Collaboration Practices and Affordances in Free/Open Source Software Development

Walt Scacchi
Institute for Software Research
University of California, Irvine
Irvine, CA 92697-3455 USA

Revised version to appear in I. Mistrik et al. (Eds.) *Collaborative Software Engineering*, Springer-Verlag, Berlin Heidelberg 2010

Abstract. This chapter examines collaborative work practices, development processes, project and community dynamics, and other socio-technical relationships in free and open source software development (FOSSD). It also describes what kinds of collaboration affordances facilitate collaborative work in FOSSD projects. It reviews a set of empirical studies of FOSSD that articulate different levels of analysis. Finally, there is discussion of limitations and constraints in understanding what collaboration practices and affordances arise in FOSSD studies and how they work, and then to emerging opportunities for future FOSSD studies.

Keywords. Open source software, empirical studies, collaboration practices, affordances, socio-technical relationships

15.1 Introduction

This chapter examines and compares collaborative work practices, processes, and affordances 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) [63]. Hundreds 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 collaborative FOSSD processes used to build and sustain these projects, and how might differences with SE be employed to explain what's going on with FOSSD?

One of the more significant features of FOSSD is the formation and enactment of collaborative software development practices and 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, FOSS developers are generally expected (or prefer) to provide their own computing resources (e.g., laptop computers on the go, or desktop computers at home), and bring their own software tools with them.

FOSS developers often work on global 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 [cf. 29]. Does the absence or limited presence of corporate authorities or sponsors encourage or facilitate collaboration in FOSSD, or do collaborative practices and affordances supporting FOSSD reduce the need to rely on traditional corporate authority or project management regimes? Is collaborative practice a defining feature of FOSSD, is FOSSD a causal attribute of collaboration, or does collaborative practice more readily produce FOSS? What motivates software developers participate in FOSSD projects? Is volunteerism and personal discretion key to collaboration in FOSSD projects? Why and how are large FOSSD projects sustained through collaborative practices and affordances? How are large FOSSD projects coordinated, controlled or managed without a traditional project management team? Why and how might answers to these questions change over time? These are the kinds of questions addressed in this chapter.

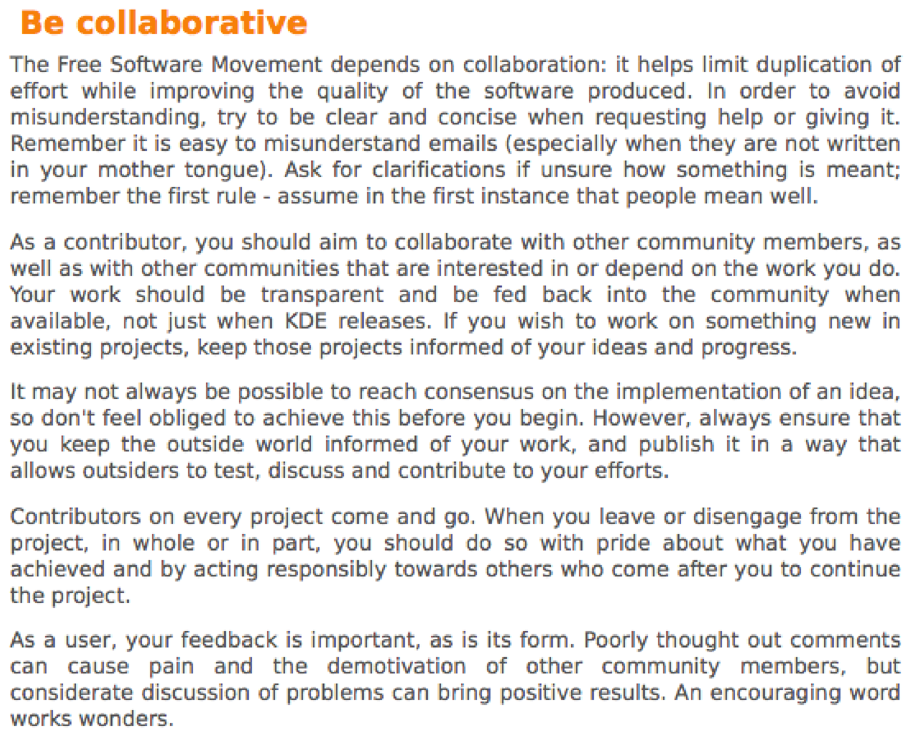
15.1.1 What is free/open source software development?

FOSSD is mostly not about SE, at least not as SE is portrayed in modern SE textbooks [cf. 63]. FOSSD is also 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, and development artifacts are publicly available over the Web. Furthermore, substantial FOSSD effort is directed at enabling and facilitating collaboration among developers (and also end-users), but generally there is no traditional SE project management regime, budget or schedule. FOSSD is also oriented towards the joint development of an ongoing community of developers and users concomitant with the FOSS system of interest.

FOSS developers are typically end-users of the FOSS they develop [57, 58, 65, 69] and other end-users often participate in and contribute to FOSSD efforts as non-developers. 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 [44]. Thus, it is reasonable to assume that FOSSD processes are not necessarily of the same type, kind, or form found in modern SE projects [63]. Subsequently, what is known about collaborative SE processes may not be equally applicable to FOSSD practices without some explicit empirical justification. Thus, it is appropriate to review what is known about FOSSD and where collaboration practices and affordances emerge along the way.

15.1.2 What are affordances supporting collaborative software development?

Affordances refers to situated, interactional properties between objects and actors that facilitate certain kinds of social interactions in a complex environment. The concept of affordances appears in the studies that employ the construct to characterize aspects of complex work settings that facilitate how people interact though computing systems [1, 47]. Computer-supported work environments, when effective, afford new ways and means for collaborative learning [36]. Subsequently, the focus in this chapter is on the interactions that facilitate collaborative activities between FOSS developers who are geographically dispersed but share access to online artifacts, networked information repositories and communication infrastructures, such as Web pages, Web sites, source code version servers, distributed file servers, virtual private networks, and the like. Consider the example in Fig. 1, a screenshot of an excerpt from the "Code of Conduct" that helps inform participants and communicate social norms on how to "be collaborative" in the K Development Environment (KDE) FOSS project.



