Opportunities and Challenges for Modeling and Simulating Free/Open Source Software Processes

Walt Scacchi
Institute for Software Research
Donald Bren School of Information and Computer Sciences
University of California, Irvine
Irvine, CA 92697-3425 USA
+1-949-824-4130 (v), +1-949-824-1715 (f)
Wscacchi@uci.edu

30 September 2004

Abstract

The focus of this article is on identifying what is known about the processes found in free and open source software development (F/OSSD) projects. The results and findings from a survey of empirical studies of F/OSSD give rise to an interesting variety of opportunities and challenges for modeling and simulating these processes, which are identified along the way. The article also presents a framework for organizing new studies according to their research focus, sampling strategy, data collection methods, modeling and simulation approach, and the strategy used to assure the quality of the modeling and simulation results. This framework thus serves to help guide research into F/OSSD processes in ways that maximize and bound their contribution.

Keywords: Open source software, software process modeling, software process simulation

Introduction

This article explores patterns and processes that emerge in free/open source software development (F/OSSD) projects. F/OSSD is a relatively new way for building and deploying large software systems on a global basis, and differs in many interesting ways from the principles and practices traditionally advocated for software engineering. Hundreds of F/OSS systems are now in use by thousands to millions of end-users, and some of these F/OSS systems entail hundreds-of-thousands to millions of lines of source code. So what's going on here, and how are F/OSSD processes that are being used to build and sustain these projects different, and how might process modeling and simulation techniques be used to explore what's new or different?

One of the more significant features of F/OSSD is the formation and enactment of complex software development processes performed by loosely coordinated software developers and contributors. These people may volunteer their time and skill to such effort, and may only work at their personal discretion rather than as assigned and scheduled. Further, these developers generally provide their own computing resources, and bring their own software development tools with them. Similarly, F/OSS developers work on software projects that do not typically have a corporate owner or management staff to organize, direct, monitor, and improve the software development processes being put into practice on such projects. But how are successful F/OSSD projects and processes possible without regularly employed and scheduled software development staff, or without an explicit regime for software engineering project management? Why will software developers participate in

F/OSSD projects? Why and how are large F/OSSD projects sustained? How are large F/OSSD projects coordinated, controlled or managed without a traditional project management team? Why and how might these answers to these questions change over time? These are the kinds of questions that will be addressed in this article.

The remainder of this article is organized as follows. The next section provides further background on what F/OSSD is and what is already known about F/OSSD practices, based on both trade studies and systematic empirical studies. This survey focuses attention to identifying opportunities and challenges in F/OSSD process modeling and simulation. Following this is the presentation of a framework and organizing and framing further studies of F/OSSD processes that build on the foundational components identified in the survey. A final discussion then argues for why the software process modeling and simulation community may itself want to adopt F/OSSD practices.

What is free/open source software development?

Free (as in freedom) software and open source software are often labeled or treated as the same thing. However, there are important differences between them with regards to the licenses assigned to the respective software. Free software generally appears licensed with the GNU General Public License (GPL), while OSS may use either the GPL or some other license that allows for the integration of software that may not be free software. Free software is a social movement [cf. Elliott and Scacchi 2004], whereas OSSD is a software development methodology, according to free software advocates like Richard Stallman and the Free Software Foundation. However, free software is always available as OSS, but OSS is not always free software[1]. This is why it often is appropriate to refer to F/OSS in order to accommodate two similar or often indistinguishable approaches to software development. Subsequently, for the purposes of this article, focus is directed at F/OSSD processes, rather than to software licenses and social movements with free or open source software, though each may impinge on F/OSSD processes.

F/OSSD is mostly not about software engineering, at least not as SE is portrayed in modern SE textbooks. F/OSSD is not SE done poorly. It is instead a different approach to the development of software systems where much of the development activity is openly visible, development artifacts are publicly available over the Web, and generally there is no formal project management regime, budget or schedule. F/OSSD is also oriented towards the joint development of a community of developers and users concomitant with the software system of interest. F/OSS developers are also end-users of the F/OSS they develop, and other end-users often participate in and contribute to F/OSSD efforts. There is also widespread recognition that F/OSSD projects can produce high quality and sustainable software systems that can be used by thousands to millions of end-users [Mockus, Fielding, Herbsleb 2002]. Thus, it is reasonable to assume that F/OSSD processes are not necessarily of the same type, kind, or form found in SE projects that follow the processes described in modern SE textbooks. While such approaches might be used within an SE project, there is no

[1] Thus at times it may be appropriate to distinguish conditions or events that are generally associated or specific to either free software development or OSSD, but not both.

basis found in the principles of SE laid out in textbooks that would suggest SE projects typically adopt or should practice F/OSSD methods. Subsequently, what is known about SE processes, or modeling and simulating SE processes, may not be equally applicable to F/OSSD processes without some explicit rationale or empirical justification. Thus, it is appropriate to survey what is known so far about F/OSSD.

Results from recent studies of F/OSSD

There are two kinds of studies that offer some insight or findings on F/OSSD practices each in turn reflects on different kinds of processes which are not well understood at this time. First, there are trade studies that focus on convenience (snowball sample) surveys of software or IT industry professionals who are early adopters of F/OSS techniques. Second, there are systematic empirical studies of F/OSSD projects using small/large research samples and analytical methods drawn from different academic disciplines. Both kinds of studies stand in contrast to the popular examination of F/OSSD practices offered by F/OSS advocates [e.g., DiBona 1999, Pavelicek 2000]. These popular treatments tend to be grounded in personal experiences of the authors, though such experiences are valuable because they are often a source of insight or questions for further inquiry.

Trade/Industry studies

Among the more widely identified industry studies are those that have been sponsored and published by *CIO* magazine (www.cio.com), starting back in 2002. These studies of the opinions and experiences of hundreds of IT managers and executives in a variety of enterprise settings report the following kinds of findings:

First, in these enterprises, OSSD projects are primarily targeted to new system deployments, rather than to supporting or replacing existing business system applications. Second, the primary benefits for engaging OSSD projects include anticipation of lower total cost of ownership (TCO), lower capital investment, and greater reliability of the resulting systems. Third, the perceived risks or weaknesses associated with in-house OSSD projects include lack of in-house OSSD skills or OSS developers in the market, uncertainty over long-term vendor support or vendor viability (especially among small and medium size OSS product firms), and uncertainty over the costs of switching from current approaches and vendors to OSS-oriented ones.

