A Critical Look at Open Source

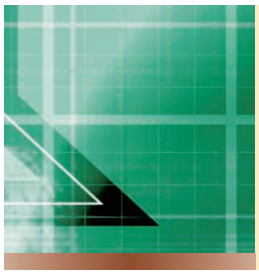
Brian Fitzgerald, University of Limerick

Despite 50 years of research and practice, the development of software—particularly complex and innovative software—is far from a predictable, exact process. Many software projects exceed their budget, fail to conform to their development schedule, and do not meet expectations when eventually delivered.

In recent years, open source software—more properly, free and open source software—has emerged as one popular solution to the so-called "software crisis." Advocates regard F/OSS as an agile, practice-led initiative that addresses three key issues:

- *Cost*: F/OSS products are usually freely available for public download.
- *Time scale*: The collaborative, parallel efforts of globally distributed developers allow many F/OSS products to be developed much more quickly than conventional software.
- *Quality*: Many F/OSS offerings are recognized for their high standards of reliability, efficiency, and robustness; products such as GNU/Linux, Apache, and Bind have become "category killers" stifling the incentive to develop any competing products.

Supporters also point out that F/OSS harnesses the scarcest resource of all: talented software developers, many of whom exhibit a long-standing commitment to their chosen projects. In addition, the resulting peer review process helps ensure the quality of the software produced.



Despite many challenges, free and open source software remains a viable model.

Moreover, researchers in other fields—including sociology, economics, management, psychology, public policy, and the law—have embraced F/OSS as an innovative model for organizing activity in a variety of social contexts.

F/OSS CHALLENGES

Despite its wide appeal, F/OSS faces a number of serious challenges that have led some naysayers within the software engineering community to declare that the model is doomed to ultimately fail.

Development talent

The main F/OSS contributors are "code gods" acknowledged to be among the world's most talented and highly motivated programmers. The software industry has long recognized the potential contribution of gifted individuals, but they are critical to F/OSS: The absence of face-to-face interaction or other normal organizational sanctions makes it vital that there be an ultimate arbiter with unquestioned authority and ability who charismatically inspires others and can resolve disputes and prevent forking.

However, the widespread interest in F/OSS may exceed the development talent available. To some extent, this is already happening. For example, SourceXchange, a brokering service for companies soliciting F/OSS developers, ceased operations in 2001. Studies of Freshmeat.net and SourceForge.net, two popular F/OSS development Web sites, revealed that most projects have only one or two developers. They also have little apparent vitality, as follow-up studies reported no change in version number or size of code base for many listed projects several months later.

Code quality

While F/OSS pioneers may have been "best-of-breed" programmers, the ability of future contributors is much more in doubt. The popularity of F/OSS increases the risk that so-called net-negative-producing programmers will become involved. Despite the long probationary period programmers serve on many F/OSS projects, NNPs could introduce errors into the code base. This could irremediably harm F/OSS projects, which lack the marketing muscle to help undo damage to volunteers' reputations. Some recent empirical studies question the axiom that all F/OSS products are of high quality.

Project initiation

Even if more companies decide to make their source code generally available, F/OSS developers are not privy to

the organizational design decisions underlying the code base's evolution. Simply releasing a large proprietary product's source code—as Netscape did with its Mozilla browser—is not alone enough to launch a vibrant, fully functioning F/OSS project.

Code modularity

Code modularity is a key to F/OSS success as it is the basis for distributing the development process. However, it can also be a project's Achilles' heel. Excessive modularity increases the risk of *common coupling*, an insidious problem in which modules unnecessarily refer to variables and structures in other modules. Thus, changes to data structures and variables in seemingly unrelated modules can have major knock-on implications. In this way, as F/OSS systems evolve, maintaining them can become difficult if not impossible in the long run.

Mundane tasks

Many software tasks—such as documentation, testing, internationalization/localization, and field support—are tedious but vital, particularly as new cohorts of developers join and maintain projects. However, F/OSS developers could cherry-pick more exciting tasks such as code design, which is understandable in a culture that does not worship the likes of "documentation gods." Unfortunately, nontechnical F/OSS contributors and users may not fill the gap to the extent originally predicted.

Stability

Companies have valid concerns about the survival of and continued support for F/OSS products. The traditional telephone hotline and maintenance contract offer a comfort factor that a voluntary bulletin board—which is the main support for many F/OSS products—cannot provide.

