Free/Open Source Software Development: Recent Research Results and Emerging Opportunities

Walt Scacchi
Institute for Software Research
University of California, Irvine
Irvine, CA 92697-3455 USA
Wscacchi@ics.uci.edu

ABSTRACT

The focus of this paper is to review what is known about free and open source software development (FOSSD) work practices, development processes, project and community dynamics, and other socio-technical relationships. It focuses on exploring how FOSS is developed and evolved based on an extensive review of a set of empirical studies of FOSSD projects that articulate different levels of analysis. These characterize what has been analyzed in FOSSD studies across levels that examine (i) why individuals participate; (ii) resources and capabilities supporting development activities; (iii) how cooperation, coordination, and control are realized in projects; (iv) alliance formation and inter-project social networking; (v) FOSS as a multi-project software ecosystem, and (vi) FOSS as a social movement. Next, there is a discussion of limitations and constraints in the FOSSD studies so far. Last, attention shifts to identifying emerging opportunities for future FOSSD studies that can give rise to the development of new software engineering tools or techniques, as well as to new empirical studies of software development.

Categories and Subject Descriptors

D.2.0 Software, SOFTWARE ENGINEERING, *General*
K.4.2 Computing Milieux, COMPUTERS AND SOCIETY, *Social Issues*

General Terms: Design, Documentation, Human Factors, Management

Keywords: Free software, open source software, empirical studies, socio-technical relationships, software development practices

1. INTRODUCTION

This paper examines and compares practices, patterns, and processes that emerge in empirical studies of free/open source software development (FOSSD) projects. FOSSD is a way for building, deploying, and sustaining large software systems on a global basis, and differs in many interesting ways from the principles and practices traditionally advocated for software engineering (SE) [Somerville 2004]. FOSSD is not software engineering done poorly. Instead, FOSSD is different. It is a community intensive approach to the development of software systems and related artifacts and communications are openly accessible and publicly available over the Web. Thousands of FOSS systems are now in use by thousands to millions of end-users, and some of these FOSS systems entail hundreds-of-thousands to millions of lines of source code. So what's going on here, and how are FOSSD processes that are being used to build and sustain these projects different, and how might differences be employed to explain what's going on with FOSSD, and why? This paper seeks to provide some answers, though it does so by drawing extensively on a larger and more comprehensive study found elsewhere [54], which may be consulted for further study and details.

ESEC/FSE'07, September 3-7, 2007, Cavtat near Dubrovnik, Croatia.

1.1 What is FOSS/D?

Free (as in freedom/liberty) software and open source software are often treated as the same thing [e.g., 16,32]. However, there are 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 can be seen as a social movement [cf. 12], whereas OSS is just a software development methodology, according to free software advocates like Richard M. Stallman and the Free Software Foundation.

The hallmark of free software and most OSS is that the source code is available for remote access, open to study and modification, and available for redistribution to other with few constraints, except the right to insure these freedoms. OSS sometimes adds or removes similar freedoms or copyright privileges depending on which OSS copyright and end-user license agreement is associated with a particular OSS code base. More simply, free software is always available as OSS, but OSS is not always free software.

FOSS developers are typically also end-users of the FOSS they develop, and other end-users often participate in and contribute to FOSSD efforts. There is also widespread recognition that FOSSD projects can produce high quality and sustainable software systems that can be used by thousands to millions of end-users [41]. Subsequently, what is known about SE processes may not be equally applicable to FOSSD processes without some explicit rationale or empirical justification. Thus, it is appropriate to review what is known about FOSSD.

1.2 Results from recent studies of FOSSD

There are a growing number of studies that offer some insight or findings on FOSSD practices each in turn reflects on different kinds of processes that are to sample a set of studies that raise

interesting issues or challenging problems for understanding what affects how FOSSD efforts are accomplished, as what kinds of socio-technical relationships emerge along the way to facilitate these efforts. As such, consider the research findings reported below as starting points for further investigation, rather than as defining characteristics of most or all FOSSD projects or processes.

We now turn to examine what has been analyzed in FOSSD studies across different levels that examine why individuals participate in FOSSD efforts; what resources and capabilities shared by individuals and groups developing FOSS; projects as organizational form for cooperating, coordinating, and controlling FOSS development effort; alliance formation and inter-project social networking; FOSS as a multi-project software ecosystem, and FOSS as a social movement. These levels thus span the study of FOSSD from individual participant to social world.

2. INDIVIDUAL PARTICIPATION IN FOSSD PROJECTS

One of the most common questions about FOSSD projects to date is why will software developers join and participate in such efforts, often without pay for sustained periods of time. Sometimes they may simply see their effort as something that is fun, personally rewarding, or provides a venue where they can exercise and improve their technical skill or competence in a manner that may not be possible within their current job or line of work [7,60]. However, people who participate, contribute, and join FOSS projects tend to act in ways where building trust and reputation, achieving "geek fame", being creative, advancing through evermore challenging technical roles [31], as well as giving and being generous with one's time, expertise, and source code [3] are valued traits.