From the perspective of software process modeling and simulation, the following kinds of observations appear to follow from these results. First, the costs associated with OSSD projects are unclear, as are the methods for accounting for them and associating them with different OSSD processes or activities. Second, if the survey participants work in enterprises that explicitly manage their traditional software development processes, they recognize that OSSD projects seem to require different, less familiar processes that may not be well understood by their current software development staff. So the opportunity exists to statically modeling and dynamically simulate and map out: (a) where resources (e.g., costs) are being assigned, used or consumed; (b) how OSSD processes differ from traditional ones; and (c) whether or how such differences can account for anticipated changes in conventional vs. open source software development and maintenance costs, and in resulting software quality.

Findings from F/OSSD research studies

A number of Web-based repositories of research papers that report on studies on F/OSSD projects have begun to appear. Among them are those at MIT (opensource.mit.edu) with almost 200 papers contributed, and at Cork College in Ireland (opensource.ucc.ie) which features links to multiple special issue journals and proceedings from international workshops of OSS research. Rather than attempt to survey the complete universe of studies in these collections, the choice instead is to just briefly sample a small set of studies that raise interesting issues or challenging problems for software process modeling and simulation.

One important qualifier to recognize is that the studies below generally examined carefully identified F/OSSD projects or a sample of projects, so the results presented should not be assumed to apply to all F/OSSD projects, or to projects that have not been studied. Furthermore, it is important to recognize that F/OSSD is no silver bullet that resolves the software crisis. Instead it is fair to recognize that most of the nearly 100,000 F/OSSD projects associated with Web portals like SourceForce.org have very small teams of two or less developers [Madey 2004], and many projects are inactive or have yet to release any operational software. However, there are now at least a few thousand F/OSSD projects that are viable and ongoing, so that there is a sufficient universe of diverse F/OSSD projects to investigate, and to model and simulate their software processes. Consequently, consider the research findings reported or studies cited below as starting points for further investigation, rather than as defining characteristics of most or all F/OSSD projects or processes.

Motivating, joining, participating, and contributing to F/OSSD projects

One of the most common questions about F/OSSD projects is why software developers will join and participate in such efforts, often without pay for sustained periods of time. A number of surveys of F/OSS developers [Ghosh 2000, Lakhani 2002, Hars 2002, Hann 2002, Hertel 2003] has posed such questions, and the findings reveal the following.

First, F/OSS developers generally find the greatest benefit from participation is the opportunity to learn and share what they know about software system functionality, design, methods, tools, and practices associated with specific projects or community leaders. F/OSSD is a venue for learning for individuals, project groups, and organizations, and learning organizations are ones which can continuously improve or adapt their processes and practices [Huntley 2003, Ye 2003]. However, though much of the development work in F/OSSD projects is unpaid or volunteer, individual F/OSS developers often benefit with higher average wages and better employment opportunities (at present), compared to their peers lacking F/OSSD experience or skill [Hann 2002, Lerner 2002].

Second, F/OSS developers appear to really enjoy their F/OSSD work [Hertel 2003], and to be recognized as trustworthy and reputable contributors [Stewart 2001]. F/OSS developers also self-select the technical roles they will take on as part of their participation in a project [Ye 2003, Gacek 2004], rather than be assigned to a role in a traditionally managed SE project, where the assigned role may not be to their liking.

Third, many F/OSS developers participate in and contribute to multiple F/OSSD projects. In one study, 5% of developers surveyed reported participating in 10 or more F/OSSD projects [Hars 2002]. However, the vast majority of source code that becomes part of F/OSS released by a project is typically developed by a small group of core developers who control the architecture and direction of development. Subsequently, most participants typically contribute to just a single module, though a small minority of modules may be include patches or modifications contributed by hundreds of contributors [Ghosh 2000].

Consequently, how and why software developers will join, participate in, and contribute to an F/OSSD project seems to represent a new kind of process affecting how F/OSS is developed and maintained [von Krogh 2003, Scacchi 2004c]. Subsequently, modeling and simulating what this process is, how it operates, and how it affects software development is an open research challenge.

Alliance formation and inter-project social networking

How does the gathering of individual F/OSS developers give rise to a more persistent project team or self-sustaining community? Through choices that developers make for their participation and contribution to an F/OSSD project, they find that there are like-minded individuals who also choose to participate and contribute to a project. These software developers find and connect with each other through F/OSSD Web sites and online discourse (e.g., threaded email discussions) [Monge 1998], and they find they share many technical competencies, values, and beliefs in common [Crowston 2002, Espinosa 2002, Elliott 2004]. This manifests itself in the emergence of an occupational network of F/OSS developers [Elliott 2003].

Becoming a central node in a social network of software developers that interconnects multiple F/OSS projects is also a way to accumulate social capital and recognition from peers. However, it also enables the merger of independent F/OSS systems into larger composite ones that gain the critical mass of core developers to grow more substantially and attract ever larger user-developer communities [Madey 2004, Scacchi 2004c]. "Linchpin developers" [Madey 2004] participate in or span multiple F/OSSD projects. In so doing, they create alliances between otherwise independent F/OSSD projects. Figure 1 depicts an example of a social network of 24 F/OSS developers within 5 F/OSS projects that are interconnected through two linchpin developers [Madey 2004]. Such interconnection enables small F/OSS projects to come together as a larger social network with the critical mass [Marwell 1993] needed for their independent systems to be merged and experience more growth in size, functionality, and user base. F/OSSD Web sites also serve as hubs that centralize attention for what is happening with the development of the focal F/OSS system, its status, participants and contributors, discourse on pending/future needs, etc

Thus interesting problems arise when investigating how best to model or simulate the processes of alliance formation and inter-project social networking, and how such processes facilitate or constrain F/OSSD activities, tool usage, and preference for which development artifacts are most valued by project participants.

