13th ICCRTS: C2 for Complex Endeavors

Governance in Open Source Software Development Projects: Towards a Model for Network-Centric Edge Organizations

Topic(s): Organizational Issues, Collaborative Technologies for Network-Centric Operations, Networks and Networking

Walt Scacchi* and Chris Jensen (STUDENT)

Institute for Software Research

University of California, Irvine

Irvine, CA 92697-3455

949-824-4130, 949-824-1715 (fax)

wscacchi@uci.edu *(Contact person)

Title of Paper

Governance in Open Source Software Development Projects: Towards a Model for Network-Centric Edge Organizations

Abstract

Open source software development (OSSD) is a community-oriented, network-centric approach to building complex software systems. OSSD projects are typically organized as edge organizations that lack an explicit management regime to control and coordinate decentralized project work. However, a growing number of OSSD projects are developing, delivering, and supporting large-scale software systems that are displacing proprietary software alternatives. The U.S. Department of Defense is now committed to the adoption and deployment of software-intensive systems with open architectures and OSS components for application areas including command and control systems. Recent empirical studies of OSSD projects reveal that OSS developers often self-organize into organizational forms we characterize as evolving socio-technical interaction networks (STINs). These STINs emerge in ways that effectively control semi-autonomous OSS developers and coordinate project activities to produce reliable and adaptive software systems. In this paper, we examine how practices and processes enable and govern edge organizations like OSSD projects when coalesced and configured as contingent, socio-technical interaction networks. In so doing, we draw on results from two ongoing case studies of governance activities and elements in a small and a large OSSD project.

Keywords: open source software, governance, decentralized organizations, edge organizations

Introduction and overview

What are the best known ways and means for governing open source software development (OSSD)? Answering this question has been the focus of a number of research publications and recent presentations (described below). Why? That is, why is there now such a growing research interest in OSSD governance? Part of this stems from the interests of government or industry practitioners who seek to provide insight, explanatory frameworks, and guidance to others trying to account for how to integrate OSS products into existing management regimens. Another community of OSS scholars seeks to understand and explain how OSSD projects enact adaptive, situated, yet informal processes that effectively self-organize and govern OSSD practices without traditional project management or administrative regimes for resource control/allocation and decision-making authority. Similarly, a new cadre of business ventures and consultancies are also emerging seeking to advice commercial enterprises, government agencies, and others about what advice/assistance they need in order to adopt and integrate OSS into their current IT systems and operations.

In this paper, we contribute to this growing understanding for how to characterize the ways and means for affecting governance within and across OSSD projects, as well as the

participants and technologies that enable these projects and the larger communities of practice in which they operate and interact. Specifically, our contribution centers around providing an alternative perspective and analytical construct that offers multi-level analysis and explanation, as well as a framework for comparison and generalization based on empirical studies of OSSD projects, work practices, development processes, and community dynamics [cf. Scacchi 2007b]. The perspective draws from socio-technical interaction networks (STINs) [Scacchi 2005] as a persistent organizational form for collective action with/through technical (computing) work systems, and also puts forward STINs as the analytical construct that serves as an organizing concept, configurational form [Markus 2007], and adaptive process that both enacts and explains how governance in OSSD projects is realized and directed.

Overall, it is our current opinion that the governance practices enacted through STINs found in OSSD projects can be framed as possible options for understanding how network-centric edge organizations can operate without an explicit centralized command authority. Further, these STINs act in a self-organizing manner to effectively realize a decentralized approach to organize and control a dispersed, somewhat autonomous work force. This in turn can then be used to both understand (a) the possible roles of OSSD organizational practices in the development, deployment, and support of future command and control systems, and (b) where and how such practices could be used in support of contemporary military activity and organization [cf. Justice 2007, Starrett 2007].

This paper therefore organized to review what is currently known about governance activities, forms, and processes in OSSD projects, to identify the analytical elements of OSSD governance, to employ case study results to articulate these analytical elements, and to discuss how the elements observed in these results begin to layout how governance works in an OSSD project, and how this might serve as a new model for describing how governance forms in network-centric, edge organizations might operate more generally.

Related research

In broad terms, there are two kinds of efforts currently gaining attention regarding how best to understand governance and OSSD. The first focuses attention to what we will call "extrinsic issues" of governance and OSSD, while the second focuses on "intrinsic issues". A focus on *extrinsic issues* draw attention for how best to govern the results, outcomes, or products arising from OSSD projects, or matters like economic analysis of OSS licenses, contracts, and economic rents [cf. Demil and Lecocq 2006, Franck and Jungwirth 2003]. In contrast, a focus on *intrinsic issues* of governance attends to matters associated with OSS development activities, actors, project communities, and surrounding organizations that seek to encourage, facilitate, protect, or prosecute OSSD projects and collective action. Intrinsic issues address matters pertaining to decision-making authority, resource allocation, personal motives, leadership, social control, coordination mechanisms, organizational forms, etc.

OSS Governance within DoD

Within the world of the DoD, there is great interest in extrinsic issues of OSS governance, and little/no comparable interest currently being directed to intrinsic governance issues. To wit, effort is presently being marshaled and put into practice for

how best to govern the adoption, acquisition, and assimilation of OSS products, and how to integrate such practices into current project management regimens. Government-Industry presentations by Golden [2007], Justice [2007], Risacher [2007]. and others [DACS 2007] all focus attention to how best to govern or manage OSS products (or their licenses), and tend to slight/ignore the socio-technical processes and project communities that create, evolve, and sustain these products [Scacchi and Alspaugh 2008].