Be collaborative

The Free Software Movement depends on collaboration: it helps limit duplication of effort while improving the quality of the software produced. In order to avoid misunderstanding, try to be clear and concise when requesting help or giving it. Remember it is easy to misunderstand emails (especially when they are not written in your mother tongue). Ask for clarifications if unsure how something is meant; remember the first rule - assume in the first instance that people mean well.

As a contributor, you should aim to collaborate with other community members, as well as with other communities that are interested in or depend on the work you do. Your work should be transparent and be fed back into the community when available, not just when KDE releases. If you wish to work on something new in existing projects, keep those projects informed of your ideas and progress.

It may not always be possible to reach consensus on the implementation of an idea, so don't feel obliged to achieve this before you begin. However, always ensure that you keep the outside world informed of your work, and publish it in a way that allows outsiders to test, discuss and contribute to your efforts.

Contributors on every project come and go. When you leave or disengage from the project, in whole or in part, you should do so with pride about what you have achieved and by acting responsibly towards others who come after you to continue the project.

As a user, your feedback is important, as is its form. Poorly thought out comments can cause pain and the demotivation of other community members, but considerate discussion of problems can bring positive results. An encouraging word works wonders.

Fig. 1. An excerpt from a FOSSD project Web page that both encourages and guides how and why project participants can collaborate. Source: http://www.kde.org/code-of-conduct/, accessed October 2008.

This collaboration affordance includes a narrative inscription on a KDE project Web page (an object in a complex online environment) that encourages and guides project participants (actors--developers or users of KDE) for how to collaborate (via certain kinds of social interaction) in the KDE project. Collaboration affordances in FOSSD may emerge in online venues and workspaces for FOSSD work, and may differ by the kind or type of FOSS being developed (e.g., operating system utility program versus network computer game), the project web site or multi-project Web portals in use, as well as by the infrastructure of online tools participants use in FOSSD work.

What makes software development in general, or FOSSD in particular, collaborative? Is collaborative software development work natural and obvious, or challenging, perplexing, and sometimes problematic? What can be done to facilitate or encourage opportunities to make software development work more collaborative, or even more fun and playful [cf. 46]? Do all multi-user software development tools, interfaces, or repositories automatically enable collaboration, or are some more effective than others? Questions like these help ground our interest in reviewing what kinds of affordances are found in empirical studies of FOSSD work, and how they facilitate collaborative software development activities with online software artifacts.

15.1.3 Results from recent studies of FOSSD

The remainder of this chapter provides a review of empirical studies of FOSSD that articulate different levels of analysis, and each level is examined in a separate section. Emphasis is directed at identifying affordances that facilitate collaborative software development activities found in different studies of FOSSD participants, practices, and projects. Section 2 provides a brief background on what motivates people to participate and contribution to FOSSD projects. Section 3 examines the different resources and capabilities that FOSS developers bring to their projects. Section 4 examines practices in cooperation, coordination, and control that arises within self-organizing FOSSD projects. Section 5 examines how multiple FOSSD projects give rise to alliances and inter-project network communities. Section 6 examines how clusters of diverse projects form FOSS ecosystems that can exhibit collective patterns of sustained exponential growth. Finally, there is a discussion of limitations and constraints in the FOSSD studies so far, followed by conclusions that highlight emerging opportunities for future studies of collaborative FOSSD work practices, development processes, information artifacts, and project communities.

15.2 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, as well as engage in sometime difficult and challenging technical work, often without pay, for sustained periods of time. Surveys of FOSS developers [e.g., 23, 27, 28] have posed and investigated such questions. There are complex motivations for why FOSS developers are willing to allocate their time, skill, and effort by joining a FOSS project [28, 66]. Some FOSS developers are motivated to see their contribution of time, effort, and code as gifts they provide to a project community [3]. Other motivations include a developer's ability to acquire skill and sustained experience from working in multiple or different roles [34, 53, 66]. It can also include a desire to work on software systems that the developer finds personally interesting, a desire to work with well-regarded FOSSD experts, or to be recognized by project peers as a valued and frequent contributor to a highly visible FOSS project [28, 25]. Similarly, it can be that the developer routinely uses the FOSS system of interest, and wants it to implement some additional feature or capability, or wants to reinvent processing capabilities found in other software systems, or to add innovative system features

[64, 65, 58]. These conditions represent different ways for how participants learn to collaboratively develop FOSS in different projects and different application domains.

Motivations for participating in FOSSD stand in contrast to the traditional view of software project management. Software project managers are suppose to design technical work activities in ways that are satisfying and thus motivating to developers [5]. Project managers are also responsible for insuring developers collaborate with one another when needed, and where developers are able to participate in setting project development goals and providing process feedback/improvement. Software project managers are expected to make SE work interesting, rewarding, and satisfying, and if they cannot do this, then the SE project may fail or produce low quality and hard to maintain software [5].

In contrast, the most frequently cited reason why software developers participate and contribute to FOSSD projects is *to learn* [23]. In other words, participating developers come to believe FOSSD projects of interest are expected to provide ways and means for individual and collaborative learning [cf. 14, 36]. Consequently, when developers no longer value or lose interest in what can be learned from a FOSSD project in which they participate, they may choose stop contributing to the project and move on. In traditional SE, project managers shape working conditions and thus the basis for collaborative work, while in FOSSD projects, individual participates must take responsibility for learning how to organize and manage themselves so as to fulfill their personal motivations when working with other FOSS developers currently participating in the projects. Thus, in this regard, the different ways and means for FOSS developers to learn things of greatest personal interest serve as individual level affordances for engaging in collaborative FOSSD project work. Conversely, developers who do not want to collaborate with the FOSS project developers at hand will not be able to realize or appropriate the common FOSSD collaborative learning affordances they find motivational, and thus they may move on to search for another project of interest.

15.3 Resources and Capabilities Supporting FOSSD

What kinds of resources or development capabilities are needed to help make FOSS efforts collaborative and successful? Based on what has been observed and reported across many empirical studies of FOSSD projects, the following kinds of socio-technical resources (or social capital) 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 [57, 59]. The following sub-sections examine collaborative practices and affordances centered on different resources and capabilities found in FOSSD projects.