Many FOSS developers participate in and contribute to multiple FOSSD projects, possibly in different roles. In one study, 5% of developers surveyed reported participating in 10 or more FOSSD projects [24]. Administrators of FOSS project Web sites and source code repositories also serve as gatekeepers in the choices they make for what information to post, when and where within the site to post it, as well as what not to post. Similarly, they may choose to create a site map that constitutes a classification of site and domain content, as well as outlining community structure and boundaries. Most frequently, participants in FOSS projects engage in online discussion forums or threaded email messages as a central way to observe, participate in, and contribute to public discussions of topics of interest to ongoing project participants [64].

The vast majority of source code that becomes part of FOSS released by a project is typically developed by a small group of core developers who control the architecture and direction of development [cf. 41]. Subsequently, most participants typically contribute to just a single module, though some modules may be include patches or modifications contributed by hundreds of contributors.

3. RESOURCES AND CAPABILITIES SUPPORTING FOSSD

What kinds of resources or development capabilities are needed to help make FOSS efforts more likely to succeed? Based on what has been observed and reported across many empirical studies of FOSSD projects, the following kinds of socio-technical resources enable the development of both FOSS software and ongoing project that is sustaining its evolution, application and refinement, though other kinds of resources may also be involved [51,52,53].

3.1 Personal software development resources

FOSS developers, end-users, and other volunteers often provide their own personal computing resources in order to access or participate in a FOSS development project. They similarly provide their own access to the Internet, and may even host personal Web sites or information repositories. Furthermore, FOSS developers bring their own choice of tools and development methods to a project. Sustained commitment of personal resources helps *subsidize* the emergence and evolution of the ongoing project, its shared (public) information artifacts, and resulting open source code. It spreads the cost for creating and maintaining the information infrastructure of the virtual organization that constitute a FOSSD project [7,12,44].

3.2 Beliefs supporting FOSSD

Why do software developers and others contribute their skill, time, and effort to the development of FOSS and related information resources? Though there are probably many diverse answers to such a question, it seems that one such answer must account for the belief in the freedom to access, study, modify, redistribute and share the evolving results from a FOSS development project. Without such belief, it seems unlikely that there could be "free" and "open source" software development projects [10,11,20]. However, one important consideration that follows is what the consequences from such belief are, and how these consequences are put into action.

A longitudinal study of the free software project GNUenterprise.org [12,13] identified many kinds of beliefs, values, and social norms that shaped actions taken and choices made in the development of the free GNUe software. Primary among them were *freedom of expression* and *freedom of choice*. Neither of these freedoms is explicitly declared, assured, or protected by free software copyright or commons-based intellectual property rights, or end-user license agreements (EULAs). However, they are central tenets free or open source modes of production and culture [2, Ghosh 2005]. In particular, in FOSS projects, these additional freedoms are expressed in choices for what to develop or work on (e.g., choice of work subject or personal interest over work assignment), how to develop it (choice of method to use instead of a corporate standard), and what tools to employ (choice over which personal tools to employ versus only using what is provided). They also are expressed in choices for when to release work products (choice of satisfaction of work quality over schedule), determining what to review and when (modulated by ongoing project ownership responsibility), and expressing what can be said to whom with or without reservation (modulated by trust and accountability mechanisms). Shared belief and practice in these freedoms of expression and choice are part of the virtual organizational culture that characterizes a FOSSD project [13]. Subsequently, putting these beliefs and cultural resources into action continues to build and reproduce socio-technical interactions networks that enabled sustained FOSSD projects and free software.

3.3 FOSSD informalisms

Software informalisms [51] are the information resources and artifacts that participants use to describe, proscribe, or prescribe what's happening in a FOSSD project. They are informal narrative resources that are comparatively easy to use, and publicly accessible to those who want to join the project, or just browse around. Subsequently, [51] and others [34] demonstrate how software informalisms can take the place of formalisms, like "requirement specifications" or software design notations which are seen as necessary to develop high quality software according to the SE community [cf. 59]. Yet these software informalisms often capture the detailed rationale and debates for why changes were made in particular development activities, artifacts, or source code files.

The most common types of informalisms used in FOSSD projects include (i) communications and messages within project Email lists, (ii) threaded message discussion forums, bulletin boards, or group blogs, (iii) news postings, (iv) project digests, and (v) instant messaging or Internet relay chat. They also include (vi) scenarios of usage as linked Web pages, (vii) how-to guides, (viii) to-do lists, (ix) FAQs, and other itemized lists, and (x) project Wikis, as well as (xi) traditional system documentation and (xii) external publications. FOSS (xiii) project property licenses are documents that also help to define what software or related project content are protected resources that can subsequently be shared, examined, modified, and redistributed. Finally, (xiv) open software architecture diagrams, (xv) intra-application functionality realized via scripting languages like Perl and PhP, and the ability to either (xvi) incorporate plug-in externally developer software modules, or (xvii) integrate software components, modules, or scripts from other OSSD efforts, are all resources that are used as needed according to the interests or actions of project participants.

