Brooks' Versus Linus' Law: An Empirical Test of Open Source Projects

Charles M. Schweik
Department of Natural Resources Conservation, Center for Public Policy and Administration, National Center for Digital Government
University of Massachusetts Amherst
Amherst, MA 01003, USA
cschweik@pubpol.umass.edu

Bob English
Center for Public Policy and Administration, National Center for Digital Government
University of Massachusetts Amherst
Amherst, MA 01003, USA
bobengl@gmail.com

Sandra Haire
Department of Natural Resources Conservation
University of Massachusetts Amherst
Amherst, MA 01003, USA
shaire@nrc.umass.edu

Meelis Kitsing
Department of Political Science
National Center for Digital Government
University of Massachusetts Amherst
Amherst, MA 01003, USA
mkitsing@polsci.umass.edu

Acknowledgement: Support for this study was provided by grant from the US National Science Foundation (NSFIIS 0447623).

The opinions, findings and recommendations are those of the authors and do not reflect the views of the funding agency.

INTRODUCTION

Free/Libre and Open Source Software (FOSS) are Internet-based "commons," where volunteers and paid professionals collaborate to produce software that is a public good. Over the last two decades, FOSS has had an active and growing community outside of the government sector, but in recent years, the FOSS phenomenon has been leading government IT managers to consider more fully open standards, interoperability of software and data, and the avoidance of vendor "lock in." In addition, FOSS technologies provide some other potential benefits in government settings, including the potential to share software solutions between agencies and governments, the transparency of code (which in some domains, such as in voting systems, could be quite important), the potential reuse of software modules, a reduction in software acquisition costs, and the sharing of programming staff and resources across agencies or even governments toward some common goal [1], [2].

Government agencies worldwide are both using more FOSS, and are showing interest in FOSS as a development paradigm for software products developed in their own shops or by contractors. However, one common concern over the use FOSS solutions is the longer-term support of the projects. If there is no specific firm or organization that owns the software, many wonder how can it be assured that the software will continue to be maintained and supported. This is a concern voiced by IT managers in all sectors. The central question relates to the concern of "project abandonment" and what factors may ensure that it doesn't occur. From an operational standpoint, public IT managers interested in FOSS have only two options that might help avoid project abandonment. They can contribute some of their own programming staff to a help the project move along, and to ensure in-house expertise, or, they can contribute financially to support programmers not on their staff who are working on or interested in working on the project. Either option tends to increase the number of developers on the project, thus helping to ensure its further development and maintenance.

RESEARCH PUZZLE

This brings us to the key research question for this paper. *Are FOSS projects more likely to be successful if additional programming support is added to the project*? There are two competing theories – Brooks' Law and Linus' Law – that predict opposite relationships between the number of developers on a team and project success. Brooks argued that "adding manpower to a late project makes it later" [3]. Factors that contribute to lateness include the communication complexities with new team members, coordination costs, and the training needed to bring them up to speed. This thesis is directly related to the literature on "collective action" situations, where groups of people work to create public goods. Olson argued in his groundbreaking work that "the larger the group, the less likely it will further its common interests" [4].

Deleted: .

It has been argued that Brooks' law doesn't translate directly to FOSS communities [5], [6]. In one of the most cited and influential early works on FOSS, *The Cathedral and the Bazaar*, Eric Raymond presented "Linus' Law" (named after Linus Torvalds, lead developer of the Linux operating system) by saying: "Given enough eyeballs, all bugs are shallow" [7]. In other words, with a relatively large testing and developer community, a problem in software code will be identified quickly and a solution will be produced. However, Jones [6] findings allow us to question the applicability of Linus' Law from a different perspective. In most FOSS projects, a few dedicated programmers do the majority of the code writing. The others contribute bug fixes or a "few hundred lines of code" here or there. In short, the idea of "more eyes" doesn't typically play out in FOSS. But this is a proposition that can be tested empirically. In general, if small core teams do most of the work, then projects with larger teams really shouldn't be more successful than projects with small teams.

RESEARCH METHODS

This paper empirically tests these competing theories by building on our previous work [8] and presenting a success/abandonment measure for projects using the Sourgeforge.net (SF) dataset. Two projects – one at the University of Notre Dame (http://www.nd.edu/~oss/) and the other based at Syracuse University (http://ossmole.sourceforge.net/) -- have realized the importance of capturing and keeping historical SF data. The result is a dataset of 107,747 FOSS projects that are classified as either

successful, abandoned or indeterminate. We test three competing hypotheses about the relationship of developer team size and FOSS project success:

The Linus' Law Hypothesis: FOSS projects with larger development teams will be more successful.

The Brooks' Law Hypothesis: FOSS projects with larger teams will face added coordination costs, which will hinder FOSS development. Consequently, they will be less successful.

The "Core Team" Hypothesis: The size of the development team won't have any effect on FOSS project success, because the core developer groups are almost always small teams.

In order to operationalize the concepts of successful, abandoned and indeterminate, we point out that FOSS projects go through two main longitudinal stages: "Initiation" and "Growth." We define the *Initiation Stage* as the period where a project is active but there has yet to be a first public release of the code. We define the *Growth Stage* as the period after the first public release of code.