15.3.1 Personal software development tools and networking support

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. It is not uncommon that a FOSS developer works on a project from a room at home, or on a laptop PC while traveling. FOSS developers bring their own choice of software development tools and methods to a project, and sometimes the number of tools employed ranges into dozens. The mobility of tools and laptop computers also enables the organization and enactment of collaborative FOSS *hackathons*[1]—marathon FOSS development experience involving dozens of developers at a chosen destination for the purpose of collaboratively analyzing, modifying, and rebuilding a given FOSS system. Participation in such events often entails travel and related expenses often borne out of pocket by each participant, though they also find such events personally and professionally rewarding, convivial, and fun, even though involving long hours of difficult and technically challenging work.

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, 48]. These in turn help create recognizable shares of the FOSS *commons* [2, 49, 50] that are linked (via hardware, software, Internet and Web) to the project's information infrastructure. So personal computers, FOSS tools, and hackathons are affordances that help enable collaborative FOSSD.

15.3.2 Beliefs supporting FOSS Development

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 [11, 12, 25]. However, it also includes *freedom of expression* and *freedom of choice* [18, 60]. Neither of these freedoms is explicitly declared, assured, or protected by copyright or commons-based intellectual property rights, nor by end-user license agreements. However, these freedoms are realized 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.

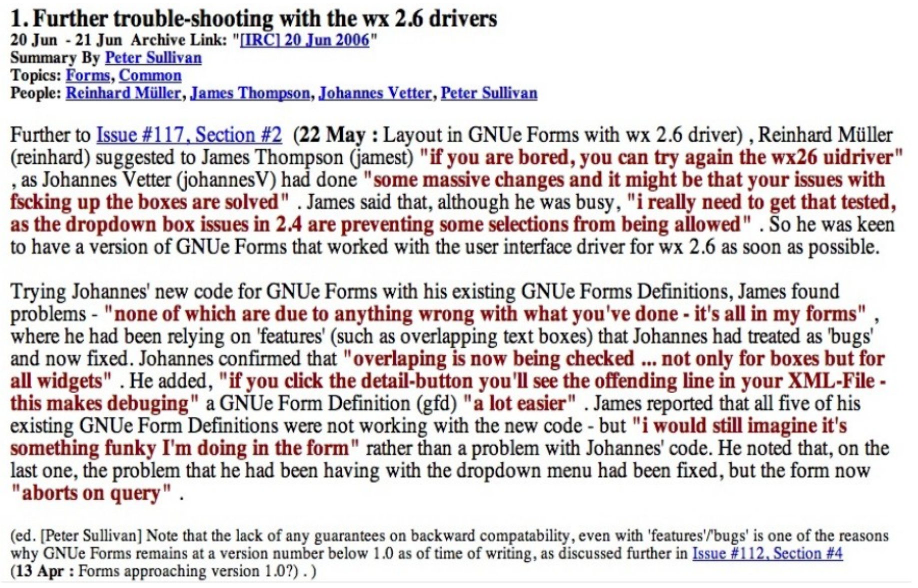
The enactment of beliefs, values, and norms for why and how to develop FOSSD, which constitute part of a FOSS developer's mental model [cf. 20], that are represented in FOSS licenses and project narratives (like Fig. 1), serve as affordances that enable collaborative FOSSD projects and teamwork. Similarly, failure to enact and sustain such beliefs can lead to participants being challenged by others regarding their commitment to collaboratively develop FOSS in a proper manner, so the absence or failure of such an affordance can drive FOSS developers apart [15, 17, 19].

[1] Description and examples of FOSS hackathons at http://en.wikipedia.org/wiki/Hackathon .

15.3.3 FOSSD informalisms

Software informalisms [57] are the information artifacts that participants use as resources to describe, proscribe, prescribe, or question 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. They are generally supported with lightweight tools [4, 68]. Nonetheless, these artifacts serve as both workspaces where collaborative FOSSD work activities (including reading, reviewing, writing, and learning) occurs, as well as the products of such collaborations [14, 16, 37, 54, 57].

The most common informalisms used in OSSD projects include (i) communications and messages within project Email [68], (ii) threaded message discussion forum, bulletin boards, or group blogs, (iii) news postings, and (iv) instant messaging or Internet relay chat. These enable developers and users to converse with one another in a lightweight, semi-structured manner, and now use of these tools is global across applications domains and cultures. As such, the discourse captured in these tools is a frequent source of OSS requirements. A handful of OSSD projects have found that summarizing these communications into (v) project digests [16] helps provide an overview of major development activities, problems, goals, or debates. These project digests represent multi-participant summaries that record and hyperlink the rationale accounting for focal project activities, development problems, current software quality status, and desired software functionality. Project digests (see Fig. 2) record the discussion, debate, consideration of alternatives, code


1. Further trouble-shooting with the wx 2.6 drivers
20 Jun - 21 Jun Archive Link: "[IRC] 20 Jun 2006"
Summary By Peter Sullivan
Topics: Forms, Common
People: Reinhard Müller, James Thompson, Johannes Vetter, Peter Sullivan

Further to Issue #117, Section #2 (**22 May :** Layout in GNUe Forms with wx 2.6 driver) , Reinhard Müller (reinhard) suggested to James Thompson (jamest) **"if you are bored, you can try again the wx26 uidriver"** , as Johannes Vetter (johannesV) had done **"some massive changes and it might be that your issues with fscking up the boxes are solved"** . James said that, although he was busy, **"i really need to get that tested, as the dropdown box issues in 2.4 are preventing some selections from being allowed"** . So he was keen to have a version of GNUe Forms that worked with the user interface driver for wx 2.6 as soon as possible.

Trying Johannes' new code for GNUe Forms with his existing GNUe Forms Definitions, James found problems - **"none of which are due to anything wrong with what you've done - it's all in my forms"** , where he had been relying on 'features' (such as overlapping text boxes) that Johannes had treated as 'bugs' and now fixed. Johannes confirmed that **"overlaping is now being checked ... not only for boxes but for all widgets"** . He added, **"if you click the detail-button you'll see the offending line in your XML-File - this makes debuging"** a GNUe Form Definition (gfd) **"a lot easier"** . James reported that all five of his existing GNUe Form Definitions were not working with the new code - but **"i would still imagine it's something funky I'm doing in the form"** rather than a problem with Johannes' code. He noted that, on the last one, the problem that he had been having with the dropdown menu had been fixed, but the form now **"aborts on query"** .