Outside of the DoD world, there is recent flow of research examining a variety of intrinsic issues that attempt to characterize or explain how governance is achieved or realized within OSSD projects.

OSS Governance beyond DoD

Shah [2006] identifies mechanisms that serve to help govern OSSD activities in the two OSSD project communities she studied that include decision-making rights, property rights, proprietary modifications, and restrictions on modification and redistribution. Use of these governance mechanisms helps to determine how an OSSD project strikes a balance between traditional concerns for organizational control of property while accommodating the informal collective needs of those who will voluntarily share the results of their labor, as long as they do not feel exploited.

O'Mahony [2007] also finds a hybrid governance form in OSSD project community she studied that allows for private parties to participate and gain advantage, while also contributing to the growth of the informal project community. She identifies fives principles of community managed governance that include autonomous participation and decentralized decision-making among others.

Markus [2007] adds to these results through a secondary analysis of prior OSSD studies that sought to identify governance issues that appear in the background of those studies. She finds that different studies of different OSSD projects reveal different patterns of governance practices, processes, and mechanisms. This leads her to observe that governance in OSSD projects is better viewed as configurational—following from a distinct configuration of collective social actions and technical system arrangements (e.g., specific OSS tools in use). We find her configurational governance concept similar in form to our socio-technical interaction networks [Scacchi 2005, Scacchi 2007b], also as a situated and contingent configuration of interrelated social actions, technical systems, and the collective work practices and development processes that can bring them together in persistent, yet continuously emerging OSSD projects and products.

Last, in a survey by de Latt [2007], he finds a distinction between what he calls spontaneous governance, internal governance, and governance toward outside parties, all of which we group under intrinsic governance issues. Though he does not provide any results for extrinsic governance, he does provide a framing that accounts for governance of OSS product development as arising from different types of social networks [cf. de Latt 2004].

Analytical Levels and Elements for Understanding Governance in OSSD Projects

Beyond the related research efforts identified above, other empirical studies of OSSD reveal that OSSD work practices, engineering processes, and project community dynamics can best be understood through observation and examination of their socio-technical elements from multiple levels of analysis [Scacchi 2007b]. In particular, OSSD projects can be examined through a "micro-level" analysis of (a) the actions, beliefs, and motivations of individual OSSD project participants, and (b) the social or technical resources that are mobilized and configured to support, subsidize, and sustain OSSD work and outcomes [Scacchi 2007a]. Similarly, OSSD projects can be examined through "meso-level" analysis of (c) patterns of cooperation, coordination, control, leadership, role migration, and conflict mitigation, and (d) project alliances and inter-project socio-technical networking [de Latt 2004]. Last, OSSD projects can also be examined through "macro-level" analysis of (d) multi-project OSS ecosystems, and (e) OSSD as a social movement and emerging global culture. As such, we will provide a multi-level analysis of the elements of OSSD governance.

We will engage in our multi-level analysis of the elements of OSSD governance using results drawn primarily from two ongoing, longitudinal case studies of OSSD projects. These projects are respectively associated with the GNUe.org [Elliott and Scacchi 2005, Elliott, Ackerman, and Scacchi 2007, Scacchi 2007a] and NetBeans.org [Jensen and Scacchi 2005, 2007] Web sites, where these projects can be found. GNUe.org is focused on the development and use of software components and libraries for developing electronic business applications and services [Scacchi 2007a]. GNUe.org is a small OSSD project with about 20 or so regular project contributors over its last six years of development. NetBeans.org is focused on the development, support, and evolution of an Integrated Development Environment (IDE), which is a tool for developing Web-based enterprise software applications coded in the Java programming language that utilize other Java-based software products and services, such as those offered by Sun Microsystems Inc. [Jensen and Scacchi 2005]. NetBeans.org is a very large OSSD project with more than 400,000 active users of its IDE, and has contribution to the project from tens of thousands of project participants. As such, it should not be surprising that our case studies are not congruent in their examination of OSSD governance elements, since our studies examine a very large and a small OSSD project, and thus were not conceived to be directly comparable in that way. Nonetheless, we recognize and acknowledge this limitation in order to move forward to layout what we have observed and learned so far with respect to OSSD governance, where we more see more elements of governance in the larger and more diverse NetBeans.org project community, while we more clearly see micro-level elements for individual action in the smaller, more personal GNUe.org OSSD project.

Finally, it is our view that the elements of OSSD governance span these multiple levels of analysis because they coalesce and are actively configured by OSSD project participants into network forms for collective action—networks we designate as socio-technical interaction networks (STINs) [Scacchi 2005]. Why? Our observation drawn from our own studies of OSSD and those of others [de Latt 2004, 2007, Markus 2007, Scacchi 2007b] suggest to us that governance activities, efforts, and mechanisms are not disjoint or unrelated to one another, but instead are arrayed and configured by OSSD project participants into networks for mobilizing socio-technical interactions, resources, rules,

and organizational forms. Project participants are only accountable to each other, and not to corporate owners, senior executives, or stock investors. They can often suffice with lightweight governance forms that they configure and adapt to their needs and situations, rather than to budget, schedules, or profit growth. Accordingly, the choose organizational forms that are neither purely decentralized market (or "bazaar") nor a centralized hierarchy (or "cathedral"), but instead choose a more general network form that can be more readily be adapted to local contingencies or emergent conditions that arise in the interactions among project participants, the technical computing systems/resources at hand, or the joint socio-technical system that is the OSSD project. Thus, our multi-level analysis is one that is construed to draw attention to the persistent yet adaptive STINs that participants enact to span and govern OSSD projects, practices, and processes that arise at different levels of socio-technical interaction.