All of the software informalisms are found or accessed from (xviii) project related Web sites or portals. These Web environments where most FOSS software informalisms can be found, accessed, studied, modified, and redistributed [51]. A Web presence helps make visible the project's information infrastructure and the array of information resources that populate it. These include (xix) FOSSD multi-project Web sites (e.g., SourgeForge.net, Savanah.org, Freshment.org, Tigris.org, Apache.org, Mozilla.org), community software Web sites (PhP-Nuke.org), and (xx) project-specific Web sites (e.g., www.GNUenterprise.org), as well as (xxi) embedded project source code Webs (directories), (xxii) project repositories (CVS [19]), and (xxii) software bug reports and (xxiv) issue tracking database like Bugzilla (see hwww.bugzilla.org/).

Together, these 20+ types of software informalisms constitute a substantial yet continually evolving web of informal, semi-structured, or processable information resources. This web results from the hyperlinking and cross-referencing that interrelate the contents of different informalisms together. Subsequently, these FOSS informalisms are produced, used, consumed, or reused within and across FOSS development projects.

3.4 Competently skilled, self-organizing, and self-managed FOSS developers

Developing complex software modules for FOSS applications requires skill and expertise in a target application domain. For example, contributing to a FOSSD project like Filezilla, requires knowledge and skill in handling file transfer conditions, events, and protocols. Developing FOSS modules or applications in a way that enables an open architecture requires a base of prior experience in constructing open systems. The skilled use of project management tools for tracking and resolving open issues, and also for bug reports contribute to the development of such system architecture. These are among the valuable professional skills that are mobilized, brought to, or drawn to FOSS development projects [cf. 6,7]. These skills are resources that FOSS developers bring to their projects.

FOSS developers organize their work as a virtual organizational form that seems to differ from what is common to in-house, centrally managed software development projects, which are commonly assumed in traditional SE textbooks. In the decentralized virtual organization of a large ongoing FOSSD project like the Apache.org or Mozilla.org, a skill-based meritocracy appears [18]. While there may be few explicit rules about what development tasks should be performed, who should perform, when, why, or how, this is not to say there are no rules that serve to govern the project or collective action within it.

The rules of governance and control are informally articulated but readily recognized by project participants. These rules serve to control the rights and privileges that developers share or delegate to one another in areas such as who can commit source code to the project's shared repository for release and redistribution [cf. 19,20]. Similarly, rules of control are expressed and incorporated into the open source code itself in terms of how, where, and when to access system-managed data via application program interfaces, end-user interfaces, or other features or depictions of overall system architecture. But these rules may and do get changed through ongoing project development. Subsequently, FOSS project participants self-organize around the expertise, reputation, and accomplishments of core developers, secondary contributors, and tertiary reviewers, as well as other peripheral volunteers.

3.5 Discretionary time and effort of FOSS developers

Are FOSS developers working for "free" or for advancing their career and professional development? There are many personal and professional career oriented reasons for why participants will contribute their time and effort to the sometimes difficult and demanding tasks of software development. These include self-determination, peer recognition, project affiliation or identification, and self-promotion, as well as belief in the inherent value of free software.

In the practice of self-determination, no one has the administrative authority to tell a project member what to do, when, how, or why. FOSS developers can choose to work on what interests them personally.

In the practice of peer recognition, a developer becomes recognized as an increasingly valued project contributor as a growing number of their contributions make their way into the core software modules. Also, project contributors who span multiple FOSS project communities serve as "social gateways" that increase the ongoing project's social mass [cf. 40], as well as affording opportunities for inter-project software composition and interoperation [30].

In self-promotion, project participants communicate and share their experiences, perhaps from other application domains or work situations, about how to accomplish some task, or how to develop and advance through one's career. Being able to move from the project periphery towards the center or core of the

development effort requires not only the time and effort of a contributor, but also the ability to communicate, learn from, and convince others as to the value or significance of the contributions.

The last source of discretionary time and effort that has been reported is found in the freedoms and beliefs in FOSSD that are shared, reiterated and put into observable interactions. If a project participant fails to sustain or reiterate the freedoms and beliefs codified in the GPL, then it is likely the person's technical choice in the project may be called into question [13], or the person will leave the project.

3.6 Trust and social accountability mechanisms

Developing complex FOSS source code and applications requires trust and accountability among project participants. Though trust and accountability in a FOSSD project may be invisible resources, ongoing software and project development work occur only when these intangible resources and mechanisms for social control are present [cf. Gallivan 2001, Hertzum, *et al*. 2002].

