Software Quality Journal **Vol. 14 No. 4, 2006**

Editorial: Free/Open Source Software, Silver Bullets, and Mythical Months.

The notion of free/open source software (F/OSS) development is intriguing. In theory, such software is developed by volunteers who see the software as fulfilling their own needs. Developers work on what they want, anyone can have a copy of the source code and contribute towards improving the system by reporting and/or fixing bugs, and developing needed new functionality. The code is high quality because of the large number of people involved in reporting and fixing errors, and adding functionality. The F/OSS process should satisfy user and developer needs.

In contrast, the list of software disasters produced through "traditional" commercial development is staggering. You can easily find a long list of (the most frightening) software failures by searching for "software disasters" using your favorite search engine. Another indicator of problems with traditional development is the high proportion of software development projects that are cancelled. Virtually all commercial software developers with five or more years of experience have worked one or many projects that have been cancelled. Estimates in software engineering textbooks, research papers, and from a web search for "cancelled software projects" indicate that from 33-44% of commercial software projects end up cancelled. I also find estimates that up to 67% of commercial software projects experience large cost overruns.

Do F/OSS projects get cancelled and experience cost overruns? Are such questions at all relevant to F/OSS projects?

F/OSS projects do not get cancelled by the "suits" or "bean counters" when the corporate mission changes, or the company reorganizes, or for some other arbitrary reason. However, F/OSS projects can wither and die in part because few developers are attracted to the project. For example, Crowston et al. found that 67% of nearly 100,000 projects listed on SourceForge attracted no more than one developer at any point in a five year period [1]. Rather than being cancelled explicitly, F/OSS projects without a "market" fade into inactivity.

However, as long as a core set of developers remain committed to an F/OSS project, the project will continue. Thus, F/OSS projects have one advantage over commercial projects: no corporate official can arbitrarily cancel the project. In spite of this advantage, if we consider projects that do not attract more than one developer to be failures, then the failure rate of 67% for F/OSS projects is much worse than the 33% cancellation rate for commercial projects. Thus, even though an "outsider" cannot arbitrarily cancel a project, the vast majority of F/OSS projects do not attract much interest.

In spite of the "failure" rate for F/OSS projects, many projects have been wildly successful. But is that really different from proprietary projects? Generally there are just a few winners in an application domain, leaving the rest of the contenders in the dust.

F/OSS projects, at least projects without commercial backers, offer one clear advantage over commercial development. They do not exhibit cost overruns, as they do not tend to have budgets or cost estimates. However, there are potential cost overruns for users of F/OSS, depending on the difficulty of integration.

As, Porter et al. point out [2], the F/OSS projects effectively add developers (really testers) to projects, which somewhat contradicts Brook's Law [3]:

"Adding manpower to a late software project makes it later."

The added developers focus on testing, and a larger group of testers will (probably) find more program faults.

In summary, F/OSS is not the long sought "silver bullet" [4]. F/OSS projects commonly fail, perhaps for different reasons than proprietary projects. However, they do not seem to suffer like proprietary software projects from cost overruns, or from adding developers "late" in the project. One question that I have not addressed here is the following: How does the quality of F/OSS products compare with that of proprietary products? I will save that question for a future column.

References

[1] K. Crowston, J. Howison, and H. Annabi. "Information Systems Success in Free and Open Source Software Development: Theory and Measures". *Software Process Improvement and Practice*. 11:123-148, 2006.

[2] A. Porter, C. Yilmaz, A. Memon, A. Krishna, D.C. Schmidt, and A. Gokhale. "Techniques and Processes for Improving the Quality and Performance of Open-Source Software. *Software Process Improvement and Practice*. 11:163-176, 2006.

[3] F. Brooks. "No Silver Bullet – Essence and Accidents of Software Engineering", *Proc. Information Processing 86*. pp. 1069-1076, 1986.

[4] F. Brooks. *The Mythical Man-Month (Anniversary Edition*). Addison-Wesley, 1995.

James Bieman
Fort Collins, Colorado
U.S.A