Micro-level analysis of OSSD governance

Our analysis of OSSD governance begins by examining what elements of individual/participant action and what resources can OSSD project participants mobilize to help govern the overall activities of their project work and contributions. These are examined in turn.

Accounting for Individual Actions and Resources

Why will OSSD project participants contribute so much of themselves, often on a voluntarily basis, to a OSSD project? In simple terms, they recognize and experience intrinsic motivations that serve their own self-interest. In a project like GNUe.org, the most commonly cited reasons for why participants continue to contribute to the project includes their desire to: (a) learn about new GNUe tools through using and enhancing them; (b) have fun building software with other people who also enjoy building the GNUe software; (c) exercise technical skills that their regular jobs might not accommodate; (d) to try out the GNUe software in their regular workplace or with their business clients; and (e) to interconnect with other OSS developers working on other OSSD projects that may or may not be related to GNUe.org. However, in realizing these personal incentives, the GNUe.org participants also (f) build trust and reputation with one another, while (f) the project's core developers (who change over time) realize "geek fame" and recognition as technical authority or "lead" of the current GNUe software system architecture. Furthermore, to be sure that the active GNUe.org software contributors know who is doing what, what is going on, and why, (g) developers spend much of their time as a project participant reading about what others are doing, have done, or are talking about using regular project communication channels, like Internet Relay Chat and project digest summaries [cf. Elliott, Ackerman, and Scacchi 2007]. These (a) through (g) elements collectively act to constrain what gets done, and how it gets done, so that GNUe.org developers believe that they do not require project managers or project development schedules in order to govern the themselves or the technical direction of the GNUe development effort [Elliott and Scacchi 2005]. The effort will only go where the participants want to take it.

Beyond the intrinsic motivations of GNUe.org developers, these participants also mobilize a variety of their personal resources at their disposal [Scacchi 2007a]. This is done in order to make clear their commitment to the project, the other participants in the

project community, and to the technical choices, system architecture, and overall development trajectory that are guiding/governing their collective OSSD efforts. The resources that participants put into play include their: (i) personal computing resources (including their PCs, network file servers, and data repositories); (ii) beliefs supporting the moral value and rightness for developing software that can be freely accessed by others, studied, modified, and redistribution into perpetuity; (iii) routine use of a multitude of various software "informalisms" [Scacchi 2002] to communicate different socio-technical issues to project members or outsiders; (iv) technical skill as a competent as a software developer, and their social skill in working well with others whom they may at times disagree with on technical matters; (v) discretionary time and effort, which often means they engage in project work "after hours" away from their day job, and often at home; and (vi) the trust and social accountability they build up and sustain through ongoing interaction with others participants in the project who they believe are acting in a similar way to sustain their overall collective activity. Once again, the choices OSSD participants make in mobilizing the personal resources they invest in the OSSD project, GNUe.org developers effectively constrain their collective effort in ways that make the ongoing project a self-governing project community that lacks a central budget, schedule, or resource allocation authority, yet realizes much of the resources needed to sustain (but not yet grow) the GNUe.org project community.

Resources and Artifacts as Objects of Interaction

Much of the development work that occurs in an OSSD project centers around the creation, update, and other actions (e.g., copy, move, delete) applied to a variety of software development artifacts. These artifacts serve as coordination mechanisms [Schmidt and Simone 1996, Simone and Mark 1999], in that they help participants communicate, document, and otherwise make sense of what the emerging software system is suppose to do, how it should be or was accomplished, who did what, what went wrong before, how to fix it, and so forth. Furthermore, within a project community these artifacts help coordinate local, project-specific development activities, whereas between multiple project communities, these artifacts emerge as boundary objects [Lee 2007] through which inter-community activities and relations are negotiated and revised. The artifacts may take the form of text messages posted to a project discussion list, Web pages, source code directories and files, site maps, and more, and they are employed as the primary media through which software requirements and design are expressed. These artifacts are software informalisms [Scacchi 2002]. They are especially important as coordination mechanisms in OSSD projects since participants generally are not co-located, they do not meet face-to-face, and authority and expertise relationships among participants is up for grabs.

In the context of the NetBeans.org project and its role within a larger Web-compatible information infrastructure, additional artifacts come into play within and across projects. These include the content transfer protocols like the HyperText Transfer Protocol (http) which are systematically specified in Internet standards like RFC documents, as well as more narrowly focused communication state controllers associated with remote procedure calls (or remote method invocations). They also include shared data description formats like the HyperText Markup Language (html) and the eXtensible Markup Language (XML), as well as client-side or server-side data processing scripts (e.g., CGI routines). Such descriptions may be further interpreted to enable externally

developed modules to serve as application/module plug-ins, which enable secondary or embedded applications to be associated with an OSS system. Other artifacts are brought in from other OSSD projects to serve as project support tools, such as those used to record and store system defect reports (Issuezilla[1]), email list managers, and even large comprehensive collaborative software development environments and project portals, like CollabNet (www.collab.net) or SourceCast [Augustin, Bressler, and Smith 2002]. Finally, OSSD projects may share static and operational artifacts in the course of collaborating or cooperating through mutually intelligible and interoperable development processes, which might take an explicit form like the Java Community Process (JCP), or an implicit and embedded form such as that which emerges from use of project repositories whose contents are shared and synchronized through tools that control and track alternative versions (CVS) or Web site content updates.