Standardization

GNU/Linux version proliferation already poses a substantial challenge to software providers, who must write and test applications for the many commercial versions available. Also, the independent development of F/OSS products results in time-consuming interoperability and compatibility problems among different product versions.

Standardization is arguably more important for F/OSS developers, who typically do not meet face to face, than for traditional proprietary developers. Any mechanism that can facilitate collective action, such as common standards for integration, would benefit the F/OSS community. Although numerous initiatives are moving broadly in this direction, their conflicting political agendas do not make the prospects of any near-term convergence on standards likely.

WHY F/OSS?

Despite these potential choke points, the future for F/OSS remains bright. Recent research effectively dispels the stereotype of F/OSS developers as anarchistic hackers operating on the fringes of society. Studies by Karim Lakhani and Bob Wolf in the United States and by Rishab Aiyer Ghosh in Europe indicate that most developers are in stable cohabiting relationships, often with children, and are experienced IT professionals paid for their work on F/OSS projects.

Individuals participate in and support F/OSS development for a number of reasons. Developers can collaborate with peers they truly respect, acquiring more rapid feedback and direct recognition through credit acknowledgments on the project lists. In addition, the informal development style of F/OSS liberates many from the formalized organizational environments in which they work every day. Also, students can boost their career prospects by gaining experience on real development projects.

Finally, research suggests that implementing F/OSS can result in significant savings. Organizations that deploy F/OSS products freely offer advice to one another, sharing insights and lessons learned. At the same time, hands-on users must proactively work with IT staff to ascertain whether available F/OSS products meet their needs. This cooperative spirit, which in part may arise from a sense of shared adventure, starkly contrasts with the tradi-

Further Reading

The following resources provide more information on the issues surrounding free and open source software:

- A. Capiluppi, P. Lago, and M. Morisio, "Evidences in the Evolution of OS Projects through Changelog Analyses," *Taking Stock of the Bazaar: 3rd Workshop Open Source Software Eng.*, 25th Int'l Conf. Software Eng., 2003, pp. 19-24; http://opensource.ucc.ie/icse2003.
- B. Fitzgerald and T. Kenny, "Developing an Information Systems Infrastructure with Open Source Software," *IEEE Software*, vol. 21, no. 1, 2004, pp. 50-55.
- E.S. Raymond, *The Cathedral & the Bazaar: Musings on Linux and Open Source by an Accidental Revolutionary*, O'Reilly & Assoc., 1999.
- S. Rusovan, M. Lawford, and D. Parnas, "Open Source Software Development: Future or Fad?" to be published in J. Feller et al., eds., *Perspectives on Free and Open Source Software*, MIT Press, 2005.
- S.R. Schach and A.J. Offutt, "On the Nonmaintainability of Open-Source Software," *Meeting Challenges and Surviving Success: 2nd Workshop Open Source Software Engineering*, 24th Int'l Conf. Software Eng., 2002; http://opensource.ucc.ie/icse2002/SchachOffutt.pdf.

tional impersonal, bilateral relationship between vendor and customer.

Some argue that F/OSS represents a paradigm shift in software engineering. For example, Eric Raymond describes the disciplined and coordinated conventional approach as a "cathedral" and F/OSS as a noisy, confused "bazaar."

At first glance, F/OSS appears to contravene fundamental tenets of software engineering: no formal requirements specification or design process, no risk assessment or measurable goals, informal coordination and control, and much duplication and parallel effort. However, well-established software engineering principles lie at the heart of F/OSS.

F/OSS does not represent a paradigm shift in software engineering.

Code modularity allows partitioning work among a global pool of developers and facilitates the recruitment of new contributors, as it reduces their learning curve to a subset of modules rather than the entire project. Indeed, developers rewrote projects such as Sendmail, Samba, and even Linux in modular form to ensure successful development.

F/OSS developers also carefully vet and incorporate contributions in accordance with sound configuration management, independent peer review, and testing—all familiar software engineering concepts. ■

Brian Fitzgerald *holds the Frederick A. Krehbiel II Chair in Innovation in Global Business and Technology at the University of Limerick, Ireland. Contact him at bf@ul.ie.*

Editor: Richard G. Mathieu, Dept. of Decision Sciences and MIS, St. Louis University, St. Louis, MO 63108; mathieur@slu.edu