Community development and system development

Developing F/OSS systems is a community and project team building process that must be institutionalized within a community [Sharma 2002, Smith 1999, Preece 2000, Ye 2004] for its software informalisms (artifacts) and tools to flourish. Downloading, installing, and using F/OSS systems acquired from other F/OSS Web sites is also part of a community building process [Kim 2000]. Adoption and use of F/OSSD project Web sites are a community wide practice for how to publicize and share F/OSS project assets. These Web sites can be built using F/OSSD Web site content management systems (e.g., PhP-Nuke) to host project contents that can be served using F/OSS Web servers (Apache), database systems (MySQL) or application servers (JBoss), and increasingly accessed via F/OSS Web browsers (Mozilla). Furthermore, ongoing F/OSSD projects may employ dozens of F/OSS development tools, whether as standalone systems like the software version control system CVS, as integrated development environments like NetBeans or Eclipse, or as sub-system components of their own F/OSS application in development. These projects similarly employ asynchronous systems for project communications that are persistent, searchable, traceable, public and globally accessible [Yamauchi 2000].

F/OSS systems, hyperlinked artifacts and tools, and project Web sites serve as venues for socializing, building relationships and trust, sharing and learning with others. Community building, alliance forming, and participatory contributing are essential and recurring activities that enable F/OSSD projects to persist without central corporate authority. Linking people, systems, and projects together through shared artifacts and sustained online discourse enables a sustained socio-technical community, information infrastructure [Jensen 2004b], and network of alliances [Monge 1998] to emerge.

Thus interesting problems arise when investigating how best to model or simulate the F/OSSD processes that facilitate and constrain the co-development and co-evolution of F/OSS project communities and the software systems they produce. The point is not to separate the development and evolution processes of the software system from its community, since each is co-dependent on the other, and the success of one depends on the success of the other. Thus, they must be modeled and simulated as integrating and intertwining processes.

Software evolution in a multi-project software ecosystem

As noted above, many F/OSSD projects have become interdependent through the networking of software developers, development artifacts, common tools, shared Web sites, and computer-mediated communications. What emerges from this is a kind of multi-project software ecosystem, whereby ongoing development and evolution of one F/OSS system gives rise to propagated effects, changes, or vulnerabilities in one or more of the projects linked to it [Jensen 2004b]. These interdependencies are most apparent when F/OSSD project share source code modules or components. In such situations, the volume of source code of an individual F/OSSD project may appear to grow at a super-linear or exponential rate [Scacchi 2004b, Schach 2002, Smith 2004].

Such an outcome, which economists and political scientists refer to as a "network externality" [Ostrom 1990], may be due to the import or integration of shared components, or the replication and tailoring of device, platform, or internationalization specific code modules. Such system growth patterns might challenge the well-established laws of software evolution [Lehman 1980, 2002]. Thus, software evolution in a multi-project F/OSS ecosystem is a process of co-evolution of interrelated and interdependent F/OSSD projects, people, artifacts, tools, code, and project-specific processes.

It seems reasonable to observe that the world F/OSSD is not the only place where multi-project software ecosystems emerge, as software sharing or reuse within traditional software development enterprises is common. However, the process of the co-evolution of software ecosystems found in either traditional or F/OSSD projects in mostly unknown. Thus, software co-evolution within an F/OSS ecosystem represents an opportunity for research that investigates understanding such a software evolution process through studies supported by modeling and simulation techniques.

Comparative analysis of development tool-centered processes

Some software processes are embodied or anchored in certain software development tools. Using the tools as intended enacts the embodied process, whereas doing software development without such tools generally implies performing a different process. One place such a situation is apparent is with activities and tools associated with software version, configuration, and release management (SVCRM).

SVCRM can be organized and performed as a process with or without SVCRM tools or environments. In F/OSSD projects, there is a diversity of approaches and tools to SVCRM [Erenkrantz 2003]. For example, most F/OSSD projects rely on CVS, the concurrent version system [Fogel 1999], to provide a centrally located and administered repository of source code and other project artifacts. However, many F/OSSD projects and developers have found CVS lacking many important configuration and release management functions that might better serve their SVCRM needs. Some are migrating to the new, still in early development, Subversion SVCRM system, which is said to remedy many of CVS limitations. The Linux Kernel project associated with Linus Torvalds supports a more decentralized approach based on a commercial, non-F/OSS SVCRM system called BitKeeper, which has been the focus of sustained debate within the Linux Kernel project. Next, the Apache Web Server Project adopted still another approach whereby the focus on frequent release of software patches (hence, "a patch(y) server"), and the infrequent release of packaged software versions for wider distribution. Last, the GNU Arch project seeks to develop a more comprehensive tool-based environment that subsumes the functionality of the SVCRM tools in these different approaches, and to do insure that SVCRM becomes a process that can be completely supported with free software.

Each of these alternative approaches proscribes different activities to be accomplished as part of the SVCRM process. Each approach relies of different tools or software functions to help accomplish the process. Thus, what are the similarities and differences among these tool-based approaches to the SVCRM process? What are the variants of the SVCRM process these tools support? What is the SVCRM process that can be supported by a single integrated and comprehensive SVCRM environment? As before, answering these questions would directly benefit from the ability to model and simulate tool-centered processes found in different F/OSSD projects. Such capability would also apply equally well to SVCRM tools and processes associated with proprietary software development projects, as well as to other SE tool-centered processes.

