# The Future of Research in Free/Open Source Software Development

Walt Scacchi
Institute for Software Research
University of California, Irvine
Irvine, CA 92697-3455 USA
+1-949-824-4130
wscacchi@ics.uci.edu

## ABSTRACT

Free/Open Source Software (FOSS) development is not the same an Software Engineering (SE). Why this is so is unclear and open to various interpretations. Both address the challenges of developing large software systems, but the development processes, work practices, and project forms differ significantly and in interesting ways according to recent empirical studies. This paper reports on highlights from a workshop held in early 2010 on the future of research in FOSS, and how such research relates to or informs our understanding of FOSS and SE, collaborative software development work, software evolution, and new software ecosystems. FOSS and SE are complementary in many ways, yet different in others, so understanding these complements and differences can help advance the future of research in both FOSS and SE. Some of these complements and differences are identified in this paper.

## Categories and Subject Descriptors

D.2 [Software Engineering]: *software management, design methodology*

## General Terms

Human Factors.

## Keywords

Open source software, software evolution, software ecosystems

## 1. INTRODUCTION

Even though Free/Open Source Software (FOSS) is widely used, much of the Computer Science research community has yet to fully recognize its potential to change the world of research and development of software-intensive systems across disciplines. As little as four years ago, FOSS was still somewhat marginal to SE, appearing in one case as simply another concurrent development methodology, rather than an a new approach to software development [1]. However, tens of thousands of FOSS projects are up and running world-wide, and millions of end-users of computing increasingly rely on FOSS-based systems. Growing numbers of research projects in physical, social, and human sciences, as well as the cultural arts are now routinely expecting to develop or use FOSS-based systems to best meet their needs. Similarly, growing numbers of businesses and government organizations are now looking to develop and use mission-critical software applications that are built with FOSS components. Why and to what ends?

In February 2010, The Computing Community Consortium sponsored a three day workshop to develop an agenda for future research in free/open source software development (FOSSD), in order to better understand the interest, diversification, and widespread growth of FOSS systems, projects and related practices [2]. The goal of this workshop was to engage researchers from academia and industry in the U.S. to help develop a broad perspective as to emerging areas for potential research investment over the next 5-10 years [13]. The Final Report from this workshop [12] elaborates four core research areas for study, as well as the research infrastructure and data needed to support empirical studies of the artifacts, work practices, development processes, project and community dynamics that characterize FOSSD. The four areas cover FOSSD and Software Engineering (SE), collaborative development work, software evolution, and new software ecosystems. Though it is not possible to cover all these topics here in appropriate detail, I instead provide a snippet of research results and opportunities for future research at the intersection of FOSSD and SE that are part of the working set of outcomes and research agenda items that were addressed in the workshop and Final Report.

## 2. SAMPLE RESEARCH RESULTS ON FOSSD

FOSSD is certainly not a panacea for developing complex software systems, nor is it simply SE, or SE done poorly. Instead, it represents an alternative community-intensive socio-technical approach to develop software systems, artifacts, and social relationships. However, it is not without its limitations and constraints.

First, in terms of participating, joining, and contributing to FOSSD projects, an individual developer's interest, motivation, and commitment to a project and its contributors is dynamic and not indefinite [8]. Some form of reciprocity and self-serving or intrinsic motivation seems necessary to sustain participation, whereas a perception of exploitation by others can quickly dissolve a participant's commitment to further contribute, or worse to dissuade other participants to abandon an open source project that has gone astray.

Second, in terms of cooperation, coordination, and control, FOSSD projects do not escape conflicts in technical decision-making, or in choices of who gets to work on what, or who gets to modify and update what. As FOSSD projects generally lack traditional project managers, then they must become self-reliant in their ability to mitigate and resolve outstanding conflicts and disagreements. Beliefs and values that shape system design choices, as well as choices over which software tools to use, and which software artifacts to produce or use, are determined through negotiation rather than administrative assignment. Negotiation and conflict management then become part of the cost that FOSS developers must bear in order for them to have their beliefs and values fulfilled. It is also part of the cost they bear in convincing and negotiating with others often through electronic communications to adopt their beliefs and values. Time, effort, and attention spent in negotiation and conflict management represent an investment in building and sustaining a negotiated socio-technical network of dependencies.