These intangible resources (or "social capital") arise in many forms. They include (a) assuming ownership or responsibility of a FOSSD project software module, (b) voting on the approval of individual action or contribution to ongoing project software [18], (c) shared peer reviewing [2, 10, 11], and (d) contributing gifts [3] that are reusable and modifiable common goods. They also exist through the project's recognition of a core developer's status, reputation, and geek fame. Without these attributions, FOSS developers may lack the credibility they need to bring conflicts over how best to proceed to some accommodating resolution. Finally, as a FOSSD project grows in terms of the number of contributing developers, end-users, and external sponsors, then project's socio-technical mass (i.e., web of interacting resources) becomes sufficient to insure that individual trust and accountability to the project are sustained and evolving, thus enabling social networking externalities [40].

4. COOPERATION, COORDINATION, AND CONTROL IN FOSSD PROJECTS

Conflicts arise in both FOSSD and SE projects. Finding ways to prevent, mitigate, or resolve conflicts becomes part of the cost (in terms of social capital) that must be incurred by FOSS developers for progress to occur. Minimizing the occurrence, duration, and invested effort in such conflicts quickly becomes a goal for the core developers in an FOSSD project. Similarly, finding tools and project organizational forms that minimize or mitigate recurring types of conflicts also becomes a goal for experienced core developers.

Software version control, as part of a software configuration management activity, is a recurring situation that requires coordination but enables stabilization and synchronization of dispersed and somewhat invisible development work [Grinter 1996]. Tools like CVS, Subversion, Git, and others are being used in FOSSD projects as both (a) a centralized mechanism for coordinating and synchronizing FOSS development, as well as (b) an online venue for mediating control over what software enhancements, extensions, or architectural revisions will be checked-in and made available for check-out throughout the decentralized project as part of the publicly released version [cf. 47].

FOSSD efforts rely on mechanisms and conditions for gentle but sufficient social control that helps constrain the overall complexity of the project. These constraints act in lieu of an explicit administrative authority or software project management regime that would schedule, budget, staff, and control the project's development trajectory with varying degrees of administrative authority and technical competence [cf. 59].

Each project team, or CVS repository administrator in it, must decide what can be checked in, and who will or will not be able to check-in new or modified software source code content. Sometimes these policies are made explicit through a voting scheme [18], or by reference to coding or data representation standards [28], while in others they are left informal, implicit, and subject to negotiation as needed. In either situation, version updates must be coordinated in order for a new system build and release to take place. Subsequently, those developers who want to submit updates to the project's shared repository rely extensively on online discussions that are supported using "lean media" such as threaded messages (via discussion forum, bulletin board, or similar) posted on a Web site [64], rather than through onerous system configuration control boards. Thus, software version control, system build and release is a coordination and control process mediated by the joint use of versioning, system building, and communication tools [14].

In a FOSSD project meritocracy, software development work appears to be logically centralized, while being physically distributed in an autonomous and decentralized manner [44 However, it is neither simply a "cathedral" or a "bazaar", as these terms have been used to describe alternative ways of organizing FOSSD projects. Instead, when meritocracy operates as a virtual enterprise, it relies on *virtual project management* (VPM) to mobilize, coordinate, control, build, and assure the quality of FOSS development activities [52]. It may invite or encourage system contributors to come forward and take a shared, individual responsibility that will serve to benefit the FOSS collective of user-developers. VPM requires multiple people to act in the roles of team leader, sub-system manager, or system module owner in a manner that may be short-term or long-term, based on their skill, accomplishments, availability and belief in ongoing project development.

Thus, FOSSD efforts rely on mechanisms and conditions for gentle but sufficient social control that helps constrain the overall complexity of the project. These constraints act in lieu of an explicit administrative authority or software project management regime that would schedule, budget, staff, and control the project's development trajectory with varying degrees of administrative authority and technical competence [cf. 59].

5. ALLIANCE FORMATION, INTER-PROJECT SOCIAL NETWORKING AND COMMUNITY DEVELOPMENT

How does the gathering of FOSS developers give rise to a more persistent self-sustaining organization or project community? Through choices that developers make for their participation and contribution to a FOSSD 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 FOSSD Web sites and online discourse in software informalisms (e.g., threaded discussions

on bulletin boards), and they find they share many technical competencies, values, and beliefs in common.

Becoming a central node in a social network of software developers that interconnects multiple FOSS projects is also a way to accumulate social capital and recognition from peers. However, it also enables the merger of independent FOSS systems into larger composite ones that gain the critical mass of core developers to grow more substantially and attract ever larger user-developer communities [39,53]. "Linchpin developers" [39] participate in or span multiple FOSSD projects. In so doing, they create alliances between otherwise independent FOSSD projects.

Sharing beliefs, values, communications, artifacts and tools among FOSS developers enables not only cooperation, but also provides a basis for shared experience, camaraderie, and learning [cf. 15,27,34,35]. FOSS developers most often participate and contribute by choice, rather than by assignment, since they find that conventional software development work provides the experience of working with others who are assigned to a development effort, whether or not they find that share technical approaches, skills, competencies, beliefs or values. As a result, FOSS developers find they get to work with people that share their many values and beliefs in common, at least as far as software development.