Role-task migration within or across F/OSSD projects

Studies have observed and identified the many roles that participants in an F/OSSD project perform [Ye 2003, Gacek 2004]. These roles are used to help explain who does what, which serves as a precursor to explanations of how F/OSSD practices or processes are accomplished. In addition, these roles are partially ordered and visualized as role-task hierarchies or as concentric/nested circles to indicate their centrality to the project. However such a division of labor is dynamic, not static. This means that participants can move through different roles throughout the course of a project over time, depending on their interest, commitment, and technical skill. Initially, participants start at the periphery of a project in the role of end-user by downloading and using the F/OSS associated with the project. They can then move into roles like bug-reporter, code reviewer, code/patch contributor, module owner (development coordinator), and eventually to core developer or project leader. Moving through these roles requires effort, and the passage requires being recognized by other participants as a trustworthy and accomplished contributor. This model of organization is sometimes called a “meritocracy” [Fielding 1999], and it serves to provide a stable form of project leadership, overall direction, a basis for shared belief in what the collective goals of the project are. In sum, these give rise to virtual project management capability—the effective accomplishment of software project management at a distance but without traditional project managers or budgets [Scacchi 2004a].

Role-task migration can and does arise within F/OSSD projects, as well as across projects. Social networking, software sharing, and project internetworking enables this. But how do role-task migration processes or trajectories facilitate or constrain how F/OSSD occurs? Role-task migration does not appear as a topic addressed in traditional SE textbooks or studies, yet it seems to be a common observation in F/OSSD projects. Subsequently, modeling and in particular simulating the role-task migration process, and how it affects or contributes to other software development or quality assurance processes, is an area requiring further investigation.

Peer review and collective decision making

Eric S. Raymond, a widely recognized advocate of F/OSSD, is known for his conjecture that "with enough eyes, all bugs are shallow" [cf. DiBona 1999, Pavelicek 2000]. This portends a relationship between software quality and collective debugging activities. Clearly, reviews and inspections by developers or others not directly involved in constructing a software system are a basic tenet of software quality assurance. It is achieved through code reading and defect detection via peer review. Software inspection and review processes are an established part of SE [Fagan 1976, Ebenau 1994], though industrial practice indicates difficulties exist in the adoption, routine application, and cycle time (inspection interval) of such processes [Porter 1997]. However, are inspection and review processes found in F/OSSD projects the same as those described in SE textbooks, or as practiced in SE projects in non-F/OSSD projects? Answering such a question poses the possibility of finding F/OSSD projects may have adopted (and adapted) a well-established SE principle and demonstrated its practical value in assuring and improving the quality of large software. But at the same time, industrial software development firms may not practice such SE principles (to their detriment). Thus, F/OSSD projects may be found to be practicing well understood SE principles in a manner more viable than current industrial practice, and thus account for some of the software quality differences that have been reported [cf. Mockus 2002].

Following this, it seems there is an opportunity to investigate whether or how (a) peer review processes in F/OSSD projects resemble/differ from those (b) advocated by SE principles, or those (c) practiced by industrial firms developing proprietary (non-F/OSS) systems. Software process modeling and simulation would seem to be an essential part of such an analysis.

Comparing F/OSSD and SE Processes

The last category of related studies seek to identify and compare software development processes found in F/OSSD projects with those described or prescribed for SE projects, rather than just the resulting software products [Succi 2004]. Mockus, Fielding, and Herbsleb [2002] very briefly describe the processes accounting for the development of the Apache Web Server and the Mozilla Web Browser. Such an account does not provide sufficient content to directly model, simulate, or compare them to traditional SE processes. Reis and Fortes [2002] provide one of the first in-depth examinations of the overall process accounting for the development of the Mozilla Web Browser. They identify different developer roles, tools being used, artifacts created, and activities performed, which potentially provides adequate information for modeling the process. Scacchi [2002] provides a narrative description of the software requirements process found in a sample of F/OSSD projects and compares it to the requirements engineering process portrayed in modern SE textbooks. The F/OSSD projects examined span applications in application domains including Internet infrastructure, networked computer games, astrophysics, and academic software design research. In a related study [Scacchi 2004a], he identifies differences in software processes for requirements and design, configuration management, evolution, project management, and software technology transfer from those in SE texts as found in comparative study of multiple F/OSSD projects of a common type, networked computer games. Further, in two addition studies, he examines data and models accounting for software evolution [cf. Lehman 2002] compared to those emerging for F/OSSD [Scacchi 2004b], and also emerging socio-technical processes found in F/OSSD projects that intermingle social (e.g., team, group, and individual) and technical development processes [Scacchi 2004c, Truex 1999]. All of these studies describe processes found in different F/OSSD projects using narrative descriptions or models. Thus these processes are yet to be modeled or simulated in more formal or computational ways.

Finally, in recent studies, effort to model and simulate F/OSSD processes of different kinds has begun to appear. Antoniades, *et al*. [2004] and Smith *et al*. [2004] both provide simulation models of processes accounting for the overall development or evolution of multiple F/OSSD projects. Both efforts rely on models expressed as continuous functions through either algebraic formulae or systems of equations. Such an approach to process simulation and modeling appears well matched to Systems Dynamics-based process simulation tools. In contrast, Jensen and Scacchi [2004a,b] model and re-enact processes found in a small sample of OSSD projects using language-based process models and a process re-enactment simulator [Noll 2001]

Overall, the sample of F/OSSD research studies and findings presented above reveals a number of interesting challenges for research in modeling and simulating F/OSSD processes. However, these studies are all grounded in an empirical basis where different types of processes are being examined in different types of F/OSSD projects of varying sample size and data collection methodology. So the basic challenge at hand is how to organize, reframe, and make clear what the challenges are in modeling and simulating F/OSSD processes.

F/OSS process modeling and simulation challenges

Based on the survey of studies and results emerging from empirical studies of F/OSSD projects, it becomes clear that there are many promising opportunities in studying, modeling and simulating F/OSSD processes. New sources of process data associated with F/OSSD are available, and new systematic samples of F/OSSD projects can be articulated. Similarly, new challenges for software process modeling and simulation research and application are also at hand. These can be examined as follows.

