correlations between the efficiency scores and these metrics are explored to uncover any relationships. All of the following analyses were performed using R (version 2.4.0), a free software environment for statistical computing and graphics. Specifically, tracker, mailing lists, forums, tasks and both CVS and SVN were explored as possible influences. In addition, a new metric was computed summing up the binary variables depicting whether or not a tool was employed, to give an indication of the overall diversity of a project in this context. This shows out of a maximum of 6 a mean of 2.53 with median 3 and 1.48 standard deviation.

The results are not conclusive: Regarding correlation coefficients, these are mostly small (below 0.3) and for all tools except forums negative. Also the correlation to the overall number of tools employed is with -0.257 negative. Using non-parametric Mann-Whithney U-tests, these results were tested for statistical significance: The negative relationships with overall tool usage ($p<0.01$), CVS ($p<0.05$), tasks ($p<0.01$), and the positive relationship with forum employment ($p<0.01$) are statistically proven. The results from [18] regarding licensing scheme and audience could not be checked due to minimal respectively no variance in these attributes, the inequality in contributions was not available in this data set.
These results seem rather surprising, given that [19] found a relationship between process maturity and success, but there are two different explanations: First, the tools as provided by SourceForge.net are not giving relevant help to the projects employing them, so projects using other tools, or even none at all for a given task perform better. The second explanation would be that all the tools for communicating with users and potential co-developers are more of a hindrance to efficient software development, detracting attention and time from the developers, which might be better spent on actual development work. Naturally, the view on this might also depend on the output factors included: Employing mechanisms like bug tracking might help to achieve higher quality in the released software, and it is unclear whether this effect is currently incorporated. Naturally, it could be assumed to higher quality projects achieve a higher number of downloads, but including quality aspects in the list of output factors might give additional insights.

5 Conclusion and Future Research

In this paper, we have used a method to compare the efficiency of open source projects to analyse potential impacts of different communication and coordination tools. The method used is the DEA, which is well-established in other fields and offers several advantages in this context as well: It is a non-parametric optimization method without any need for the user to define any relations between different factors or a production function, can account for economies or diseconomies of scale, and is able to deal with multi-input, multi-output systems in which the factors have different scales. Using a data set of 30 open source project retrieved from SourceForge.net, we have demonstrated the application of DEA. Results show that the efficiency of the projects is in general relatively high with low variance. Regarding the effects of tool employment on the efficiency of projects, the results were surprising: Most of the possible tools, and overall usage, showed a negative relationship to efficiency. This could be either due to more efficient tools being available elsewhere, or a negative influence of all activities except software development per se.

In future research, additional work has to be done on arriving at a common understanding of input and output factors and their definitions. For example, using the size in bytes instead of lines-of-code might be problematic, but on the other hand captures other output aspects like audio or others as well. Also the selection of projects to be included might be worked on, to preclude projects without real development work, which only serve as assemblers of others. Further, additional analyses based on the results using other project characteristics would be of high interest. For example, the definition of different process models would be of high interest for efficiency comparisons. These could also include comparisons within application areas, different project scales, and comparisons to commercial or mixed-mode development projects.

References

[1] Albrecht, A.J., & 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.

[2] Amor, J.J., Robles, G., & Gonzalez-Barahona, J.M. (2006). Effort Estimation by Characterizing Developer Activity. In *Proceedings 8th International Workshop on Economics-Driven Software Engineering Research (ICSE 2006)*, Shanghai, China.

[3] Banker, R.D., Charnes, A., & Cooper, W. (1984). Some Models for Estimating Technical and Scale Inefficiencies in Data Envelopment Analysis. *Management Science*, 30, 1078-1092.

[4] Banker, R.D., & Kemerer, C. (1989). Scale Economies in New Software Development. *IEEE Transactions on Software Engineering*, 15(10), 416-429.

[5] Banker, R.D., & Slaughter, S.A. (1997). A Field Study of Scale Economies in Software Maintenance. *Management Science*, 43(12), 1709-1725.