(ed. [Peter Sullivan] Note that the lack of any guarantees on backward compatability, even with 'features'/'bugs' is one of the reasons why GNUe Forms remains at a version number below 1.0 as of time of writing, as discussed further in Issue #112, Section #4 (**13 Apr :** Forms approaching version 1.0?) .)

Fig. 2: A project digest that summarizes multiple messages including those hyperlinked (indicated by highlighted underlined text) to their originating online sources. Source: http://www.kerneltraffic.org/GNUe/latest.html, accessed July 2006.

patches and initial operational/test results drawn from discussion forums, online chat transcripts, and related online artifacts [16].

Other common informalisms include (vi) scenarios of usage as linked Web pages or screenshot galleries, (vii) how-to guides, (viii) to-do lists, (ix) Frequently Asked Questions, and other itemized lists, and (x) project Wikis, as well as (xi) traditional system documentation and (xii) external publications [e.g., 24, 25]. OSS (xiii) project property licenses (whether to assert collective ownership, transfer copyrights, insure "copyleft," or some other reciprocal agreement) 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 externally developed software modules or "plug-ins", or (xvii) integrate software modules from other OSSD efforts, are all resources that are used informally, where or when needed according to the interests or actions of project participants.

All software informalisms are found or accessed from (xix) project related Web sites or portals. These Web environments are where most OSS software informalisms can be found, accessed, studied, modified, and redistributed [57]. A Web presence helps make visible the project's information infrastructure and the array of information resources that populate it. These include 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 project-specific Web sites (e.g., www.GNUenterprise.org), as well as (xx) embedded project source code Webs (directories), (xxi) project repositories (CVS [24] or Subversion), and (xxii) software bug reports [31] and (xxiii) issue tracking data base like Bugzilla [http://www.bugzilla.org/]. Last, giving the growing global interest in online social networking, it is not surprising to find increased attention to documenting various kinds of social gatherings and meetings using (xxiv) social media Web sites (e.g, YouTube, Flickr, MySpace, etc.) where FOSS developers, users, and interested others come together to discuss, debate, or work on FOSSD projects, and to use these online media to record, and publish photographs/videos that establish group identity and affiliation with different FOSSD projects.

Software informalisms as online artifacts which developers employ as their online workspaces are where and in what FOSSD is organized, captured, reviewed, and managed. Accordingly, these informalisms serve as affordances that facilitate, enculturate, and document collaborative work in FOSSD projects.

15.3.4 Skilled, self-organizing, and self-managed software 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 [http://filezilla.sourceforge.org] requires knowledge and skill in handling file transfer states, actions, 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 [51].

FOSS developers organize their work as a virtual organizational form [7, 18, 48] 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 hierarchical role/skill-based meritocracy [22, 6, 34] can arise. In such a meritocracy, there is no proprietary software development methodology or standard tool suite that all developers must employ, but critical decisions for what to do (e.g., overall system design) and how to do it will follow from respected core developers [cf. 51]. Similarly, there are few explicit rules about what development tasks should be performed, who should perform them, when, why, or how. However, this is not to say there are no rules that serve to govern the project or collective action within it.

FOSS project participants self-organize around the expertise, reputation, and accomplishments of core developers, secondary contributors, and tertiary reviewers and other peripheral users [9, 38]. FOSSD participants nearer the core have greater control and discretionary decision-making authority, compared to those further from the core [cf. 6, 9, 38]. Subsequently, core developers are expected to provide guidance, example artifacts, routinely use FOSSD coordination tools (e.g. CVS/Subversion, Bugzilla), and make critical decisions meritocratically. Together these afford collaborative FOSSD. Similarly, other participants must be able to both learn from and contribute to the efforts of the core developers. Together these realize a virtual, meritocratic, or self-managed form of decentralized software project management [7, 58].

15.3.5 Discretionary time and effort of developers

Are FOSS developers working for "free" or for advancing their career and professional development? Following the survey results of Hars and Ou [28] and others [23, 27], 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. Results from case studies in free software projects like GNUenterprise.org appear consistent with these observations [18, 19, 60]. These include self-determination, peer recognition, project affiliation or identification, and self-promotion, as well as belief in the inherent value of free software [cf. 11, 12, 25]. Core developers are expected to provide example through their own work practices, artifact contributions, and virtual project management style that other participants can observe, acknowledge, and learn from in ways that continue to afford collaborative FOSSD over time. Accordingly, the discretionary time, skill and effort of FOSS developers commit to their FOSSD project give rise to increased opportunity to collaborate, and increased collaborative activity can give rise to increased commitment of discretionary time and effort.

15.3.6 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. Actions that embody trust and accountability arise in many forms. They include (a) assuming ownership or responsibility of a project software module, (b) voting on the approval of individual action or contribution to ongoing project software [22], (c) shared peer reviewing [2, 11, 12], and (d) contributing gifts [3] that are reusable and modifiable common goods [49]. They also arise through the project's recognition of a developer's status, emerging reputation, and migration from peripheral roles to core contributor [34]. The ways and means through which FOSS developers exercise trust and social accountability mechanisms act to afford, build, and sustain collaborative work practices in FOSSD projects.

15.4 Cooperation, coordination, and control in FOSS projects

Getting software developers to work together, even when they desire to cooperate is not without its challenges for coordinating and controlling who does what when, and to what they do it to. Conflicts arise in both FOSSD [18, 33] and traditional software development projects [56], and finding ways to resolve conflicts becomes part of the cost (in terms of social capital) that must be incurred by FOSS developers for development 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. Focus of this section is thus directed to examining how different tools, artifacts, and socio-technical interaction practices are employed to enable FOSS developers to self-organize and govern project activities in an effective and adaptive manner.

Software version control tools such as the concurrent versions system, CVS--itself an FOSS system and document base [24]--have been widely adopted for use within FOSS projects [cf. 11, 12, 21, 25]. Tools like CVS are being used as both (a) a logically 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. In addition, the FOSS architecture the project organizes itself about may commonly be expressed through informal access/update rules and file/directory archiving schemes that are coded and agreed to by FOSS code contributors [cf. 51].