Third, in terms of forming alliances and building community through participation, artifacts, and tools points to a growing dependence on other FOSSD projects. The emergence of non-profit foundations that were established to protect the property rights of large multi-component FOSSD projects create a demand to sustain and protect such foundations. If such a foundation becomes too bureaucratic, then this may drive contributors away from its FOSSD projects. So, these foundations need to stay lean, and not become a source of occupational careers, in order to survive and evolve. Similarly, as FOSSD projects give rise to new types of requirements for community building, community software, and community information sharing systems, these requirements need to be addressed and managed by FOSSD project contributors in roles above and beyond those involved in enhancing the source code of a FOSSD project. FOSSD alliances and communities depend on a rich and growing web of socio-technical relations. Thus, if such a web begins to come apart, or if the new requirements cannot be embraced and satisfied, then the FOSSD project community and its alliances will begin to come apart.

Fourth, in terms of the co-evolution of FOSS systems and community, individual and shared resources of people's time, effort, attention, skill, sentiment (beliefs and values), and computing resources are part of the socio-technical web of FOSSD. Reinventing existing software systems as FOSS coincides with the emergence or reinvention of a community who seeks to make such system reinvention occur. FOSS systems are common pool resources that require collective action for their development, mobilization, use, and evolution. Without the collective action of the FOSSD project community, the common pool will dry up, and without the common pool, the community begins to fragment and disappear, perhaps to search for another pool elsewhere.

Last, empirical studies of FOSSD are expanding the scope of what we can observer, discover, analyze, or learn about how large software systems can be or have been developed. In addition to traditional methods used to investigate FOSSD like reflective practice, industry polls, survey research, and ethnographic studies, comparatively new techniques for mining software repositories [7] and multi-modal modeling and analysis of the socio-technical processes and networks found in sustained FOSSD projects [9, 15] show that the empirical study of FOSSD is growing and expanding. This in turn will contribute to and help advance the empirical science in fields like SE, which previously were limited by restricted access to data characterizing large, proprietary software development projects. Thus, the future of empirical studies of software development practices, processes, and projects will increasingly be cast as studies of FOSSD efforts.

## 3. RESEARCH OPPORTUNITIES FOR FOSSD AND SE

There are a significant number of opportunities and challenges that arise when we look to identifying which software development or socio-technical interaction practices found in studies of FOSSD projects might be applied in the world of SE. As such, let us consider some research opportunities for SE that can arise from FOSSD studies.

First, FOSSD poses the opportunity to favorably alter the costs and constraints of accessing, analyzing, and sharing software process and product data, metrics, and data collection instruments. *FOSSD is poised to fundamentally alter the cost and calculus of empirical SE* [3, 6, 10]. Software process discovery, modeling, and simulation research [e.g., 15] is one arena that can take advantage of such a historically new opportunity. Similarly, the ability to extract or data mine software product content (source code, development artifacts, team communications, public user feedback) within or across FOSSD project repositories [7] to support its visualization, refactoring, or redesign can be a high-yield, high impact area for SE study and experimentation. Another would be examining the effectiveness and efficiency of traditional face-to-face-to-artifact SE approaches or processes for software inspections [e.g., 16] compared to the online informal peer reviews involving "many eyeballs" prevalent in FOSSD efforts.

Second, based on results from studies of *motivation, participation, role migration, and turnover of individual FOSS developers*, it appears that the SE community would benefit from empirical studies that examine similar conditions and circumstances in conventional software development enterprises. Current SE textbooks and development processes seem to assume that individual developers have technical roles and motivations driven by financial compensation, technical challenge, and the quality assuring rigor that purportedly follows from the use of formal notations and analytical schemes. Said simply, is FOSSD more fun, more interesting, more convivial, and more personally rewarding than SE, and if so, what can be done differently to make SE more like FOSSD?

Third, based on results from studies of *resources and capabilities employed to support FOSSD projects*, it appears that conventional software cost estimation or accounting techniques (e.g., "total cost of operation" or TCO) are limited to analyzing resources or capabilities that are easily quantified or monetized. This in turns suggests that many social and organizational resources/capabilities are slighted or ignored by such techniques, thus producing results that miscalculate the diversity of resources and capabilities that affect the ongoing/total costs of software development projects, whether FOSS or SE based.

Fourth, based on results from studies of *cooperation, coordination, and control practices in FOSSD projects*, it appears that virtual project management and meritocratic socio-technical role migration/advancement can provide a slimmer and lighter weight approach to SE project management. However, it is unclear whether we will see corporate experiments in SE that choose to eschew traditional project management and administrative control regimes in favor of enabling software developers the freedom of choice and expression that may be necessary to help provide the intrinsic motivation to self-organize and self-manage their SE project work.