Multi-project clustering and interconnection enables small FOSS projects to come together as a larger social network with the critical mass [40] needed for their independent systems to be merged and experience more growth in size, functionality, and user base. It also enables shared architectural dependencies to arise (perhaps unintentionally) in the software components or sub-systems that are used/reused across projects [cf. 9,47]. FOSSD Web sites also serve as hubs for distributed cognition that centralize attention for what is happening with the development of the focal FOSS system, its status, participants and contributors, discourse on pending/future needs, etc.

The values and beliefs associated with free software or open source software are both signaled and institutionalized in the choice of intellectual property licenses (e.g., GPL) that FOSSD projects adopt and advocate. These licenses in turn help establish norms for developing free software or open source software, as well as for an alliance with other FOSSD projects that use the same licenses.

Almost half of the over 150K FOSS projects registered at SourceForce.net Web portal (as of July 2007) employ the GNU General Public License (GPL) for free/libre software. The GPL seeks to preserve and reiterate the beliefs and practices of sharing, examining, modifying and redistributing FOSS systems and assets as common property rights for collective freedom [2]. A few large FOSSD projects seek to further protect the collective free/open intellectual property rights through the formation of legally constituted non-profit organizations or foundations (e.g., Free Software Foundation, Apache Software Foundation, GNOME Foundation) [45]. Other OSS projects, because of the co-mingling of assets that were not created as free property, have adopted variants that relax or strengthen the rights and conditions laid out in the GPL. Dozens of these licenses now exist, with new ones continuing to appear (cf. www.opensource.org). Finally, when OSSD projects seek to engage or receive corporate sponsorship, and the possible co-mingling of corporate/proprietary intellectual property, then some variation of a non-GPL open source license is employed, as a way to signal a "business friendly" OSSD project, and thus to encourage participation by developers who want to work in such a business friendly and career enhancing project [23,62].

Community building, alliance forming, and participatory contributing are essential and recurring activities that enable FOSSD projects to persist without central corporate authority. Thus, linking people, systems, and projects together through shared artifacts and sustained online discourse enables a sustained social network [38,39] and socio-technical community, Web-based information infrastructure [30], and network of alliances [28,34,42] to emerge.

6. FOSS AS A MULTI-PROJECT SOFTWARE ECOSYSTEM

As noted above, many FOSSD 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 FOSS system gives rise to propagated effects, architectural dependencies, or vulnerabilities in one or more of the projects linked to it [4, 30].

Interdependencies within a FOSS ecosystem are most apparent when FOSSD projects share source code modules, components, or sub-systems, as well as common (linchpin) developers. In such situations, the volume of source code of an individual FOSSD project may appear to grow at a super-linear or exponential rate [53,56] when modules, components, or sub-systems are integrated in whole into an existing FOSS system [53]. Such system growth patterns therefore seem to challenge the well-established laws of software evolution [36,37]. Thus, software evolution in a multi-project FOSS ecosystem is a process of co-evolution of interrelated and interdependent FOSSD projects, people, artifacts, tools, code, and project-specific processes [e.g., 4,43,61,65].

It seems reasonable to observe that the world FOSSD is not the only place where multi-project software ecosystems emerge, as software sharing or reuse within traditional software development enterprises is common [29,48]. However, the process of the co-evolution of software ecosystems found in either traditional or FOSSD projects in mostly unknown, though some study has begun [48]. Thus, co-evolution of interdependent software systems and standards for interoperation within an FOSS ecosystem represents an opportunity for research that investigates understanding such a software evolution process through studies supported by modeling and simulation techniques [1,30,55].

Overall, FOSS systems co-evolve with their development communities. This means the evolution of one depends on the evolution of the other. Said differently, a FOSS project with a small number of developers (most typically one) will not produce and sustain a viable system unless/until the team reaches a larger critical mass of 5-15 core developers. However, if and when critical mass is achieved, then it may be possible for the FOSS system to grow in size and complexity at a sustained exponential rate, defying the laws of software evolution that have held for decades [36,37,53]. Furthermore, user-developer communities co-evolve with their systems in a mutually dependent manner [13,43,45,51,65], and system architectures and functionality grow in discontinuous jumps as independent FOSS projects decide to join forces [e.g., 43,53]. Whether this trend is found in traditional or closed source software projects is unclear. But what these findings and trends do indicate is that

it appears that the practice of FOSS development processes is different from the processes traditionally advocated for SE.

7. FOSS AS A SOCIAL MOVEMENT

Social movements reflect sustained and recurring large-scale collective activities within a society. Social movements can be characterized by (a) their recurring structural forms (e.g., boundaries around movement sub-segments, multiple centers of activity, and social networks that link the segments and centers) and venues for action, (b) ideological beliefs, and (c) organizations whose purpose is to advance and mobilize broader interest in the movement [58]. The OSS movement arose in the 1990's [10,12,63] from the smaller, more fervent "free software" movement [Gay 2003] started in the mid 1980's.