FOSSD projects teams can take the organizational form of a meritocracy [cf. 22, 58] operating as a dynamically organized virtual enterprise [7, 18, 48]. A layered meritocracy is a hierarchical organizational form that centralizes and concentrates certain kinds of authority, trust, and respect for experience and accomplishment within the team [cf. 6]. Such an organizational form also makes administrative governance more tractable and suitable, especially when a FOSS project seeks to legally constitute a non-profit foundation to better address its legal concerns and property rights [49]. However, it does not necessarily imply the concentration of universal authority into a single individual or directorial board, since decision-making may be shared among core developers who act as peers at the top layer, and they may be arrayed into overlapping groups with other project contributors with different responsibilities and interest areas [cf. 34].

Traditional software project management stresses planning, staffing, budget and schedule control activities. Virtual project management exists within FOSS communities, for example within projects developing FOSS-based computer games [58], to enable control via project decision-making, Web site administration, and administration of CVS/Subversion repositories (or other similar source code control tools). 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. The implied requirement for VPM can be seen within Fig. 3, from the FOSS project developing *Planeshift*, a free massively multiplayer online role-playing game. Similarly, VPM exists to mobilize and sustain the use of privately owned resources (e.g., Web servers, network access, site administrator labor, skill and effort) that are made available for shared use or collective reuse by the ongoing project [cf. 2, 60].

```
To be a leader you must pass the approval of the
director. Before that you will be considered a W.T.B.
(Want To Be) Leader and only after proving that you have
the right skills and dedication to the project you will
officially become a leader.

There's one leader for each department and he can have
also one co-leader helping in his job. He will ensure
progress in his department completing the most important
tasks in his area and will organize work of other
members. He is the primary reference for development.
Required Skills:
    * Strong committment to the project.
    * Good skill to organize work of the Team.
    * Team leadership.
    * Good knowledge of the area in which he applies
The leader is the most important contributor of his
department! He will complete critical tasks, he will
always have job to do. His tasks are similar to the ones
of the members (see below in the section of a specific
department). He will also manage work of other guys.
```

Fig. 3. Description of virtual project management skills implied for a Team Leader. Source: http://www.planeshift.it/recruitment.html, accessed October 2008.

Many FOSSD projects also post guidelines for how to report and discuss bugs, unintended features, or flaws in the current FOSS system release. These guidelines are embodied in online artifacts that developers follow in ways that suggest they have elevated certain informalisms into community norms (see Fig. 1) that act to encourage or control appropriate behavior within FOSSD projects.

Thus, a variety of socio-technical arrangements are put into motion in a FOSSD project in ways that encourage developers to cooperate, coordinate, and control their development activities through tools, informalisms, shared resources, and contribution practices. These collectively afford a lightweight centralized project management scheme through decentralized collaborative FOSSD practices.

15.5 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 (e.g., threaded discussions on bulletin boards) [45], and they find they share many technical competencies, values, and beliefs in common [7, 18, 20]. This manifests itself in the emergence of an alliance of collaborating FOSSD projects that share either common interests or development methods in projects that adopt a given FOSS system for subsequent application development, or in a occupational network of FOSS developers [18].

Examples of FOSS multi-project alliances are readily recognized. First, there are those that have established non-profit corporations or foundations like Apache, Mozilla, Gnome, Perl, Eclipse, NetBeans, or Free Software Foundation [49]. Second, there are those organized and supported by for-profit corporations by Sun Microsystems (e.g., OpenOffice), Hewlett-Packard, IBM, Nokia, and others [13, 52]. Third, other FOSS multi-project networks arise as the result of the architectural integration of multiple, disparate FOSS systems into larger, more encompassing system of systems [60]. Fourth, some FOSS projects produce systems that are platforms, frameworks or libraries of components which in turn give rise to application projects which are developed using these core systems. The Open Graphics Rendering Engine (OGRE at http://www.ogre3d.org), for instance, serves as the basis for dozens of user-led projects that build applications (like computer games) using OGRE. These projects both depend on OGRE project, as well as the network of other application projects, for FOSS code, updates, development expertise and advice. In turn, the peripheral participation of FOSS developers in these application projects can supplement the base of collaborating developers and users of the core systems.

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 [42, 59]. Multi-project clustering and interconnection enables small FOSS projects to come together as a larger social network with the critical mass [43] 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, 32, 51]. FOSSD Web sites also serve as hubs that centralize attention for what is happening with the decentralized development of the focal FOSS system, its status, participants and contributors, discourse on pending/future needs, etc. Subsequently, there is growing research interest in understanding, modeling, and analyzing the social and technical networks of FOSS developers [9, 30, 41, 42]. Fig. 4 provides an example of a social network of FOSS developers spanning five projects, but interlinked by just two developers.

Other studies [28, 35] indicate that upwards of two out of three OSS developers contributes to two or more FOSSD projects, and perhaps as many as 5% contribute to 10 or more FOSSD projects. The density and interconnectedness of this social networking characterizes the membership and in-breeding of the FOSS movement [15,17,19], but at the same time, the multiplicity of projects reflects its segmentation into specific socio-technical FOSSD domains.

All of these conditions for inter-project networking and alliance formation point to new kinds of requirements for collaborative software development—for example, network community building requirements, community software requirements, and community information sharing system (Web site and interlinked communication channels for email, forums, and chat) requirements [46, 57]. These requirements may entail both functional and non-functional requirements, but they will most typically be expressed using FOSS informalisms, rather than using formal notations based on