New data sources for software process studies

One of the defining characteristics of data about the F/OSSD projects is that in general is it publicly available on a global basis. Data about F/OSSD products, artifacts, and other resources is kept in repositories associated with a project's Web site. This may include the site's content management system, computer mediated communication systems (email, persistent chat facilities, and discussion forums), SVCRM systems, and networked file systems. F/OSSD process data is generally either extractable or derivable from data/content in these repositories. First-person data may also be available to those who participate in a project, even if just to inquire other participants about development activities, tools being used, the status of certain artifacts, and the like. The availability of such data perhaps suggest the a growing share of empirical software engineering research will be performed in the domain of F/OSSD projects, rather than using traditional sources of data from in-house or proprietary software development projects. These traditional non-F/OSS projects will continue to have constraints on access and publication. F/OSSD process data collection from publicly accessible repositories may also be found to be more cost-effective compared to studies of in-house software development repositories [cf. Cook 1998].

Process modeling and simulation challenges

There are a substantial variety of new or under-explored challenges for modeling and simulating F/OSSD processes. These include the following:

- How best to collect, extract, clean, compose and update software process models derived from continuously updated project repositories.
- How and when to use multiple kinds of models to characterize or simulate F/OSSD processes.
- What are the most effective methods for combining or composing multi-disciplinary software process models. As studies of F/OSSD processes may include analytical tools, techniques, or strategies from multiple disciplines (e.g., economics, organization science, management of technology, anthropology, computer-supported cooperative work, and software engineering), so should the process modeling and simulation methods acknowledge and employ these discipline-specific capabilities.
- New kinds and types of software processes have been observed and studied in F/OSSD projects, including role-task migration, socio-technical community and software co-development, and exponential evolutionary growth of the most successful F/OSSD project's source code base. How are these processes most effectively modeled?
- Basic software development processes associated with requirements development and software design seem to be primarily dependent of the use of software informalisms. These informalisms tend to be more like conversational artifacts (threaded email messages or chat transcript), than traditional engineering artifacts

(e.g., design diagrams). As such, what is the form of software processes that can be modeled to account for the production and consumption of F/OSSD informalisms through conversational or computer-mediated communication systems?

- Last, most F/OSSD projects do not succeed in achieving a critical mass of core developers, but those that do often develop their software systems in a continuously emerging manner. Thus, yesterday's system and functionality is not tomorrow's system and functionality. As such, what is needed to model software processes that are continuously evolving and adapting to new circumstances, platforms and applications.

The diversity of this set of enumerated software process modeling challenges points to the richness of the field of software process research targeted to modeling F/OSSD processes. Simulating F/OSSD processes also poses a complementary diversity of challenges.

A framework for F/OSSD process modeling and simulation research

From the preceding section, it appears that the most likely research focus in F/OSSD process modeling and simulation will examine one or more of the following:

- exploratory studies of new F/OSSD process types;
- modeling and simulation of F/OSSD processes;
- multi-disciplinary modeling and simulation of F/OSSD processes; and

- modeling and simulation of continuous F/OSSD processes.

Sampling strategies that support of the F/OSSD process modeling and simulation foci will examine either:

- *Single process within a single project*, such as the SVCRM process in the Linux Kernel project. Such a sample is important when examining high profile F/OSSD projects, where the selected process may be unique or untypical of other F/OSSD projects.
- *Multiple processes within a single project,* where such a sample focuses attention to a high-profile F/OSSD project in order to account for some overall development phenomena, such as how the constellation of F/OSSD processes accounts for the evolution of the project's software system growth.
- *Single process found in multiple projects*, where emphasis is on understanding the form and variations of the selected F/OSSD process through comparative analysis across F/OSSD projects.
- *Multiple processes found in multiple projects*, where emphasis is on understanding the form and variation of overall F/OSS development or evolution process, across projects over time. Sub-samples may further focus attention to F/OSSD processes within multiple projects of a common type (e.g., Internet infrastructure or networked computer games), and finally multiple projects across multiple project types (infrastructure, games, and science).

- *Population studies* are focuses on studies that seek to characterize the overall population or universe of F/OSSD projects, as perhaps might be associated with a specific F/OSS Web portal like SourceForge.org, Freshmeat.org, or Savannah.org.

Data collection methodologies in support of F/OSSD process modeling and simulation include:

- *Ethnographic and qualitative field studies*, especially when emphasizing social, cultural, or socio-technical processes within a single project [Viller 2000].
- *Case studies and comparative case studies*, when focussing on in-depth comparisons of processes of the same type in different F/OSSD projects, or more comprehensive studies that examine multiple types of processes across multiple types of F/OSSD projects.
- *Data mining F/OSS repositories*, when relying on software product data as the source for extracting or discovering software process through automated indirect means, either for repositories within a single project, or across multiple project types.
- *Surveys, questionnaires, or online polls* are well suited when seeking to ascertain processes that are shaped by participants perceptions, beliefs, or opinions, especially when large samples of participants is available.
- *Triangulation and convergence methods* seek to build on the use of many of the preceding data collection methodologies, so as to be able to characterize, model, and simulate F/OSSD process from multiple perspectives, supported by multiple kinds of process data.

Last, every empirical study requires or benefits from an explicit strategy for assuring the quality of the models and simulations produced. As before, a variety of choices are available, though they generally depend on choices made for the preceding framework components. The strategies seen in the surveyed studies cover the following range of assurance alternatives:

- *Packaging and fit* is the baseline form of assurance that addresses how the analytical variables were identified and composed, which determines whether the analysis presented makes sense, is coherent, fits the data to the model, and rules out other alternative explanations that would refute the model.
- *Reliability and construct validity* are often used to explain variance measures that result from a factor analysis of quantitative data. Such assurance is focused on quantitative process data.
- *External validity and traceability* focuses attention to whether the participants engaged in performing the process can make sense, can trace process fragments back to their source data, and be satisfied that the analysis offers them something of value, such as process improvement options.
- *Cross-comparative grounded theory* assures the resulting process model is based on data arising from comparative ethnographic methods. The process model is composed, compared, cross-checked, and presented incrementally so as to provide a rich account of the process and data sources. New data will not refute the model, but instead may realize incremental model improvement and refinement.

- *Cross-domain theories* provide multi-discipline analytical methods and theoretical perspectives that collectively serve to explain the modeled process was constructed, what it explains, and what multi-disciplinary associations it makes.