FINDINGS

We used simple logistic regression using Developer Count as a single independent variable against our dichotomous dependent variable "Abandoned in the Growth Stage (0) and Successful in the Growth Stage (1)". In our dataset, 46,374 projects fell in these two categories. 30,592 were abandoned projects and 15,782 were successful projects. The results show that the number of developers on a project is a statistically significant in explaining variability in the data (Model L.R. $P<0.02$) and the coefficients have significant explanatory power ($P<0.02$ based on Wald Z). . The odds ratio for the Developer Count coefficient is 1.24. This indicates that for each developer added to a project, the odds that the project will become successful in the growth stage increases 1.24 times. We calculated the actual percentage of Successful in the Growth Stage projects in our dataset. All projects with greater than 86 developers have a predicted and an actual probability of 1. However, the model represents a poor fit for the data and ability to discriminate differences in response is fairly low ($C=0.638$).

In addition, we fully recognize that the simple univariate regression model does not fully explain the factors that lead FOSS projects to success or abandonment. Obviously, a multivariate model that includes other theoretically-driven covariates is needed to improve the model's goodness of fit. But the point for this paper is that even this simple model does provide evidence for choosing one hypothesis over another. Our findings suggest that adding more programmers improves the chances that a FOSS project will be successful, consistent with Linus' Law.

Our finding demonstrate a correlation, not necessarily a causal relationship.Do more developers lead to project success? Or is it that successful projects attract more developers, in part because of the economic motivations that drive some programmers to participate (e.g., signaling programming skill to a broad community, self-learning through reading others' programs and peer-review) that the rise in success rate holds strongly for projects with smaller numbers of developers tends to favor the idea that more developers produces success. We can't answer this question with the dataset we have, because it represents generally one point in time (August-October, 2006). But even with the above limitations acknowledged, the findings above lend strong support to the argument that the relatively flat, modular system of coordination in FOSS projects allows the addition of programmers without too much impact regarding coordination costs. So the real concern in FOSS appears to be not the issue of slowing projects down with the addition of more programmers, but rather, given the large number of small developer team projects, getting "more eyeballs" participating in the projects.

CONCLUSION AND IMPLICATIONS

We see at least two implications for public sector organizations. First, from a FOSS user perspective, it may be worth checking the project website to see how many developers are associated with the project before choosing a particular product. If there is a relatively large number of developers, that's a good sign. Of course, that shouldn't be the sole criteria– there will still be lots of successful small projects. Nevertheless, development team size should be one of many selection criteria. Second, the research here suggests that adding programmers to help existing FOSS projects is beneficial to these projects. It also suggests that in situations where a new project is being initiated using public sector programming or contractual staff, it might be worthwhile to actively look for other development partners within the agency or even externally (other agencies or governments) who might have programming staff to contribute to the project, as long as some consideration is made related to the architecture (e.g., modularity) of the project. Finally, the findings here may have broader implications outside of software in how public organizations may wish to organize work in the future. Recently, we are seeing the emergence of "open source like" collaborations in other public sector work. While more research is needed, our findings suggest that flatter hierarchies and modularity create some advantages in the way some work can be accomplished.

REFERENCES

[1] Ghosh, R.A., Glott, R., Gerloff, K.,Schmitz, P-E., Aisola, K., and Boujraf, A.2007. "Study on the Effect on the Development of the Information Society of European Public Bodies Making Their Own Software Available as Open Source: Final Report." http://www.publicsectoross.info/images/resources/15_154_file.pdf.

[2] Wong, K. 2004. Free/Open Source Software:Government Policy. United Nations Development Programme-Asia Pacific Development Information Programe (UNDP-APDIP).

[3] Brooks, F. P. Jr. [1975] 1995. *The Mythical Man-MonthEssays on Software Engineering*. Anniversary Edition. Reading, MA: Addison-Wesley.p. 25

[4] Olson, M. 1965. *The Logic of Collective Action: Public Goods and the Theory of Groups*. Cambridge, MA: Harvard University Press.p. 35

[5] Abdel-Hamid, T.K., and Madnick, S.E. 1990. "The Elusive Silver Lining: How We Fail to Learn from Software Development Failures." Sloan Management Review 32: 1.pp.39-48.

[6] Jones, P. 2000. Brooks' law and Open Source: The more the merrier? IBM, 4(10)http://www-106.ibm.com/developerworks/library/merrier.html.

[7] Raymond, E. 2000. "The Cathedral and the Bazaar. http://www.catb.org/~esr/writings/cathedralbazaar/cathedralbazaar/ar01s05.html.

[8] English, R. and Schweik, C.M. 2007. "Identifying Sucess andTragedy of Free/Libre and Open Source (FLOSS) Commons: A Preliminary Classification of Sourceforge.net projects." Proceedings of the 29th International Conference on Software Engineering + Workshops (ICSE-ICSE Workshops 2007), 20-26 May, Minneapolis, Minnesota.