Accordingly, in order to explore where issues of collaboration, leadership, control and conflict may arise within or across related OSSD projects, then one place to look to see such issues is in how project participants create, update, exchange, debate, and make sense of the software informalisms that are employed to coordinate their development activities. This is the approach taken here in exploring the issues both within the NetBeans.org project community, as well as across the fragile software ecosystem of inter-related OSSD projects that situate NetBeans.org within a Web information infrastructure [Jensen and Scacchi 2005].

Meso-Level Analysis of OSSD Project Community Issues

As noted earlier, NetBeans.org is a large and complex OSSD project. We have observed at least three kinds of governance elements that arise *within* an OSSD community like NetBeans.org. These are collaboration, leadership and control, and conflict.

Collaboration

According to the NetBeans.org community Web site, interested individuals may participate in the community by joining in discussions on mailing lists, filing bug and enhancement reports, contributing Web content, source code, newsletter articles, and language translations. These activities can be done in isolation, without coordinating with other community members, and then offered up for consideration and inclusion. As we'll see, reducing the need for collaboration is a common practice in the community that gives rise to positive and negative effects. We discuss collaboration in terms of policies that support process structures that prevent conflict, looking at task completion guidelines and community architecture.

Policies and Guidelines

The NetBeans.org community has detailed procedural guidelines[2] for most common development tasks, from submitting bug fixes to user interface design and creating a new release. These guidelines come in two flavors: development task and design style

1 http://www.netbeans.org/kb/articles/issuezilla.html

2 http://www.netbeans.org.org/community/guidelines/

guidelines. In general, these policies are practiced and followed without question. Ironically, the procedures for policy revision have not been specified.

Precedent states that revisions are brought up on the community or module discussion mailing lists, where they are debated and either ratified or rejected by consensus. Developers are expected to take notice of the decision and act accordingly, while the requisite guideline documents are updated to reflect the changes. In addition, as some communities resort to "public flogging" for failure to follow stated procedures, requests for revision are rare and usually well known among concerned parties, so no such flogging is done within NetBeans.org.

Overall, these policies allow individual developers to work independently within a process structure that enables collaboration by encouraging or reinforcing developers to work in ways that are expected by their fellow community members, as well as congruent with the community process.

Separation of Concerns: an Architectural Strategy for Collaborative Success

Software products are increasingly developing a modular, plug-in application program interface (API) architectural style in order to facilitate development of add-on components that extend system functionality. This strategy has been essential in an open source arena that carries freedom of extensibility as a basic privilege or, in some cases, the right of free speech or freedom of expression through contributed source code. But this separation of concerns strategy for code management also provides a degree of separation of concerns in developer management, and therefore, collaboration.

In concept, a module team can take the plug-in API specification and develop a modular extension for the system using any development process in complete isolation from the rest of the community. This ability is very attractive to third-party contributors in the NetBeans.org community who may be uninterested in becoming involved with the technical and socio-political issues of the community, or who are unwilling or unable to contribute their source code back to the community. Thus, this separation of concerns in the NetBeans.org design architecture engenders separation of concerns in the process architecture. Of course, this is limited by the extent that each module in NetBeans.org is dependent on other modules.

Last, volunteer community members have periodically observed difficulties collaborating with volunteer community members. For example, at one point a lack of responsiveness of the (primarily Sun employed) user interface team[3], whose influence spans the entire community, could be observed. This coordination breakdown led to the monumental failure of usability efforts for a period when usability was arguably the most-cited reason users chose competing tools over NetBeans.org. Thus, a collaboration failure gave rise to product failure. Only by overcoming collaboration issues was NetBeans.org able to deliver a satisfactory usability experience[4].

Leadership and Control

3 http://www.netbeans.org.org/servlets/ReadMsg?msgId=531512&listName=nbdiscuss

4 http://www.javalobby.org/thread.jspa?forumID=61&threadID=9550#top

Ignoring internal Sun (and third party) enterprise structure, there are five observable layers of the NetBeans.org community hierarchy. Members may take on multiple roles some of which span several of these layers. At the bottom layer are users, followed by source contributors, module-level managers, project level release managers (i.e. IDE*or* platform), and finally, community level managers (i.e. IDE *and* platform) at the top-most layer. Interestingly, the "management" positions are simply limited to coordinating roles; they carry no other technical or managerial authority. The release manager, for example, has no authority to determine what will be included in and excluded from the release[5]. Nor does s/he have the authority to assign people to complete the tasks required to release the product. The same is true of module and community managers. Instead, their role is to announce the tasks that need to be done and wait for volunteers to accept responsibility.

Accountability and expectations of responsibility are based solely on precedent and volunteerism rather than explicit assignment, leading to confusion of the role of parties contributing to development. Leadership is not asserted until a community member champions a cause and while volunteerism is expected, this expectation is not always obvious. The lack of a clear authority structure is both a cause of freedom and chaos in open source development. Though often seen as one of its strengths in comparison to closed source efforts, it can lead to process failure if no one steps forward to perform critical activities or if misidentified expectations cause dissent.

