

Is Open Source Software Development *Faster, Better, and Cheaper* than Software Engineering?

Walt Scacchi
Institute for Software Research
University of California, Irvine
Irvine, CA 92697-3425 USA
+1-949-824-4130, +1-949-824-1715
Wscacchi@uci.edu

ABSTRACT

In this position paper, I draw attention to the question of determining whether open source software development may represent a significant alternative to modern software engineering techniques for developing large-scale software systems. OSSD often entails shorter time frames, producing higher quality systems, and incurring lower costs than may be realized through developing systems according to SE techniques. Understanding why and how this may arise is the focus of this paper.

Keywords

Software Engineering, Open Source Software Development

1. Introduction

The likelihood and circumstances in which open source represents a more effective and efficient approach than software engineering merits serious review. Such conditions may point to the need to critically reflect on how the practice and principles of software engineering needs a serious rethinking and possible reformulation that addresses and accommodates OSSD, as well as how it differs from current SE principles.

If it is true that OSSD is faster, better, and cheaper than SE, then why bother with SE? Does OSSD address and resolve the "software crisis" that gave rise to SE? Has OSSD demonstrated the practical value and success of informal approaches, compared to the formal notation-based approaches widely advocated by SE scholars? Is "humanated" OSSD more productive than automated SE? Answering these questions cannot be ignored or slighted by reference to more than three decades of academic and industrial SE research. This position paper seeks to bring questions like these into the foreground so as to advocate the position that the SE community needs to recognize how, and under what conditions, OSSD may represent a faster, better, and cheaper alternative for how to engineer complex software systems. Failure of the SE community to embrace OSSD as something different, than current SE principles, may relegate the future of SE research to evermore of an academic curiosity, rather than as an engineering discipline whose capabilities are maximized when operationalized as a complex web of socio-technical processes and practices.

2. How is OSSD *faster* than SE?

OSSD projects like those at Tigris.org enact "internet time" development practices, much like Microsoft, Netscape, and

others [2,8]. Internet time efforts emphasize minimizing time to market, instead of complete well-engineered functionality. OSSD projects rely on software *informalisms* [12] as information representations that serve as resources that can be browsed, crosslinked, and updated on demand. These informalisms are socially lightweight mechanisms for managing, communicating, and coordinating globally dispersed knowledge about who did what, why, and how. These informalisms are easy to learn and use as semi-structured representations that capture software requirements, system design, and design rationale. As OSS developers are themselves end-users of their systems, then software requirements and design take less time to articulate and negotiate, compared to SE projects that must elicit requirements and validate system design

3. How is OSSD *better* than SE?

OSSD projects are iteratively developed, incrementally released, and peer reviewed in an ongoing agile manner [cf. 1]. These methods ensure acceptable levels of quality and coherency of system-wide software via continuous distributed testing and profiling [13]. OSSD efforts are hosted within decentralized communities [7, 11, 12, 14] that is interconnected via Web sites and repositories. Community oriented OSSD also gives rise to new kinds of requirements for community building, community software, and community information sharing systems (Web site and interlinked communication channels for email, forums, and chat). In contrast, most SE projects are targeted for hosting within a centralized corporate setting, where access and visibility may be restricted to local participants. OSSD standards [6] that reinforce best practices are apparently easier to access and follow due to their Web-based deployment, and a long history of community oriented participation in developing implementation oriented standards in an open source manner, compared to the institutionally oriented processes used to develop SE standards.

4. How is OSSD *cheaper* than SE?

OSSD tools are inexpensive/free, comparatively easy to use and learn. These tools are both given and received as public goods or gifts. Faster and better OSSD conditions in turn tend to drive down the cost of developing software, at least in terms of schedule and budget resources. Most OSSD projects are voluntarily staffed who want to work on the project, who will potentially commit their own time and effort, and who find personal and professional benefit from

the OSSD development efforts. Minimal management or governance forms [5, 14] are used to direct OSSD efforts, compared to the more rigidly hierarchical, managed, planned, staffed, controlled, and budgeted project activities typical for SE efforts.