The components of a framework that accounts for how F/OSSD process modeling and simulation studies may be organized or structured. Table 1 presents a sample of different values of each of the framework's components, and thus suggests possible study designs that may follow. Other choices allow for other research study designs, but what should be clear is that the domain of F/OSSD process modeling and simulation is diverse and wide-ranging. Thus, it offers a huge selection of research opportunities and challenges that require further study and contribution.

Research focus	**Sample strategy**	**Data collection methodology**	**Modeling and Simulation strategy**	**Assurance strategy**
New F/OSSD process types	1-1	Ethnography	Narrative models (no simulation)	Packaging and fit
Multi-disciplinary process modeling and simulation	1-N	Mining F/OSS product/artifact repositories	Systems Dynamics models and simulation	Reliability
F/OSSD process modeling and simulation	M-1	Comparative case studies	Semi-structured hypermedia and discrete-event simulation	Validity and comparative domain theories
Modeling and simulating of continuous F/OSSD processes	M-N	All of the above via triangulation and convergence	Knowledge-based models and re-enactment simulators	Validity and comparative grounded theory

Table 1. A framework of possible studies for modeling and simulating F/OSSD processes.

Discussion

One important dimension that has not yet been addressed in this article is whether and how the software process modeling and simulation community might adopt F/OSSD practices themselves. For example, one traditional barrier to engaging students in software process studies is the paucity of free or low-cost modeling and simulation tools. Sharing one's models and simulations with colleagues is difficult at present, if they must buy new and unfamiliar tools. The ability to reuse, re-analyze, or extend a colleague's models and simulations is similarly limited. The community needs and should directly benefit from F/OSS process models, modeling and simulation tools, and process data/model repositories that can be easily acquired or shared, studied, modified, and redistributed to the mutual benefit of all. Finally, it can also be noted that it further serves the collective interest of the community to consider how to develop a globally shared and interoperable information infrastructure for modeling and simulating software processes. This is true whether these processes are found in SE or F/OSSD projects. As a consequence, these are all opportunities for the process modeling and simulation community to realize and pursue. After all, we are the ones who will benefit from efforts to develop such free (as in freedom) and open source resources and further our own community building efforts.

Conclusions

Free and open source software development is emerging as an alternative approach for how to develop large software systems. F/OSSD employs new types and new kinds of software processes, when compared to those found in industrial software projects, and those portrayed in software engineering textbooks. As a result, F/OSSD offer new types and new kinds of processes to model and simulate. Similarly, understanding how F/OSSD processes are similar to or different from SE processes is an area ripe for further research and comparative study. Many new research opportunities exist in the empirical examination, modeling, and simulation of F/OSSD processes.

F/OSSD projects represent and offer new publicly available data sources of a size, diversity, and complexity not previously available for research, on a global basis. 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. F/OSSD projects and project repositories contain process data and product artifacts that can be collected, analyzed, shared, and be re-analyzed in an free and open source manner. F/OSS thus 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. F/OSSD is thus poised to alter the calculus of empirical software engineering, and software process modeling and simulation research is an arena that can take advantage of such a historically new opportunity.

Last, through a survey of empirical studies of F/OSSD projects and other analyses presented in this article, it should be clear there are an exciting variety and diversity of opportunities for new software process modeling and simulation research. Thus, you are encouraged to consider how your efforts to research or apply software process modeling and simulation concepts, techniques, or tools can be advanced through studies that examine processes found in F/OSSD projects.

Acknowledgements

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

References

Antoniades, I.P., Samoladas, I., Stamelos, I., Angelis, L., and Bleris, G.L., Dynamic Simulation Models of the Open Source Development Process, in S. Koch (ed.), *Free/Open Source Software Development*, 174-202, Idea Group Publishing, Hershey, PA, 2004.

Bergquist, M. and Ljungberg, J., The power of gifts: organizing social relationships in open source communities, *Info. Systems J.*, 11, 305-320, 2001.

Beyer, H. and Holtzblatt, K., *Contextual Design: A Customer-Centered Approach to Systems Designs*, Morgan Kaufmann Publishers, San Francisco, CA, 1997.

Capilupppi, A., Lago, P. and Morisio, M., Evidences in the Evolution of OS projects through Changelog Analyses, *Proc. 3rd Workshop on Open Source Software Engineering,* Portland, OR, May 2003.

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.

Crowston, K., Annabi, H., and Howison, J., Defining Open Source Software Project Success, *Proc. 24th Intern. Conf. Information Systems (ICIS-2003),* December 2003.

Crowston, K., and Scozzi, B., Open Source Software Projects as Virtual Organizations: Competency Rallying for Software Development, *IEE Proceedings--Software*, 149(1), 3-17, 2002.

Curtis, B., Krasner, H., and Iscoe, N., A Field Study of the Software Design Process for Large Systems, *Communications ACM*, 31(11), 1268-1287, 1998.

DiBona, C., Ockman, and Stone, M., *Open Sources: Voices from the Open Source Revolution*, O'Reilly Press, Sebastopol, CA, 1999.

Ebenau, R. G. & S. H. Strauss, *Software Inspection Process*. McGraw-Hill, 1994.

Elliott, M. and Scacchi, W., Free Software Developers as an Occupational Community: Resolving Conflicts and Fostering Collaboration, *Proc. ACM Intern. Conf. Supporting Group Work*, 21-30, Sanibel Island, FL, November 2003.

Elliott, M. and Scacchi, W., Free Software Development: Cooperation and Conflict in A Virtual Organizational Culture, in S. Koch (ed.), *Free/Open Source Software Development*, 152-172, Idea Group Publishing, Hershey, PA, 2004.

Erenkrantz, J., Release Management within Open Source Projects, *Proc. 3rd. Workshop on Open Source Software Engineering*, 25th. Intern. Conf. Software Engineering, Portland, OR, May 2003.

Erickson, T., Making Sense of Computer-Mediated Communication (CMC): CMC Systems as Genre Ecologies, *Proc. 33rd Hawaii Intern. Conf. Systems Sciences*, IEEE Press, 1-10, January 2000.