The difficulties in collaboration across organizations within the community occasionally brought up in the community mailing lists stem from the lack of a shared understanding leadership in the community. This manifests itself in two ways: a lack of transparency in the decision making process and decision making without community consent. While not new phenomenon, they are especially poignant in a movement whose basic tenets include freedom and knowledge sharing.

Transparency in the Decision Making Process

In communities with a corporately backed core development effort, there are often decisions made that create a community-wide impact that are made company meetings. However, these decisions may not be explicitly communicated to the rest of the community. Likewise private communication between parties that is not made available on the community Web space or to the forwarded to other members is also hidden. This lack of transparency in decision-making process makes it difficult for other community members to understand and comply with the changes taking place if they are not questioned or rejected. This effect surfaced in the NetBeans.org community recently following a discussion of modifying the release process[6].

Given the magnitude of contributions from the primary benefactor, other developers were unsure of the responsibility and authority Sun assumed within the development process. The lack of a clearly stated policy outlining these bounds led to a flurry of excitement when Sun members announced major changes to the licensing scheme used by the community

---

5 http://www.netbeans.org.org/community/guidelines/process.html

6 http://www.netbeans.org/servlets/BrowseList?listName=nbdiscuss&by=thread&from=19116&to=19116&first=1&count=41

without any warning. It has also caused occasional collaboration breakdown throughout the community due to expectations of who would carry out which development tasks. The otherwise implicit nature of Sun's contributions in relation to other organizations and individuals has been revealed primarily through precedent rather than assertion.

Consent in the Decision Making Process

Without an authority structure, all decisions in development are done through consensus, except among those lacking transparency. In the case of the licensing scheme change, some developers felt that Sun was within its rights as the major contributor and the most exposed to legal threat [7] while others saw it as an attack on the "democratic protection mechanisms" of the community that ensure fairness between participating parties[8]. A lack of consideration and transparency in the decision making process tend to alienate those who are not consulted and erode the sense of community.

Conflict Resolution

Conflicts in the NetBeans.org community are resolved via community discussion mailing lists. The process usually begins when one member announces dissatisfaction with an issue in development. Those who also feel concern with the particular issue then write responses to the charges raised. At some point, the conversation dissipates- usually when emotions are set aside and clarifications have been made that provide an understanding of the issue at hand. If the problem persists, the community governance board is tasked with the responsibility of resolving the matter.

The governance board is composed of three individuals and has the role of ensuring the fairness throughout the community by solving persistent disputes. Two of the members are elected by the community, and one is appointed by Sun Microsystems. The board is, historically, a largely superficial entity whose authority and scope are questionable and untested. While it has been suggested that the board intercede on a few rare occasions, the disputes have dissolved before the board has acted. Nevertheless, board elections are dutifully held every six months[9].

Board members are typically prominent members in the community. Their status carries somewhat more weight in community policy discussions, however, even when one member has suggested a decision, as no three board members have ever voted in resolution on any issue, and thus, it is unclear what effect would result. Their role, then, is more of a mediator: to drive community members to resolve the issue amongst themselves. To this end, they have been effective.

Macro-Level Analysis of OSSD Governance Across Community Issues

As noted earlier, the NetBeans.org project is not an isolated OSSD project. Instead, the NetBeans IDE which is the focus of development activities in the NetBeans.org project community is envisioned to support the interactive development of Web-compatible

---

7 http://www.netbeans.org.org/servlets/ReadMsg?msgId=534707&listName=nbdiscuss

8 http://www.netbeans.org.org/servlets/ReadMsg?msgId=534520&listName=nbdiscuss

9 http://www.netbeans.org.org/about/os/who-board.html

software applications or services that can be accessed, executed, or served through other OSS systems like the Mozilla Web browser and Apache Web server. Thus, it isreasonable to explore how the NetBeans.org project community is situated within an ecosystem of inter-related OSSD projects that facilitate or constrain the intended usage of the NetBeans IDE. Figure 1 provides a rendering of some of the more visible OSSD projects that surround and embed the NetBeans.org within a Web information infrastructure. This rendering also suggests that issues of like collaboration and conflict can arise at the boundaries between projects, and thus these issues constitute relations that can emerge between project communities in OSSD ecosystem.