[6] Charnes, A., Cooper, W., & Rhodes, E. (1978a). *A Data Envelopment Analysis Approach to Evaluation of the Program Follow Through Experiments in U.S. Public School Education* (Management Science Research Report No. 432). Carnegie-Mellon University, Pittsburgh, PA.

[7] Charnes, A., Cooper, W., & Rhodes, E. (1978b). Measuring the Efficiency of Decision Making Units. *European Journal of Operational Research*, 2, 429-444.

[8] Cooper, W., Seiford, L., & Tone, K. (2000). *Data Envelopment Analysis: A Comprehensive Text with Models, Applications, References and DEA-Solver Software*, Boston, MA: Kluwer Academic Publishers.

[9] Crowston, K., Annabi, H., & Howison, J. (2003). Defining Open Source Software Project Success. In *Proceedings of ICIS 2003*, Seattle, WA.

[10] Crowston, K., Annabi, H., Howison, J., & Masango, C. (2004). Towards A Portfolio of FLOSS Project Success Measures. In *Collaboration, Conflict and Control: The 4th Workshop on Open Source Software Engineering (ICSE 2004)*, Edinburgh, Scotland.

[11] Demetriou, N., Koch, S. & Neumann, G. (2006). The Development of the OpenACS Community. In Lytras, M. & Naeve, A. (eds.) *Open Source for Knowledge and Learning Management: Strategies Beyond Tools*, Hershey, PA: Idea Group.

[12] Farell, M.J. (1957). The Measurement of Productive Efficiency. *Journal of the Royal Statistical Society*, Series A 120(3), 250-290.

[13] Howison, J. & Crowston, K. (2004). The perils and pitfalls of mining SourceForge. In *Proceedings of the International Workshop on Mining Software Repositories*, pp. 7-11, Edingburgh, Scotland, UK.

[14] Kitchenham, B. (2002). The question of scale economies in software - why cannot researchers agree? *Information & Software Technology*, 44(1), 13-24.

[15] Kitchenham, B., & Mendes, E. (2004). Software Productivity Measurement Using Multiple Size Measures. *IEEE Transactions on Software Engineering*, 30(12), 1023-1035.

[16] Koch, S. (2004). Profiling an open source project ecology and its programmers. *Electronic Markets*, 14(2), 77-88.

[17] Koch, S. (2005). *Effort Modeling and Programmer Participation in Open Source Software Projects* (Arbeitspapiere zum Tätigkeitsfeld Informationsverarbeitung, Informationswirtschaft und Prozessmanagement, Nr. 03/2005). Wirtschaftsuniversität Wien, Vienna, Austria.

[18] Koch, S. (to appear). Measuring the Efficiency of Free and Open Source Software Projects Using Data Envelopment Analysis. In Sowe, S.K., Stamelos, I. and Samoladas, I. (eds.): *Emerging Free and Open Source Software Practices*.

[19] Michlmayr, M. (2005). Software Process Maturity and the Success of Free Software Projects. In Zielinski, K. and Szmuc, T. (eds.): *Software Engineering: Evolution and Emerging Technologies*, pp. 3-14, IOS Press, Amsterdam, The Netherlands.

[20] Myrtveit, I., & Stensrud, E. (1999). Benchmarking COTS Projects Using Data Envelopment Analysis. In *Proceedings of 6th International Software-Metrics-Symposium,* pp. 269-278, Boca-Raton.

[21] Stewart, K.J. (2004). OSS Project Success: From Internal Dynamics to External Impact. In *Collaboration, Conflict and Control: The 4th*

Workshop on Open Source Software Engineering (ICSE 2004), Edinburgh, Scotland.

[22] Stewart, K.J., & Ammeter, T.A. (2002). An Exploratory Study of Factors Affecting the Popularity and Vitality of Open Source Projects. In *Proceedings of ICIS 2002*, Barcelona, Spain.

[23] Weiss, D. (2005). Measuring Success of Open Source Projects Using Web Search Engines. In *Proceedings of the 1st International Conference on Open Source Systems*, pp. 93-99, Genoa, Italy.