Espinosa, J. A., Kraut, R.E., Slaughter, S. A., Lerch, J. F., Herbsleb, J. D., Mockus, A. Shared Mental Models, Familiarity, and Coordination: A Multi-method Study of Distributed Software Teams. *Intern. Conf. Information Systems*, 425-433, Barcelona, Spain, December 2002.

Fagan, M. E., Design and Code Inspections to Reduce Errors in Program Development. *IBM Systems Journal*, 15. 182-211, 1976.

Fielding, R.T., Shared Leadership in the Apache Project. *Communications ACM*, 42(4):42-43, 1999.

Fischer, G., External and shareable artifacts as opportunities for social creativity in communities of interest, in J. S. Gero and M. L. Maher (eds), *Proc. Computational and Cognitive Models of Creative Design,* 67-89, Heron Island, Australia, December 2001.

Fogel, K., *Open Source Development with CVS*, Coriolis Press, Scottsdale, AZ, 1999.

Gacek, C. and Arief, B., The Many Meanings of Open Source, *IEEE Software*, 21(1), 34-40, January/February 2004.

Ghosh, R. and Prakash, V.V., The Orbiten Free Software Survey, *First Monday*, 5(7), July 2000. Also see http://www.infonomics.nl/FLOSS/ for further information.

Hann, I-H., Roberts, J., Slaughter, S., and Fielding, R., Economic Incentives for Participating in Open Source Software Projects, in *Proc. Twenty-Third Intern. Conf. Information Systems*, 365-372, December 2002.

Hars, A. and Ou, S., Working for Free? Motivations for participating in open source projects, *Intern. J. Electronic Commerce*, 6(3), 2002.

Hertel, G., Neidner, S., and Hermann, S., Motivation of software developers in Open Source projects: an Internet-based survey of contributors to the Linux kernel, *Research Policy*, 32(7), 1159-1177, July 2003.

Huntley, C.L., Organizational Learning in Open-Source Software Projects: An Analysis of Debugging Data, *IEEE Trans. Engineering Management*, 50(4), 485-493, 2003.

Jensen, C. and Scacchi, W., Collaboration, Leadership, and Conflict Negotiation in the NetBeans.org Community, *Proc. 4th Workshop on Open Source Software Engineering*, Edinburgh, UK, May 2004a.

Jensen, C. and Scacchi, W., Process Modeling Across the Web Information Infrastructure, *Proc. 5th. Intern. Workshop on Software Process Simulation and Modeling*, Edinburgh, Scotland, May 2004b.

Kim, A.J., *Community-Building on the Web: Secret Strategies for Successful Online Communities*, Peachpit Press, 2000.

Lakhani, K.R., Wolf, B., Bates, J., DiBona, C., The Boston Consulting Group Hacker Survey, July 2002. http://www.bcg.com/opensource/BCGHackerSurveyOSCON24July02v073.pdf.

Lehman, M.M., Programs, Life Cycles, and Laws of Software Evolution, *Proc. IEEE*, 68, 1060-1078, 1980.

Lehman, M.M., Software Evolution, in J. Marciniak (ed.), *Encyclopedia of Software Engineering,* 2nd Edition, John Wiley and Sons Inc., New York, 1507-1513, 2002. Also see "Software Evolution and Software Evolution Processes," *Annals of Software Engineering*, 12, 275-309, 2002.

Lerner, J. and Tirole, J., Some Simple Economics of Open Source, *J. Industrial Economics,* 50(2), 197-234, 2002.

Madey, G., Freeh, V., and Tynan, R., Modeling the F/OSS Community: A Quantative Investigation, in S. Koch (ed.), *Free/Open Source Software Development*, 203-221, Idea Group Publishing, Hershey, PA, 2004.

Marwell, G. and Oliver, P., *The Critical Mass in Collective Action: A Micro-Social Theory*. Cambridge University Press, 1993.

Mockus, A., Fielding, R., & Herbsleb, J.D., Two Case Studies of Open Source Software Development: Apache and Mozilla, *ACM Transactions on Software Engineering and Methodology*, 11(3), 309-346, 2002.

Monge, P.R., Fulk, J., Kalman, M.E., Flanagin, A.J., Parnassa, C., and Rumsey, S., Production of Collective Action in Alliance-Based Interorganizational Communication and Information Systems, *Organization Science*, 9(3), 411-433, 1998.

Nakakoji, K., Yamamoto, Y., Nishinaka, Y., Kishida, K., and Ye,Y., Evolution Patterns of Open-Source Software Systems and Communities, *Proc. 2002 Intern. Workshop Principles of Software Evolution*, 76-85, 2002.

Noll, J. and Scacchi, W., Supporting Software Development in Virtual Enterprises, *J. Digital Information*, 1(4), February 1999.

Ostrom, E., Calvert, R., and T. Eggertsson (eds.), *Governing the Commons: The Evolution of Institutions for Collective Action*, Cambridge University Press, 1990.

Paulson, J.W., Succi, G., and Eberlein, A., An Empirical Study of Open-Source and Closed-Source Software Products, *IEEE Trans. Software Engineering*, 30(4), 246-256, April 2004.

Pavelicek, R., *Embracing Insanity: Open Source Software Development*, SAMS Publishing, Indianapolis, IN, 2000.

Porter, A.A., Siy, H.P., Toman, C.A. & Votta, L.G., An Experiment to Assess the Cost-Benefits of Code Inspections in Large Scale Software Development. *IEEE Trans. on Software Engineering*, 23, 329-346, 1997.

Preece, J., *Online Communities: Designing Usability, Supporting Sociability*. Chichester, UK: John Wiley & Sons, 2000.

Reis, C.R. & Fortes, R.P.M., An Overview of the Software Engineering Process and Tools in the Mozilla Project, *Proc. Workshop on Open Source Software Development*, Newcastle, UK, February 2002.

Sawyer, S., Effects of intra-group conflict on packaged software development team performance, *Information Systems J.,* 11, 155-178, 2001.