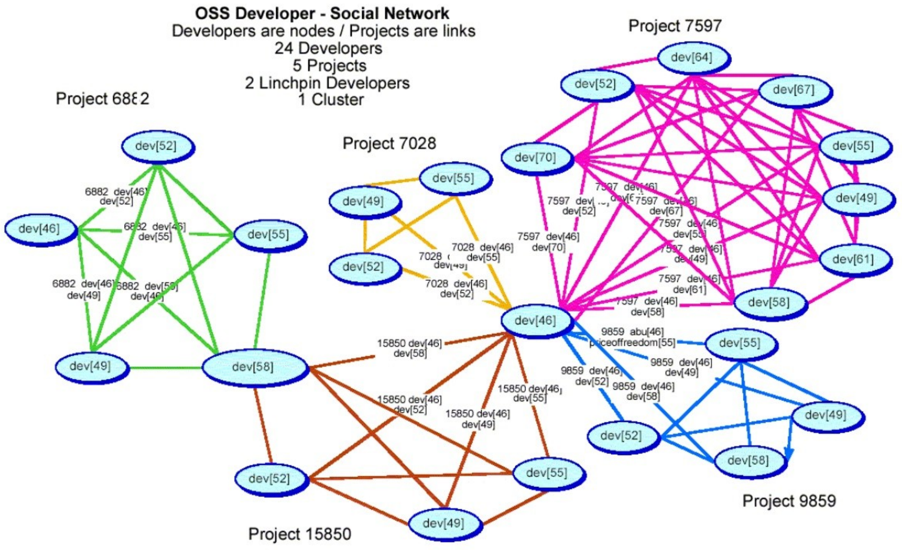


Fig. 4. A social network linking 24 FOSS developers in five projects through two "linchpin" developers into a larger multi-project community [42].

some system of mathematical logic known by few. Similarly, sharing beliefs, values, communications, artifacts and tools among FOSS developers enables not only cooperation, but also provides a basis for "common ground," shared mental models and experiences, camaraderie, and learning [cf. 20, 31, 38].

As such, the emergence of alliances among multiple, internetworked FOSSD projects helps to sustain and expand the viability of each participating project, along with the community of contributing developers (who are also users) and peripheral users. Together, they collectively afford collaborative software development connections and opportunities that transcend the boundaries of the constituent FOSSD projects.

15.6 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 and integration dependencies, functional conflicts, or vulnerabilities in one or more of the projects linked to it [33]. Fig. 5 depicts part of the FOSS ecosystem that supports a Web-based information infrastructure that interlinks Mozilla/Firefox Web browsers (and also Internet Explorer), Apache Web servers, NetBeans interactive development environment, Java development community (JCP), and others.

Interdependencies that span a software ecosystem are most apparent when FOSSD projects share source code modules, components, or sub-systems. In such situations, the volume of source code of an individual FOSSD project may appear to

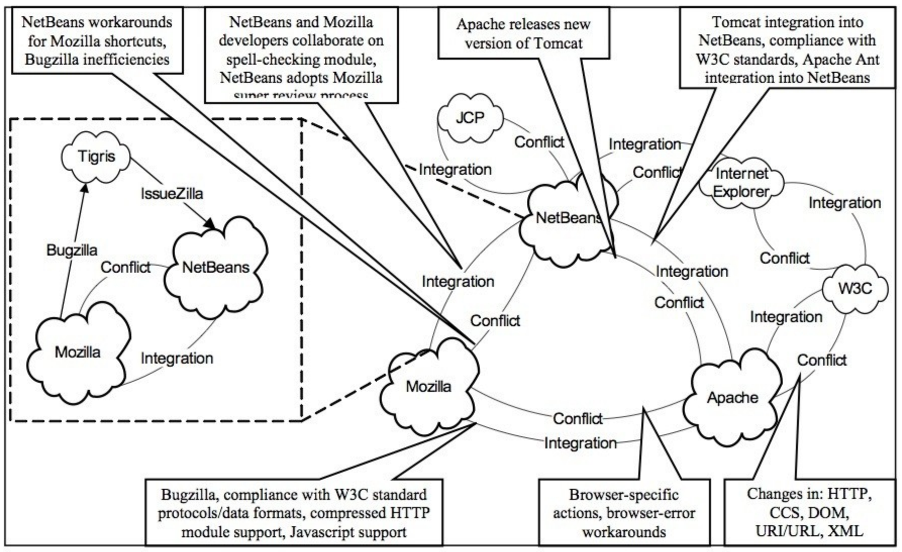

NetBeans workarounds for Mozilla shortcuts, Bugzilla inefficiencies
NetBeans and Mozilla developers collaborate on spell-checking module, NetBeans adopts Mozilla super review process
Apache releases new version of Tomcat
Tomcat integration into NetBeans, compliance with W3C standards, Apache Ant integration into NetBeans
JCP
Conflict
Integration
Conflict
Integration
NetBeans
Internet Explorer
Integration
Conflict
W3C
Integration
Conflict
Integration
Integration
Conflict
Conflict
Tigris
IssueZilla
Bugzilla
Conflict
NetBeans
Mozilla
Integration
Mozilla
Apache
Conflict
Integration
Bugzilla, compliance with W3C standard protocols/data formats, compressed HTTP module support, Javascript support
Browser-specific actions, browser-error workarounds
Changes in: HTTP, CCS, DOM, URI/URL, XML

Fig. 5. A depiction of a multi-project software ecosystem that supports Web-based information infrastructures [33].

grow at an *exponential rate* when modules, components, or sub-systems are integrated in whole into an existing FOSS system [8, 35, 59, 62, 67]. Such an outcome, which economists and political scientists refer to as a “network externality”

[50], 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 therefore seem to challenge the well-established laws of software evolution [39, 40]. Thus, software evolution in a multi-project FOSS ecosystem is a collaborative evolution ("co-evolution") process spanning interrelated FOSSD projects, people, artifacts, tools, code, and project-specific activities [59, 69].

It may also be useful to characterize a key evolutionary dynamic of FOSS as *reinvention* [cf. 58]. Reinvention is enabled through the sharing, examination, modification, and redistribution of concepts and techniques that have appeared in closed source systems, research and textbook publications, conferences, and the interaction and discourse between developers and users across multiple FOSS projects. Thus, reinvention is a continually emerging source for how to recreate, improve or invent new software functionality and quality in FOSS, as well as also a collaborative approach to organizational learning in FOSS projects [31, 38]. Said differently, reinvention is an effective way to learn how to innovate and invent, by re-producing and re-experiencing the technical problems, dead-ends, anomalous bugs, and challenges that others before them may have done. Reinvention is a way to (virtually) collaborate with those who have come before, which has long been a pedagogical strategy for education and learning.