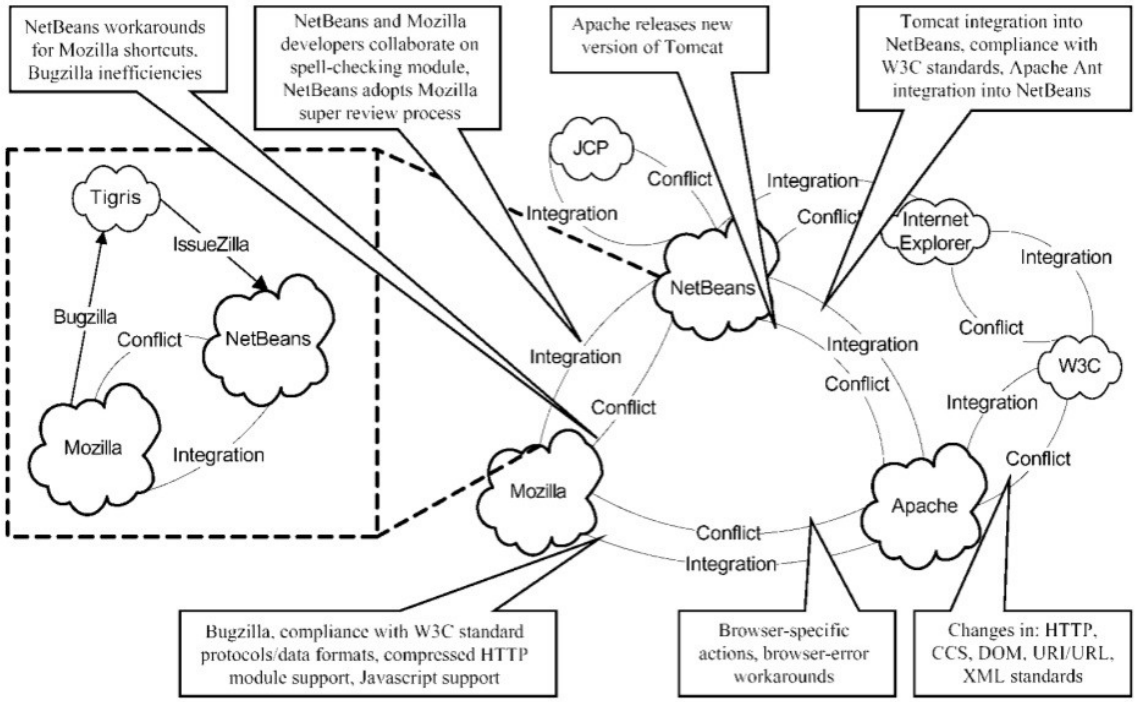

Figure 1. An overview of some of the OSS projects that surround and situate development activities within the NetBeans.org OSS project [Jensen and Scacchi 2005].

With such a framing in mind, we have observed at least three kinds of issues arise across OSSD communities that surround the NetBeans.org community. These are communication and collaboration, leadership and control, and conflict resolution.

Communication and Collaboration

In addition to their IDE, NetBeans.org also releases a general application development platform on which the IDE is based. Other organizations, such as BioBeans and RefactorIT communities build tools on top of or extending the NetBeans platform or IDE. How do these organizations interact with NetBeans.org, and how does NetBeans.org interact with other IDE and platform producing organizations? For some organizations, this collaboration may occur in terms of bug reports and feature requests submitted to the NetBeans.org issue-tracking repository. Additionally, they may also submit patches or participate in discussions on community mailing list or participate in the NetBeans.org

"Move the Needle" branding initiative. Beyond this, NetBeans.org participates in the Sun sponsored *Java.net* meta-community, which hosts hundreds of Java-based OSSD projects developed by tens of thousands of individuals and organizations.

A fellow member of the Java.net community is an attempt to bring tool developers together to form standards for tool interoperability. However, the Java Tools Community, considered by some to be a working group[10] for the Java Community Process. Thus NetBeans.org, through its relationship with Sun, is a collaborating community in the development of, and through compliance with, these standards, and looks to increasing collaboration with other tool developing organizations.

Leadership and Control

OSSD generally embrace the notion of choice between software products to build or use. At the same time, developers in any community seek success for their community, which translates to market share.

In some cases, communities developing alternative tools do so in peaceful coexistence, even collaboratively. In other cases, there is a greater sense of competition between rivals. NetBeans and its chief competitor Eclipse (backed largely by IBM) fall into the latter category. Eclipse has enjoyed some favor from users due to performance and usability issues of NetBeans, as well as IBM's significant marketing and development resource contributions. Yet, they have a willingness to consider collaborative efforts to satisfy demands for a single, unified IDE for the Java language that would serve as a platform for building Java development tools and a formidable competitor to Microsoft's .NET. Ultimately, the union was defeated, largely due to technical and organizational differences between Sun and IBM[11], including the inability or unwillingness to determine how to integrate the architectures and code bases for their respective user interface development frameworks (Swing for NetBeans and SWT for Eclipse).

Conflict Resolution

Conflicts between collaborating communities are resolved in similar fashion to their means of communication--through discussion between Sun and Eclipse representatives, comments on the NetBeans.org mailing lists, or other prominent technical forums (e.g. Slashdot and developer blogs). Unfortunately, many of these discussions occur after the collaborating developer has moved away from using NetBeans.org (often, in favor of Eclipse). Nevertheless, the feedback they provide gives both parties an opportunity to increase understanding and assists the NetBeans.org community by guiding their technical direction.

Discussion

Public communication channels we have seen used in OSSD projects like the GNUE.org and NetBeans.org include mailing lists, defect repositories, requests for enhancement, Internet Relay Chat (IRCs), developer/stakeholder blogs and Web pages, trade forums, and developer conferences. Of these, mailing lists, defect repositories, and requests for

---

10 http://www.internetnews.com/dev-news/article.php/3295991

11 http://www.adtmag.com/article.asp?id=8634, http://www.eweek.com/article2/0,1759,1460110,00.asp

enhancement (RFEs) are intra-organizational--they exist within project community boundaries. IRC chats and developer conferences that facilitate communication may be intra or inter-organizational, in that they can be hosted by the community or by other organizations. On the other hand, stakeholder blogs and Web pages and trade forums are purely inter-organizational. Communication channels provide means for enabling intrinsic governance in OSSD projects through collaboration, leadership, control, and conflict negotiation processes. But they do not tell us much about how developers collaborate, lead, control, and resolve conflicts, nor what is collaborated on, led, controlled, and causing/resolving conflicts. We address that here.