Fifth, based on results of studies on *alliance formation, inter-project social networking, community development, and multi-project software ecosystems*, it appears that SE projects currently operate at a disadvantage compared to FOSSD projects. In SE projects, it is commonly assumed that developers and end-users are distinct communities, and that software evolution is governed by market imperatives, the need to extract maximum marginal gains (profit), and resource-limited software maintenance effort. SE efforts are setup to produce systems whose growth and evolution is limited, rather than capable of sustaining exponential growth of co-evolving software functional capability and developer-user community seen in successful FOSSD projects [10].

Last, based on studies of *FOSS as a social movement*, it appears that there is an opportunity and challenge for encouraging the emergence of a social movement that combines the best practices of FOSSD and SE. The world of open source software engineering (OSSE) is the likely locus of collective action that might enable such a movement to arise. For example, the community Web portal for Tigris.org is focused on cultivating and nurturing the emerging OSSE community. More than 700 OSSE projects are currently affiliated with this portal and community. It might therefore prove fruitful to closely examine different samples of OSSE projects at Tigris.org to see which SE tools, techniques, and concepts are being employed, and to what ends, in different FOSSD projects.

## 4. BROADER IMPACT AREAS FOR FOSS RESEARCH AND DEVELOPMENT

Two major categories of broader impact arising from research in FOSS systems over the next 5-10 years. These are (a) software development, and (b) science and industry. Each of these broader impacts categories can be described in turn.

### 4.1 Software Development

The development of reliable large, very-large, or ultra-large scale software-intensive systems requires more than robust, formalized, and mathematically grounded approaches to SE. They also require the engagement of decentralized communities of practitioners who can participate in and contribute to the ongoing development, use, and evolution of software system tools, online artifacts, and other information infrastructure resources, either on a local or global basis. The development of software-intensive systems at large-scale and beyond needs to be recognized as something now essential to the advancement of science, technology, industry, government, and society across geographic borders and cultural boundaries.

FOSS systems research is likely to change how SE research and practice are now accomplished. The openness of FOSS system development means that new participants are coming into the world of software systems to contribute to the ongoing development and evolution of such systems. The engagement and contribution of FOSS system development participants who are not necessarily skilled in the traditional principles and practices of SE means there will be a long-term need to adapt SE concepts, techniques and tools to people lacking skills in SE, while also seeking new ways and means for motivating these new participants to engage in learning and practicing emerging SE processes, practices, and principles. In addition, the public availability of FOSSD artifacts will likely become a primarily source of data for empirical SE research, as such data will often be far less encumbered by corporate non-disclosure agreements that have historically limited what software development data can be made available for scientific research purposes.

FOSS systems research will continue to be a rich source of observation and experimentation for collaborative software development processes, practices, and project forms. As many successful, ongoing, and large-scale FOSS systems and project communities are typically physically decentralized but logically centralized, this ways and means for such sustained software development must rely on collaboration tools, techniques, and patterns of use that whose fundamental principles we do yet fully understand. Yet, FOSS system development is a clear, recurring demonstration that the development of complex systems can be performed, governed, and sustained in a decentralized manner, with little/no corporate oversight or enterprise governance, while corporate oversight and governance regimes have long been a hallmark for the development and maintenance large complex systems. Collaborative FOSS system development processes, practices, project forms, project infrastructures, and surrounding ecosystem represent new ways and means for developing complex systems that help meet societal needs.

FOSS systems depend on and co-evolve with their surrounding ecosystem. FOSS systems are both social and technological endeavors, where socio-technical interactions are more critical to system development, use, and evolution than a formal mathematical basis for specifying the system's analytical intent. Yet understanding how FOSS system ecosystems operate is at a very early stage of study, and how best to study, explain, and rationalize it is still in a very formative stage. But human-made complex systems are increasingly recognized as being products of their own complex ecosystems, and of the networks of producer, integrators, and consumers who create, assemble, and use such systems. Thus, research into complex system ecosystems like those that situate and embed FOSS systems are within the grasp of scientific study, comprehension, and explanation. These eventual accomplishments would provide the basis for rationalizing, predicting, controlling, and transferring such knowledge to other complex ecosystems, especially those that are mediated by information infrastructures or cyberinfrastructure. Thus research into FOSS ecosystems is critical to advancing scientific knowledge and technology development in many areas beyond software systems.