Last, the layered meritocracies that arise in FOSS projects [34] tend to embrace or cultivate incremental innovations such as evolutionary mutations to an existing software code base, over radical innovations. These incremental mutations are most common in contributed revisions incorporated into daily/nightly builds of FOSS code. Radical software system changes might be advocated by a minority of code contributors who challenge the status quo of the core developers. However, their success in such advocacy usually implies creating and maintaining a separate version of the system through forking. *Forking* entails creating a duplicate copy of architected source code, then modifying or refactoring into a distinct new architectural configuration. Such forking may split/fragment a FOSSD project team into distinct sets of collaborators, which may results in no group having a sufficient critical mass of core developers. Thus, incremental FOSS mutations tend to win out over time since they more easily afford and sustain current collaboration patterns. Such affordance limits major FOSS system changes to arise slowly through meritocratic coordination and consensus building that give rise to new system versions with alternative architectural configurations [cf. 51, 58, 59].

15.7 Discussion

One discussion topic that immediately may come to mind is whether the collaboration affordances found in the FOSSD studies cited above might also be found in SE projects. At least four views of this topic can be considered.

First, we do not yet have in hand such a review of empirical studies of SE projects that identifies the collaboration affordances found therein, though such studies are starting to appear [cf. 29]. Though it might be an academic exercise to examine common SE textbooks to see what affordances for collaborative SE they might suggest or the reader might hypothesize, the point of this chapter was to focus review and examination of empirical studies of FOSSD to find what collaboration affordances are observed in these studies. So a fair and balanced comparison grounded in empirical studies is not yet possible due to a lack of such studies in SE projects.

Second, it is unclear to what extent such affordances found in SE projects that build proprietary (closed source) software systems in a centrally managed and controlled way, would be readily comparable to FOSSD projects that are self-organized and self-managed in a decentralized way. In traditional SE projects, developers are generally assumed to be collocated (although there are exceptions, like subcontracted, outsourced, or offshore development), while in FOSSD projects, developers are generally assumed to not be collated (with few exceptions like hackathons). Thus, while such an investigation might produce some sharp comparisons and keen insights, this is a matter that requires further empirical study.

Last, it is unclear whether there are studies of closed source SE projects that are organized as internetworked alliances, though it seems likely that networked multi-projects exist, though perhaps within the boundaries of a large corporate framework, or behind the corporate firewall [cf. 13].

15. Conclusions

This chapter provides a multi-level analysis of collaboration affordances that support free/open source software development work, through a review of dozens of empirical studies of FOSSD. Various kinds of collaboration affordances were identified with respect to individual participation in FOSSD projects, resources and capabilities that FOSS developers bring to a project, how FOSS developers cooperate and coordinate decentralized development activities, how multiple FOSSD projects coalesce into inter-networked alliances, and how FOSS ecosystems give rise to co-evolutionary patterns of growth and diversity. FOSSD can be understood as a socio-technical approach to collaborative software development supported through an array of collaboration affordances. The development of FOSS systems entails both the collaborative development of a networked project community, as well as the collaborative development of a network of software components and online artifacts. Consequently, some topics for further study can also be identified from this review.

First, it is possible to engage in systematic case studies of collaboration affordances that arise in comparable set of FOSSD projects. The findings reviewed in this chapter span multiple studies with different research methods, tools, data sets, and discipline-specific analytical lens [cf. 26, 30]. As collaboration affordances supporting software development are a relatively new topic of study, then it is appropriate to consider examining multiple FOSSD projects close up and in-depth to determine what affordances enable different kinds of collaborative activities in different development task situations. Case studies indicate such studies may rely more on qualitative, ethnographic field study and participant observation (i.e., become an active participant in one or more FOSSD projects to observe or discover collaboration affordances in action) [26, 55, 61].

Second, it may be possible to develop ways and means for mapping, visualizing, or animating collaboration affordances in action. As affordances associate properties of objects and actors that give rise to interactions in a situated environment, then it may be possible to identify and graphically portray these data elements in various kinds of networked representations [61]. There is a growing trend in studies focusing on social networks or technical dependencies within FOSSD projects to render their data and associations as different kinds of networks [10]. As such, how best to visualize collaboration affordances would be an intriguing avenue for exploration.

Last, as suggested in the Discussion section, there are numerous opportunities to study collaboration affordances within traditional software engineering projects. Similarly, there is need to systematically compare collaboration affordances found in FOSSD and SE projects so as to see what's similar, what's different, and why. The study of collaboration affordances in projects that seek to actively embrace and practice both FOSSD and SE is mostly unexplored territory, and many such projects can be found at the Tigris.org "open source software engineering" Web portal. Finally, as the review in this chapter indicates that affordances for collaborative software development can be analyzed at different/multiple levels of analysis, then multiple analytical lenses are now available to help focus new studies of collaborative software engineering. This chapter marks a starting point for further study.

Acknowledgments. The research described in this chapter has been supported by grants #0534771 and #0808783 from the National Science Foundation; also a grant from the Center for the Edge at the Naval Postgraduate School. No endorsement implied. Contributors to this research include Chris Jensen, Margaret Elliott, John Noll, Mark Ackerman, and others at the Institute for Software Research at the University of California, Irvine.

References

[1] Anderson, R. and Sharrock, W. (1992), Can Organisations Afford Knowledge? *Computer Supported Cooperative Work,* 1(3), 143-616.

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

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

[4] Churchill, E.F. and Bly, S. (1999). It's all in the words: supporting work activities with lightweight tools, *Proc. ACM Conf. Supporting Group Work,* 40-49, Phoenix, AZ.

[5] Couger, J. D., and Zawacki, R. A., (1980). *Motivating and managing computer personnel,* John and Wiley and Sons, New York.

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

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

[8] Deshpande, A. and Riehle, D. (2008). The Total Growth of Open Source, in IFIP Intern. Federation Info. Processing, Vol. 275; *Open Source Development,*

Community and Quality; B. Russo, E. Damiani, S. Hissan, B. Lundell, and G. Succi (Eds.), Boston, Springer, 179-209.

[9] de Souza, C. R. B., Froehlich, J., and Dourish, P. (2005). 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.