In the NetBeans community, we have observed the following objects of interaction which guide OSSD technical development and social integration processes:

- Project and software system architecture
- The community vision/mission statement
- Release plans and development roadmap
- Community policies, task guidelines, and interaction guidelines
- Defect reports and request for enhancements (RFEs)
- Mailing list discussions
- Private meetings (work done by organizations associated with the community)

Arguing that project architecture is a primary coordination mechanism for software development, Ovaska and colleagues [2003], and also Baldwin and Clark [2006], observed six coordination processes in multi-site software development like OSSD projects. These include managing interfaces between system components, managing assembly order of system components, managing the interdependence of system components, communication, overall responsibility, and orientation (configuration) of the organization. There are several processes in which governance and coordination is manifested.

As suggested above, community interaction modes act as communication channels for governing, coordinating, and articulating of development tasks. Community mission statements are important to the formation of the community social and technical infrastructure early in the community's lifespan when more concrete guidelines have not been explicitly stated (if established). They are the core instructions for the way individuals and organizations will interact with the community as a whole. But they are also a metric by which each release will be judged.

Additional release planning activities in OSSD typically consist of asserting the requirements for the release (*what* work will be done), the schedule of the release (*when* will the work be completed), and who will be responsible for what work (*who* will do *what* work) [Scacchi, 2002].

Defect/product recovery and redesign, as registered through submission of bug/defect reports is an integral coordination process. Like release planning, defect reports and RFCs (Request for Comments) tell developers both what work needs to be done as well as what has not been done yet, without an explicit owner or administrative supervisor to assign responsibility for doing it.

These suggest that governance processes are inherent in activities requiring coordination or leadership to determine which development tasks need to be done and when they need to be completed. This is analogous to what has previously been observed by management scholars (and also OSS developers) as adaptive "Internet Time" development practices [Cusumano 1999] that enable a kind of project self-governance through adaptive synchronization and stabilization activities.

In some instances, leadership in coordinating development tasks is done in private meetings or communications between developers, for which little evidence is public or observable. However, we observed leadership and control of OSSD project community through:

- Contribution of software informalisms (e.g., source, defect reports, requests for changes, news, internationalizations, etc. [Scacchi 2002])

- Articulating and sharing technical expertise (e.g., on the mailing lists and defect repository reports, [Elliott, Ackerman, and Scacchi 2007])

- Coordination of development and other tasks (e.g., through the role of the release manager, module maintainer, and source code contributors with "commit access" to shared source code repositories).

OSSD communities are often controlled and governed through a skill and commitment-based meritocracy. With a contribution-based reputation scheme, GNUe.org and NetBeans.org are no exception. Control of each community's technical direction depends on what OSS code contributions, defect reports, patches, and enhancement requests are submitted and enacted. Developers volunteering on high-demand or critical path aspects of the project will likely have many peers needing to coordinate, synchronize and stabilize their activities in order to integrate the outcomes of their work. If the implementation is consistent with the requirements and does not negatively impact the efforts of other community members, the direction of the contribution will stand. Otherwise, community members will execute a conflict resolution process. As developers consistently demonstrate quality in their social and technical contributions, their peers take notice. There are also non-technical aspects of the community that are controlled. These include control of source licensing schemes, social community infrastructure, and inter-organizational relationships.

The NetBeans.org community is a complex case: it receives the majority of its financial and core developmental support from Sun Microsystems. Sun, as the primary benefactor and community founder, established the community vision, social and technical infrastructure, many core developers, and initiates most release plans, driving the development roadmap. Thus, Sun is most exposed to risks from community failure and external threats. As demonstrated by Sun's move to alter the project licensing scheme,

exercising this authority unilaterally led to division within the community, risking breakdown of the project and development process. As such, social process conflict can give rise to conflict within the overall technical development process.

In contrast, the GNUe.org community though significantly smaller, is a similar case in some ways. First, external enterprises that contribute paid or volunteer software development staff do so as an investment that seeks financial returns to these companies buy providing them with software that they can in turn market to their customers [Scacchi 2007a]. Second, GNUe.org is ideologically aligned with the Free Software Foundation (www.fsf.org) and the General Public License (GPL) for software. The FSF serves as a non-profit enterprise that advocates the advancement of "free software" (as opposed to "open source software") as a social movement seeking to transform and liberate software users from the technical and moral confines of proprietary software (cf. www.fsf.org). The need to practice such liberation is subsequently reiterated by community members through informalisms routinely used by GNUe.org participants, and through technical choices made by participants regarding which tools and techniques should (those aligned with free software) and should not (those aligned with proprietary software) be used. Last, the FSF does however identify and support GNUe.org as one of its official projects, and provides modest support through hosting of GNUe.org project Web site and information repositories.