Last, FOSS systems are complex software systems with an open evolutionary history and future. Such openness is in many ways historically unprecedented for complex technical systems. So we should not miss or scuttle such a rare opportunity to study FOSS system and ecosystem evolution, as a software system, as a decentralized social system for peer production, and as a complex socio-technical system.

## 4.2 Science and Industry

Many grand challenges for science and engineering going forward depend on the research and development of a new generation of complex, software-intensive systems (cf. http://www.engineeringchallenges.org/, accessed September 2010). Advanced healthcare informatics, advanced personalized learning systems, secure cyberspace, engineering automated tools for scientific discovery, and enhanced virtual reality are all readily recognized as problem domains that depend on future software systems. Making solar energy economical, managing the nitrogen cycle, preventing nuclear terror, provide energy from fusion, providing access to clean water, engineering better medicines, developing carbon sequestration methods, improving urban infrastructure, and reverse engineering the human brain are also areas where new generations of software systems are needed to enable and deploy the sought after scientific advances. But meeting these grand challenges depends on more than robust or well-engineered software systems.

It appears likely that the development of software systems in these other application domains will increasingly depend in part or full on FOSS systems and ecosystems, as well as FOSSD processes, practices, and project forms. The reasons for this are many, but not inevitable. Social choices and economic constraints may make proprietary or closed source system solutions less practical and less desirable. For example, if scientific research into fusion energy centers around the International Thermonuclear Energy Research (ITER) project, with its forecasted budget of more than $100B (and it is still in the early stages of development), then how much of that budget will be allocated to development of ITER control system software, and who will be called upon to develop or engineer the requisite software? ITER is a multi-national effort, and there is likely to be a common call for openness in its software development projects, as well as openness in science practices, rather than an expectation that some company or contractor will be called upon to develop a proprietary, closed source software system. As such, it may be the case that grand challenge problems are more likely to embrace or demand openness in their system development efforts in part or full, or at least prior to any commercialization of supporting software systems.

FOSS system development has already begun to transform the global software industry and all major software and Information Technology (IT) firms. Proprietary, closed source systems are not likely to disappear, but there will be growing pressure to expect that proprietary systems offer innovative features or functions that are not yet available as FOSS systems. FOSS systems may help to drive technological advances in proprietary systems and closed source system developers, out of self interest and preservation of commercial or market position. Once again, FOSS system are a creative driver that helps stimulate advances to the broader economy and IT marketplace. Companies and firms that actively resist the progressive transition to FOSS systems in different application or service areas will be increasingly marginalized, rather than embraced. FOSS systems will increasingly take over mundane, infrastructural, and non-competitive IT domains, and this will help to clarify where IT or software system value truly is to be found. So stimulating research and development into FOSS systems and FOSSD projects are a strategic national investment, if the goal is to improve national and industrial IT system capabilities and related industries.

Finally advances in enterprise information systems that help realize new ways and means for improving or streamlining enterprise operations, creating new products or services, and creating more stimulating jobs, careers, and workforce development opportunities depend on faster, better, and cheaper software systems. Helping to make regional and national government agencies more transparent, open, and trustworthy requires public access to information systems that are easy to access, open for study and open to citizen participation. FOSS systems are the most likely technology that can realize these societal needs.

## 5. CONCLUSIONS

Free and open source software development is emerging as an alternative approach for how to develop large software systems. FOSSD employs socio-technical work practices, development processes, and community networking often different from those found in industrial software projects, and those portrayed in software engineering textbooks [17]. As a result, FOSSD offer new types and new kinds of practices, processes, and organizational forms to discover, observe, analyze, model, and simulate. Similarly, understanding how FOSSD practices, processes, and projects are similar to or different from traditional SE counterparts is an area ripe for further research and comparative study. Many new research opportunities exist in the empirical examination, modeling, and simulation of FOSSD activities, efforts, and communities.

FOSSD project source code, artifacts, and online repositories represent and offer new publicly available data sources of a size, diversity, and complexity not previously available for SE research, on a global basis. For example, software process modeling and simulation research and application has traditionally relied on an empirical basis in real-world processes for analysis and validation. However, such data has often been scarce, costly to acquire, and is often not available for sharing or independent re-analysis for reasons including confidentiality or non-disclosure agreements. FOSSD projects and project artifact repositories contain process data and product artifacts that can be collected, analyzed, shared, and be re-analyzed in a free and open source manner.

Last, through surveys of empirical studies of FOSSD projects [4, 5, 11], it should be clear there are an exciting variety and diversity of opportunities for new research into software development processes, work practices, project/community dynamics, and related socio-technical interaction networks. Thus, you are encouraged to consider how your efforts to engage in SE research or apply FOSSD concepts, techniques, or tools can be advanced through studies that examine FOSSD activities, artifacts, and projects.