The OSS movement is populated with thousands of OSS development projects, each with its own Web site. Whether the OSS movement is just another computerization movement [cf. 12,32], or is better recognized as a counter-movement to the proprietary or closed source world of commercial software development is unclear. For example, executives from proprietary software firms have asserted that that OSS (specifically that covered by the GNU Public License or "GPL") is a cancer that attaches itself to intellectual property [22]. However, other business sources seem to clearly disagree with such characterizations and see OSS as an area for strategic investment [21,46].

More than 150K projects are registered at OSS portals like SourceForge.org, while other OSS portals like Freshment.org, and Tigris.org contain thousands more. However, the vast majority of these OSS projects at SourceForge appear to be inactive, with less than two contributing developers, as well as no software available for download, evaluation, or enhancement. Nonetheless, at least a few thousand OSS projects seem to garner most of the attention and community participation, but no one project defines or leads the OSS movement. The Linux Kernel project is perhaps the most widely known OSS project, with its celebrity leaders, like Linus Torvalds. Ironically, The Linux Kernel is also the most studied OSS project. However, there is no basis to indicate that how things work in this project prescribe or predict what might be found in other successful OSS projects. Thus, the OSS movement is segmented about the boundaries of each OSS project, though some of the larger project communities have emerged as a result of smaller OSS projects coming together.

In contrast to the OSS movement, Richard M. Stallman initiated the free software movement [cf. 12,13]. Its participants or advocates identify their affiliation and commitment by openly developing and sharing their software following the digital civil liberties expressed in the GPL. The GPL is a license agreement that promotes and protects software source code using the GPL copyright to always be available (always assuring a "copy left"), that the code is open for study, modification, and redistribution, with these rights preserved indefinitely. Furthermore, any software system that incorporates or integrates free software covered by the GPL, is asserted henceforth to also be treated as free software. This so-called "viral" nature of the GPL is seen by some to be an "anti-business" position, which is the most commonly cited reason for why other projects have since chose to identify them as open source software [Fink 2003]. However, new/pre-existing software that does not integrate GPL source code is not infected by the GPL, even if both kinds of software co-exist on the same computer or operating system, or that access one another through open or standards-based application program interfaces.

Overall, recognizing the free software and OSS have facilitated the emergence of global-scale social (or computerization) movements, indicates that FOSS is increasingly permeating society at an industrial, governmental, and international level, and is doing so in ways that no prior software technology or development method has come close to achieving. Why this has come about, what consequences it portends for the future of FOSS, and whether corporate or public (government) policy initiatives will increasingly address the development, adoption, deployment, usage, and support of FOSS applications and projects, all require further study. But is also in clear that it is increasingly unlikely the any company, government, or nation can successfully inhibit the near-term and mid-term societal dispersion of FOSS or the FOSS movements.

8. DISCUSSION AND LIMITATIONS

One of the defining characteristics of data about the FOSSD projects is that in general is it publicly available on a global basis [25,39,40,53]. Data about FOSSD products, artifacts, and other resources is kept in repositories associated with a project's Web site [cf. 26]. This may include the site's content management system, computer mediated communication systems (email, persistent chat facilities, and discussion forums), software versioning or configuration management systems, and networked file systems. FOSSD process data is generally either extractable or derivable from data/content in these artifact repositories. First-person data may also be available to those who participate in a project, even if just to remotely observe ("lurk") or to electronically interview 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 SE research will be performed in the domain of FOSSD projects, rather than using traditional sources of data from in-house or proprietary software development projects that have constraints on access and publication. FOSSD process data collection from publicly accessible artifact repositories may also be found to be more cost-effective compared to studies of traditional closed-source, proprietary, and in-house software development repositories [cf. 5,26].

The modest sample of studies of FOSSD cited above is drawn from a larger set of studies reviewed elsewhere [54]. However, though this review does not examine the alternative research methods employed in different empirical studies of FOSSD, it is worth noting that such studies cover a wide range of empirical methods. These include reflective practice and industry polls, systematic surveys, ethnographic studies, mining FOSSD artifact or informalism repositories, and multi-modal modeling and analysis of FOSSD socio-technical interaction networks [cf. 50,55].

FOSSD is certainly not a panacea for developing complex software systems, nor is it simply 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. Thus, we should be able to help see these limits as manifest within the level of analysis or research for empirical FOSSD studies examined above.

First, in terms of participating, joining, and contributing to FOSS projects, an individual developer's interest, motivation,

and commitment to a project and its contributors is dynamic and not indefinite [cf. 49]. 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, FOSS 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 FOSS 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 FOSS projects. The emergence of non-profit foundations that were established to protect the property rights of large multi-component FOSS project creates a demand to sustain and protect such foundations. If a foundation becomes too bureaucratic as a result to streamline its operations, then this may drive contributors away from a project. So, these foundations need to stay lean, and not become a source of occupational careers, in order to survive and evolve. Similarly, as FOSS 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 FOSS project contributors in roles above and beyond those involved in enhancing the source code of a FOSS project. FOSS 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 FOSS project community and its alliances will begin to come apart.