5. How to make SE faster, better, and cheaper via OSSD processes and practices

OSSD projects enact teamwork structures and decentralized community forms [7, 11, 12, 14] that reduce/supplant functional organizational forms inherent in traditional SE techniques that increase bureaucratic tendencies. OSSD avoids reliance on formal project management techniques and administrative structures that pervade industrial SE projects. OSSD is generally community oriented and agile [1], rather than customer oriented and formal [12]. Developers as users mitigate need to spend resources trying to figure out what users want, and whether what is developed and delivered meets user needs [12]. Thus, the opportunity exists for developing new SE processes, practices, and community forms that are decentralized, peer-oriented, and rely on semi-structured, informal representations of software artifacts. SE community Web sites and community development tools also appear to be candidates for adoption.

6. Conclusions

OSSD appears to be changing the world of software development at a faster, better, and cheaper pace, and with a broader impact and audience, than SE has achieved. Understanding why this is so may be key to advancing the state of the art of both SE and OSSD. Failing to recognize the differences between the two may result in OSSD characterizing more of the leading edge of software system development, while SE characterizes more of the trailing edge. Where do you want to be?

7. Acknowledgements

The research described in this report is supported by grants from the National Science Foundation #IIS-0083075, ITR-#0205679, ITR #0205724, and IIS-#0350754. No endorsement implied. Mark Ackerman at University of Michigan, Ann-Arbor; Les Gasser, UIUC; John Noll, Santa Clara University; Margaret Elliott and Chris Jensen at the UCI Institute for Software Research, are collaborators on this research.

8. References

1. B. Boehm, Get Ready for Agile Methods, with Care, *Computer*, 35(1), 64-69, Jan. 2002.

2. M. Cusumano and D.B. Yoffie, Software Development on Internet Time, *Computer*, 32(10), 60-69, October 1999.
3. C. DiBona, S. Ockman and M. Stone, *Open Sources: Voices from the Open Source Revolution*, O'Reilly Press, 1999.
4. J. Feller and B. Fitzgerald, *Understanding Open Source Software Development*, Addison-Wesley, NY, 2002.
5. R.T. Fielding. Shared Leadership in the Apache Project. *Communications ACM*, 42(4), 42-43, 1999.
6. C. Freericks, Open Source Standards on Software Process: A Practical Approach, *IEEE Comm. Mag.*, 116-123, April 2001.
7. B. Kogut and A. Metiu, Open Source Software Development and Distributed Innovation, *Oxford Review of Economic Policy*, 17(2), 248-264, 2001.
8. A. MacCormack, R. Verganti, and M. Iansiti, Developing Products on Internet Time: The Anatomy of a Flexible Development Process, *Mgmt. Science*, 47(1), January 2001.
9. A. Mockus, R.T. Fielding, and J. Herbsleb, A Case Study of Open Source Software Development: The Apache Server, *Proc. 22nd. Intern. Conf. Soft. Eng.*, 263-272, 2000.
10. R. Pavlicek, *Embracing Insanity: Open Source Software Development*, SAMS Publishing, Indianapolis, IN, 2000.
11. W. Scacchi, Software Development Practices in Open Source Development Communities, *1st. Workshop on Open Source Software Engineering*, Toronto, Ontario, May 2001.
12. W. Scacchi, Understanding the Requirements for Developing Open Source Software Systems, *IEE Proceedings - Software*, 149(1), 25-39, 2002.
13. D. Schmidt and A. Porter, Leveraging Open-Source Communities to Improve the Quality & Performance of Open-Source Software, *1st. Workshop on Open Source Software Engineering*, Toronto, Ontario, May 2001.
14. S. Sharman, V. Sugurmaran, and B. Rajagopalan, A Framework for Creating Hybrid-Open Source Software Communities, *Info. Systems J.*, 12(1), 7-25, 2002.