Scacchi, W., Understanding the Requirements for Developing Open Source Software Systems, *IEE Proceedings--Software*, 149(1), 24-39, February 2002.

Scacchi, W., Free/Open Source Software Development Practices in the Computer Game Community, *IEEE Software*, 21(1), 59-67, January/February 2004a.

Scacchi, W., Understanding Free/Open Source Software Evolution, in N.H. Madhavji, M.M. Lehman, J.F. Ramil and D. Perry (eds.), *Software Evolution*, John Wiley and Sons Inc, New York, to appear, 2004b.

Scacchi, W., Socio-Technical Interaction Networks in Free/Open Source Software Development Processes, in S.T. Acuña and N. Juristo (eds.), *Peopleware and the Software Process*, World Scientific Press, to appear, 2004c.

Schach, S.R., Jin, B., Wright, D.R., Heller, G.Z., and Offutt, A.J., Maintainability of the Linux Kernel, *IEE Proceedings – Software*, 149(1), 18-23, February 2002.

Sharma, S., Sugumaran, and Rajagopalan, B., A Framework for Creating Hybrid Open-Source Software Communities, *Information Systems J.,* 12(1), 7-25, 2002.

Sim, S.E. and Holt, R.C., "The Ramp-Up Problem in Software Projects: A Case Study of How Software Immigrants Naturalize," *Proc. 20th Intern. Conf. Software Engineering*, Kyoto, Japan, 361-370, 19-25 April, 1998.

Smith, M. and Kollock, P. (eds.), *Communities in Cyberspace*, Routledge, London, 1999.

Smith, N., Capiluppi, A. and Ramil, J.F., Qualitative Analysis and Simulation of Open Source Software Evolution, *Proc. 5th Software Process Simulation and Modeling Workshop (ProSim'04),* Edinburgh, Scotland, UK, May 2004.

Stewart, K.J. and Gosain, S., An Exploratory Study of Ideology and Trust in Open Source Development Groups, *Proc. 22nd Intern. Conf. Information Systems (ICIS-2001)*, in New Orleans, LA. 2001.

Truex, D., Baskerville, R., and Klein, H., Growing Systems in an Emergent Organization, *Communications ACM*, 42(8), 117-123, 1999

Viller, S. and Sommerville, I., Ethnographically informed analysis for software engineers, *Intern. J. Human-Computer Studies*, 53, 169-196, 2000.

von Krogh, G., Spaeth, S., and Lakhani, K., Community, joining, and specialization in open source software innovation: a case study, *Research Policy*, 32(7), 1217-1241, July 2003.

Yamauchi, Y., Yokozawa, M., Shinohara, T., and Ishida, T., Collaboration with Lean Media: How Open-Source Software Succeeds, *Proc. Computer Supported Cooperative Work Conf.* (CSCW'00), 329-338, Philadelphia, PA, ACM Press, December 2000.

Ye, Y., Nakajoki, K., Yamamoto, Y., and Kishida, K., The Co-Evolution of Systems and Communities in Free and Open Source Software Development, in S. Koch (ed.), *Free/Open Source Software Development*, 59-82, Idea Group Publishing, Hershey, PA, 2004.

Ye, Y. & Kishida, K., Towards an understanding of the motivation of open source software developers, *Proc. 25th Intern. Conf. Software Engineering*, Portland, OR, 419-429, IEEE Computer Society, May 2003.

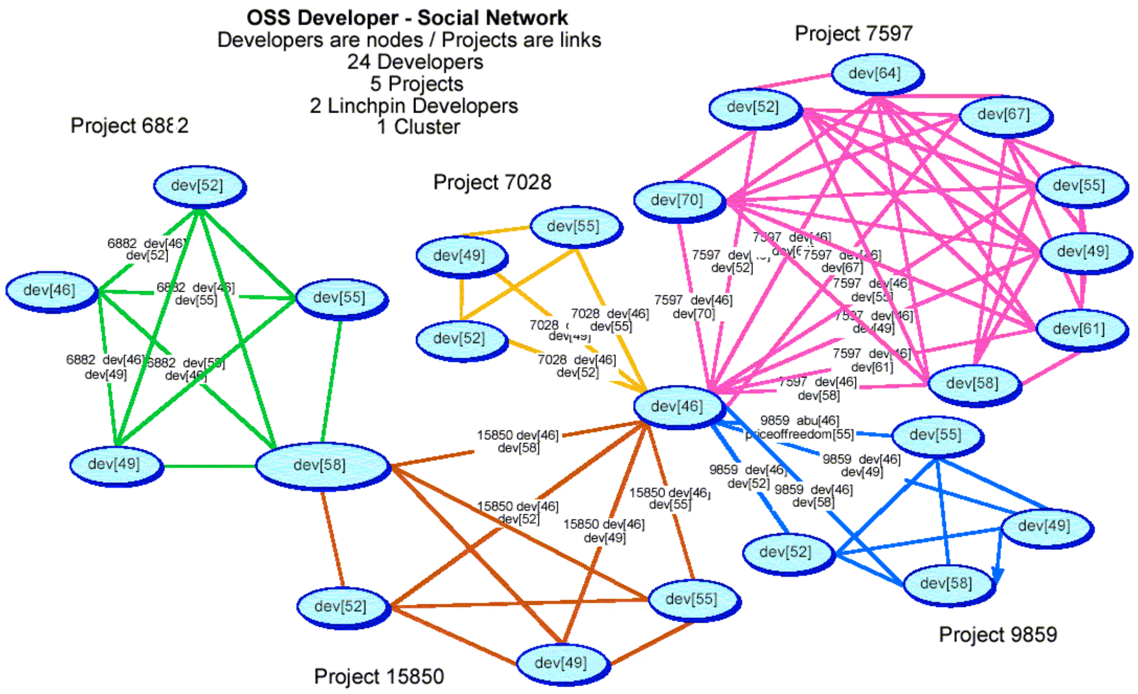

Figure 1. A social network that links 24 developers in five projects through two key developers into a larger F/OSS project community [cf. Madey 2004].