Fourth, in terms of the co-evolution of FOSS systems and community, as already noted, 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 FOSS. 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 FOSS 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 [26] and multi-modal modeling and analysis of the socio-technical processes and networks found in sustained FOSSD projects [50,55] 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.

9. 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. Some of these opportunities follow. However, it is perhaps surprising to observe that it is unclear whether the world of FOSSD is interested in adopting the best practices found in the world of SE, since after all, why would software developers seek out to engage in FOSSD projects and practices is they were the same or less than those found in SE? As such, let us consider the opportunities for what FOSSD might contribute to the world SE.

First, FOSS 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 thus poised to alter the calculus of empirical SE [5,25,53]. Software process discovery, modeling, and simulation research [e.g., 55] 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) within or across FOSS project repositories [26] to support its visualization, restructuring/refactoring, or redesign is likely to 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., 57] compared to the online peer reviews 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 simple technical roles and motivations driven by financial compensation, technical education, and seek the quality assuring rigor that purportedly follows from the use of formal notations and analytical schemes. Said simply, is FOSSD more fun, more interesting, and more rewarding than SE?

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, 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 in FOSSD projects, it appears that virtual project management and 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 bee 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.

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. Nearly 1000 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 brought to bear and to what ends in different OSS projects.

10. CONCLUSIONS

Free and open source software development is emerging as an alternative approach for how to develop large software systems. FOSSD employs new types and new kinds of socio-technical work practices, development processes, and community networking when compared to those found in industrial software projects, and those portrayed in software engineering textbooks [59]. 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 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 a survey of empirical studies of FOSSD projects and other analyses presented in this article, 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 research or apply FOSSD concepts, techniques, or tools can be advanced through studies that examine FOSSD activities, artifacts, and projects.

11. ACKNOWLEDGMENTS

The research described in this paper has been supported by grants #0083075, #0205679, #0205724, #0350754, and #0534771 from the U.S. National Science Foundation. No endorsement implied.

12. REFERENCES

[1] Antoniades, I.P., Samoladas, I., Stamelos, I., Angelis, L., and Bleris, G.L., Dynamic Simulation Models of the Open Source Development Process, in [32], 174-202, 2005.

[2] Benkler, Y. *The Wealth of Networks: How Social Production Transforms Markets and Freedom*, Yale University Press, New Haven, CT, 2006.

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

[4] Capaluppi, A. and Michlmayr, M., From the Cathedral to the Bazaar: An Empirical Study of the Lifecycle of Volunteer Community Projects, in [OSS07], 31-44, 2007.

[5] 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.

[6] Crowston, K. and Howison, J., Hierarchy and centralization in free and open source software team communications, *Knowledge Technology & Policy,* 18(4), Winter, 65-85, 2006.

[7] 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.

[8] Damiani, E., Fitzgerald, B., Scacchi, W., Scotto, M. and Succi, G., (Eds.), *Open Source Systems*, IFIP Vol. 203, Springer, Boston, 2006.

[9] De Souza, C. R. B., Froehlich, J., and Dourish, P., Seeking the Source: Software Source Code as a Social and Technical Artifact. *Proc. ACM Intern. Conf. Supporting Group Work (GROUP 2005)*, Sanibel Island, Florida, 197-206, 2005.

[10] DiBona, C., Cooper, D., and Stone, M., *Open Sources 2.0*, O'Reilly Media, Sebastopol, CA. 2005.

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

[12] Elliott, M.S., Examining The Success of Computerization Movements in the Ubiquitous Computing Era: Free and Open Source Software Movements, in [33], 2008.

[13] Elliott, M. and Scacchi, W., Free Software Development: Cooperation and Conflict in A Virtual Organizational Culture, in [32], 152-172, 2005.

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

[15] 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*, Barcelona, Spain, December. 425-433, 2002.

[16] Feller, J., Fitzgerald, B., Hissam, S. and Lakhani, K. (Eds.), *Perspectives on Free and Open Source Software*, MIT Press, Cambridge, MA, 2005.

[17] Feller, J., Fitzgerald, B., Scacchi, W., and Sillitti, A., (Eds.), *Open Source Development, Adoption and Innovation*, IFIP Vol. 234, Springer, Boston, 2007.

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

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

[20] Fogel, K., *Producing Open Source Software: How to Run a Successful Free Software Project*, O'Reilly Press, Sebastopol, CA, 2005.

[21] Goldman, R. and Gabriel, R.P., *Innovation Happens Elsewhere: Open Source as Business Strategy*, Morgan Kaufmann Publishers, San Francisco, CA, 2005.