[10] de Souza, C. R. B., Quirk, S., Trainer, E., Redmiles, D., (2007). Supporting Collaborative Software Development through Visualization of Social and Technical Dependencies, *Proc. ACM Conf. Supporting Group Work (Group07)*, Sanibel, Island, FL, 147-156.

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

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

[13] Dinkelacker, J., Garg, P.K., Miller, R., and Nelson, D. (2002). Progressive Open Source, *Proc. 24th Intern. Conf. Software Engineering*, Orlando, FL, 177-184.

[14] Ducheneaut, N. (2005). Socialization in an Open Source software community: A socio-technical analysis. *Computer Supported Cooperative Work*, 14(4), 323-368.

[15] Elliott, M., (2008) Examining The Success of Computerization Movements in the Ubiquitous Computing Era: Free and Open Source Software Movements, in Elliott, M. and K.L. Kraemer, *Computerization Movements and Technology Diffusion,* Information Today, Inc.

[16] Elliott, M., Ackerman, M., and Scacchi, W. (2007). Knowledge Work Artifacts: Kernel Cousins for Free/Open Source Software Development, *Proc. ACM Conf. Support Group Work (Group07)*, Sanibel Island, FL, 177-186.

[17] Elliott, M. and K.L. Kraemer, (Eds.), (2008). *Computerization Movements and Technology Diffusion,* Information Today, Inc.

[18] Elliott, M. and Scacchi, W., (2005). Free Software Development: Cooperation and Conflict in a Virtual Organizational Culture, in Koch, S. (Ed.), *Free/Open Source Software Development*, IGI Publishing, Hershey, PA, 152-172.

[19] Elliott, M. and Scacchi, W., (2008) Mobilization of Software Developers: The Free Software Movement, *Information, Technology and People*, 21(1), 4-33.

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

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

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

[23] FLOSS (2002). *Free/Libre and Open Source Software: Survey and Study*, FLOSS Final Report, http://www.flossproject.org/report/ .

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

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

[26] Gasser, L. and Scacchi, (2008), Towards a Global Infrastructure for Multidisciplinary Studies of Free/Open Source Software, in IFIP Intern.

Federation Info. Processing, Vol. 275; *Open Source Development, Community and Quality*; B. Russo, E. Damiani, S. Hissan, B. Lundell, and G. Succi, (Eds.), Springer, Boston, 143-158.

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

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

[29] Herbsleb, J.D., Paulish, D.J. And Bass, M. (2005). Global software development at Siemens: Experience from nine projects, *Proc. 27th Intern. Conf. Software Engineering*, St. Louis, MO, 524-533.

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

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

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

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

[34] Jensen, C. and Scacchi, W., (2007). 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.

[35] Koch S. (2005a). Evolution of Open Source Software Systems—A Large-Scale Investigation, in *Proc. 1st Intern, Conf. Open Source Systems (OSS2005)*, Genoa, Italy.

[36] Kreijns, K. and Kirschner, P.A. (2001). The Social Affordances of Computer-Supported Collaborative Learning Environments, *Proc. 31st. ASEE/IEEE Frontiers in Education Conference*, TIF 12-17, Reno, NV.

[37] Lanzara, G.F. and Morner, M., (2005). 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.

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

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

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

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

[42] Madey, G., Freeh, V., and Tynan, R., (2005). Modeling the F/OSS Community: A Quantitative Investigation, in Koch, S. (Ed.) *Free/Open Source Software Development*, IGI Publishing, Hershey, PA, 203-221.

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

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

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

[46] Mynatt, E.D., O'Day, V.L., Adler, A., and Ito, M. (1998). Network Communities: Something Old, Something New, Something Borrowed,..., *Computer Supported Cooperative Work*, 7(1), 123-156.

[47] Norman, D. (1999). Affordances, Conventions, Design. *Interactions*, 6(3), 38–43.

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

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

[50] Ostrom, E., Eggertssons, T., and Calvert, R. (1990). *Governing the Commons: The Evolution of Institutions for Collective Action*, Cambridge University Press, Cambridge, UK.

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

[52] Robles, G., Duenas, S., and Gonzalez-Baharona, J.M., (2007). Corporate Involvement in Libre Software: Study of Presence in Debian Code over Time, in Feller, J., Fitzgerald, B., Scacchi, W., and Sillitti, A. (Eds.), *Open Source Development, Adoption and Innovation*, IFIP Vol. 234, Springer, Boston, 121-132.

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

[54] Robles, G., Gonzalez-Baharona, J.M., and Merelo, J.J., (2006b). Beyond Source Code: The importance of other artifacts in software development (a case study). *J. Systems and Software*, 79(9), 1233-1248.

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

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

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

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

[59] Scacchi, W., (2006). 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.

[60] Scacchi, W. (2007). Understanding the Development of Free E-Commerce/E-Business Software: A Resource-Based View, in S.K. Sowe, I. Stamelos, and I. Samoladas (eds.), *Emerging Free and Open Source Software Practices*, IGI Publishing, Hershey, PA, 170-190.

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

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

[63] Sommerville, I., (2006). *Software Engineering, 8th Edition*, Addison-Wesley, New York.

[64] von Hippel, E. (2005). *Democratizing Innovation,* MIT Press, Cambridge, MA.

[65] von Hippel, E., and von Krogh, G., (2003). Open Source Software and the "Private-Collective" Innovation Model: Issues for Organization Science, *Organization Science*, 14(2), 209-223.

[66] von Krogh, G., Spaeth, S., and Lakhani, K., (2003). Community, Joining, and Specialization in Open Source Software Innovation: A Case Study, *Research Policy*, 32(7), 1217-1241.

[67] Weiss, M., Moroiu, G. and Zhao, P., (2006). Evolution of Open Source Communities, in Damiani, E., Fitzgerald, B., Scacchi, W., Scotto, M. and Succi, G., (Eds.), *Open Source Systems*, IFIP Vol. 203, Springer, Boston., 21-32.

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

[69] Ye, Y., Nakajoki, K., Yamamoto, Y., and Kishida, K., The Co-Evolution of Systems and Communities in Free and Open Source Software Development, in Koch, S. (Ed.), *Free/Open Source Software Development*, IGI Publishing, Hershey, PA, 59-82. 2005.