## 6. ACKNOWLEDGMENTS

The research described in this paper has been supported by grant #0808783 from the U.S. National Science Foundation, grant #N00244-10-1-0038 from the Acquisition Research Program at the Naval Postgraduate School, and a grant from the Computing Community Consortium. No review, approval, or endorsement implied.

## 7. REFERENCES

[1] Boehm, B.E., A View of 20th and 21st Century Software Engineering. *Proc. 28th International Conference on Software Engineering (ICSE 2006)*, Shanghai, China, 12-29, ACM Press, 2006.

[2] Computing Community Consortium Workshop on Free/Open Source Software Development, http://cra.org/ccc/foss.php and http://foss2010.isr.uci.edu/, 10-12 February 2010, accessed June 2010.

[3] Cook, J.E., Votta, L.G., and Wolf, A.L., Cost-Effective Analysis of In-Place Software Processes, *IEEE Trans. Software Engineering*, 24(8), 650-663, 1998.

[4] Crowston, K., Wei, K., Howison, J., and Wiggins, A. (2010). Free/libre open source software development: what we know and what we do not know. *ACM Computing Surveys*, (in press).

[5] Hauge, O., Ayala, C. and Conradi, R. (2010). Adoption of Open Source Software in Software-Intensive Organizations - A Systematic Literature Review. *Information and Software Technology,* (in press).

[6] Harrison, W., Editorial: Open Source and Empirical Software Engineering, *Empirical Software Engineering*, 6(2), 193-194, 2001.

[7] Howison, J., Conklin, M., and Crowston, K., FLOSSmole: A Collaborative Repository for FLOSS Research Data and Analyses. *Intern. J. Info. Tech. and Web Engineering*, 1(3), 17-26, 2006.

[8] Robles, G. and Gonzalez-Baharona, J.M., Contributor Turnover in Libre Software Projects, in Damiani, E., Fitzgerald, B., Scacchi, W., Scotto, M. and Succi, G., (Eds.), *Open Source Systems*, IFIP Vol. 203, Springer, Boston, 273-286, 2006.

[9] Sack, W., Detienne, F., Ducheneaut, Burkhardt, Mahendran, D., and Barcellini, F., A Methodological Framework for Socio-Cognitive Analyses of Collaborative Design of Open Source Software, *Computer Supported Cooperative Work*, 15(2/3), 229-250, 2006.

[10] Scacchi, W., Understanding Free/Open Source Software Evolution, in N.H. Madhavji, J.F. Ramil and D. Perry (Eds.), *Software Evolution and Feedback: Theory and Practice*, John Wiley and Sons Inc, New York, 181-206, 2006.

[11] Scacchi, W. Free/Open Source Software Development: Recent Research Results and Methods, in M. Zelkowitz (Ed.), *Advances in Computers*, 69, 243-295, 2007.

[12] Scacchi, W., Crowston, K. Jensen, C., Madey, G., Squire, M., et al., *Towards a Science of Open Source Systems*, Final Report, Institute for Software Research, University of California, Irvine, Fall 2010. http://foss2010.isr.uci.edu/content/foss-2010-reports accessed September 2010.

[13] Scacchi, W., Crowston, K., Madey, G., and Squire, M. Envisioning National and International Research on the Multi-Disciplinary Study of Free/Open Source Software, Spring 2009. http://www.ics.uci.edu/~wscacchi/ProjectReports/CCC-FOSS-SPRING2009.pdf, accessed June 2010.

[14] Scacchi, W., Feller, J. Fitzgerald, B., Hissam, S., and K. Lahkani, Understanding Free/Open Source Software Development Processes, *Software Process--Improvement and Practice*, 11(2), 95-105, March/April 2006.

[15] Scacchi, W., Jensen, C., Noll. J. and Elliott, M.E. Multi-Modal Modeling, Analysis and Validation of Open Source Software Development Processes, *Intern. J. Internet Technology and Web Engineering*, 1(3), 49-63, 2006.

[16] Seaman, C.B. and Basili, V., Communication and Organization: An Empirical Study of Discussion in Inspection Meetings, *IEEE Trans. Software Engineering*, 24(6), 559-572, 1998.

[17] Sommerville, I., *Software Engineering, 8$^{th}$ Edition*, Addison-Wesley, New York, 2006.