[22] Greene, T.C., Ballmer: "Linux is a Cancer", *The Register*, http://www.theregister.co.uk/2001/06/02/ballmer_linux_is_a_cancer/, 2 June 2001.

[23] 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, 2002.

[24] Hars, A. and Ou, S., Working for Free? Motivations for participating in open source projects, *Intern. J. Electronic Commerce*, 6(3), 25-39, 2002.

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

[26] 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.

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

[28] Iannacci, F. Beyond Markets and Firms: The Emergence of Open Source Networks, *First Monday*, 10(5), 2005.

[29] Jaaski, A. Experiences on Product Development with Open Source Software, in [OSS07], 85-96. 2007.

[30] Jensen, C. and Scacchi, W., Process Modeling Across the Web Information Infrastructure, *Software Process — Improvement and Practice*, 10(3), 255-272, 2005.

[31] Jensen, C. and Scacchi, W., Role migration and advancement processes in OSSD projects: A comparative case study, in *Proc. 29th Intern. Conf. Soft. Eng.*, ACM, Minneapolis, MN, 364-374, 2007.

[32] Koch, S. (Ed.), *Free/Open Source Software Development*, IGI Publishing, Hershey, PA, 2005.

[33] Kraemer, K.L. and Elliott, M. (Eds.), *Computerization Movements and Technology Diffusion: From Mainframes to Ubiquitous Computing*, Information Today, Inc., to appear, 2008.

[34] Lanzara, G.F. and Morner, M., Artifacts rule! How organizing happens in open source software projects, in B. Czarniawska and T. Hernes (Eds.), *Actor-Network Theory and Organizing*, Liber & Copenhagen Business School Press, Malmo, Sweden, 197-206, 2005.

[35] Lave, J. and Wenger, E., *Situated Learning: Legitimate Peripheral Participation*, Cambridge University Press, Cambridge, UK, 1991.

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

[37] Lehman, M.M., Software Evolution and Software Evolution Processes, *Annals of Software Engineering*, 12, 275-309, 2002.

[38] Lopez-Fernandez, L., Robles, G., Gonzalez-Barahona, J.M., and Herraiz, I., Applying Social Network Analysis to Community-Driven Libre Software Projects, *Intern. J. Info. Tech. and Web Engineering*, 1(3), 27-28, 2006.

[39] Madey, G., Freeh, V., and Tynan, R., Modeling the F/OSS Community: A Quantitative Investigation, in [32], 203-221, 2005.

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

[41] Mockus, A., Fielding, R., & Herbsleb, J.D., Two Case Studies of Open Source Software Development: Apache and Mozilla, *ACM Trans. Soft. Eng. Meth.*, 11(3), 309-346, 2002.

[42] 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.

[43] 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.

[44] Noll, J. and Scacchi, W., Supporting Software Development in Virtual Enterprises, *J. Digital Information*, 1(4), February, http://jodi.tamu.edu/Articles/v01/i04/Noll/, 1999.

[45] O'Mahony, S. Guarding the Commons: How Community Managed Software Projects Protect their Work, *Research Policy* 32(7), 1179-1198, 2003.

[46] OSBC, *Open Source Business Conference*, http://www.osbc.com, (accessed 15 July 2006, 30 June 2007).

[47] Ovaska, P., Rossi, M. and Marttiin, P. Architecture as a Coordination Tool in Multi-Site Software Development, *Software Process—Improvement and Practice*, 8(3), 233-247, 2003.

[48] Robles, G., Duenas, S., and Gonzalez-Baharona, J.M., Corporate Involvement in Libre Software: Study of Presence in Debian Code over Time, in [OSS07], 121-132, 2007.

[49] Robles, G. and Gonzalez-Baharona, J.M., Contributor Turnover in Libre Software Projects, in [8], 273-286, 2006.

[50] 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.

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

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

[53] 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.

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

[55] 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.

[56] 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, 2002.

[57] 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.

[58] Snow, D.A., Soule, S.A., and Kriesi, H., *The Blackwell Companion to Social Movements*, Blackwell Publishers Ltd., Victoria, Australia, 2004.

[59] Sommerville, I., *Software Engineering, 7th Edition*, Addison-Wesley, New York, 2004.

[60] 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, 2003.

[61] Weiss, M., Moroiu, G. and Zhao, P., Evolution of Open Source Communities, in [8], 21-32, 2006.

[62] West, J. and O'Mahony, S., Contrasting Community Building in Sponsored and Community Founded Open Source Projects, *Proc. 38th. Hawaii Intern. Conf. Systems Sciences*, Waikola Village, HI, 2005.

[63] West, J. and Dedrick, J., The Effect of Computerization Movements Upon Organizational Adoption of Open Source, in [33], 2008.

[64] 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), Philadelphia, PA, ACM Press, 329-338, 2000.

[65] Ye, Y., Nakajoki, K., Yamamoto, Y., and Kishida, K., The Co-Evolution of Systems and Communities in Free and Open Source Software Development, in [32], 59-82. 2005.