Based on this, sources of conflict that precipitate some form of governance may arise from:

- Community infrastructure, sociopolitical vision, and direction

- Technical direction (what should be in the release, when should a release occur, which tools to use to develop software)

- How developers can get involved in making decisions and what roles they play

- Relationships between and alignment of the diverse goals of many organized groups

(e.g., corporations) and unaffiliated volunteers involved in the community

These conflicts are resolved through OSSD governance activities in a variety of ways. When conflicts arise due to miscommunication or lack of communication between developers, or between developers and organized groups contributing to the community, resolution is reached by talking it out on community mailing lists. In more pronounced cases, it may take project veterans and highly influential community members to act as mediators. Failing this, in NetBeans.org, the project culture prescribes that developers shall bring the issue to the governance board for deliberation, who will issue a final decision on the matter. Board involvement is viewed as a last resort, and community members are encouraged to resolve their conflicts through other means. In contrast, in GNUe.org, the culture of free software aligned with the FSF espouses moral values and choices should guide technical choices, and thus these values, norms, and beliefs govern project activities [Elliott and Scacchi 2005].

All told, social and technical conflicts are intertwined in OSSD. Lacking formal process prescriptions, if the social processes are running smoothly, all that needs to be articulated and synchronized are technical development processes. But if the technical process is too ambiguous or not specified explicitly as is often the case in OSSD, developers must rely on collaboration, leadership, control, and conflict negotiation to fill in the gaps in governing the OSSD process. In other words, social processes like collaboration, leadership and control, and conflict resolution are ways for governing OSSD through articulating and reconfiguring the technical processes that are either unstated or understated. In a way, articulation is the background process of making sure people understand the technical development process [Strauss 1988]. As such, when there is a breakdown, whose responsibility is it to address or resolve the breakdown? In both the NetBeans.org and GNUe.org project communities, accountability is only partially assigned but does exist in some fashions. No complete articulation of the management infrastructure exists in NetBeans.org or GNUe.org. The emerging processes to do this are collaboration, leadership, control, and conflict negotiation, which are used to continually re-articulate the process and figure out what is going on at present. Based on our study, the OSSD process is best understood neither as primarily a technical development or social process, but instead as an inherent network of interacting socio-technical processes, where its technical and social processes are intertwined, co-dependent, co-evolving, and thus inseparable in performance.

Implications of OSS Governance for DoD

Based on the results of the studies presented here, the appears to be a great opportunity for the DoD community to look for ways to acquire expertise and practice in decentralized edge organizations, where OSS projects may provide such an opportunity. In addition to recommending on-going studies of governance practices in different OSS projects of various size, complexity and degree of decentralization, it also seems prudent for DoD or the services to undertake their own OSS development projects. Building or sustaining an OSS project provides the opportunity and experience of working within a network-centric edge organization. For example, it now appears that it is both technically possible and feasible to construct software-based command and control systems or applications from freely available OSS technologies. Why develop command and control systems using OSS? First, to demonstrate its feasibility, since if it can be done using current military forces who are skilled in network-centric software development (and more of the forces are increasingly computer-skilled), it can potentially be done by any group with sufficient interest, skilled contributors, and networked computing resources where ever they might be in the world. This is a strategic concern. Second, in order to gain first-hand experience and insight for learning how to develop complex systems within a decentralized organization, then it seems that a system or application of high interest is a natural area to investigate or develop. Last, in order to experiment with whether or how smaller, decentralized military units might operate as edge organizations with both local versions of an open command and control system that can interoperate with a larger, regional command and control system/application, then it is necessary to both have low cost C2 systems, as well as forces who are skilled in developing, using, and updating their own network-centric C2 systems, rather than expecting a remote contractor to do all of the necessary OSS development work.

Conclusions

OSS governance is realized through intrinsic socio-technical interaction networks (STINs). The contingent configuration of such STINs determines which forms of socio-technical activity will be governed with the locally relevant governance process, situation, or mechanism. Though these STINs may be effective in realizing intrinsic governance of OSSD actors and activities, STINs do not yet appear to be a convenient mechanism that can be employed proactively, or subjected to administrative control or manipulation. However, this may also just reflect that fact that they are intrinsic forms and capabilities, rather than extrinsic forms or mechanisms that are easy to mesh with existing project management or administrative authority regimes.

The results and interpretations we present on intrinsic governance forms, conditions, and activities as STINs are limited and therefore preliminary, though based on empirical case studies. They are limited in that our analysis focuses on two contrasting case studies, which differ in many ways, and thus represent an initial sample with little knowledge about whether what we have observed in representative of other types, sizes, or samples of OSSD project communities. Additional studies may in turn lead us to revise our emerging, but preliminary model of how governance is realized in globally distributed OSSD project communities. However, we do believe that there is immediate value in recognizing the distinction between extrinsic versus intrinsic views of how governance in OSSD project communities operates, and whether/how such knowledge might be of use within the world of the DoD, or within applications of OSS within command and control systems, or future combat systems [cf. Justice 2007, Starrett 2007].

Acknowledgments

This research is sponsored in part by the Office of the Assistant Secretary of Defense for Networks and Information Integration, through its Command & Control Research Program and the Center for Edge Power at the Naval Postgraduate School., and also the National Science Foundation, #0534771. No endorsement implied. Contributors to work described in this paper include Mark Ackerman at the University of Michigan Ann Arbor; Les Gasser at the University of Illinois, Urbana-Champaign; John Noll at Santa Clara University; Margaret Elliott and others at the UCI Institute for Software Research.

References

Augustin, L., Bressler, D., and Smith, G. 2002. Accelerating Software Development through Collaboration, *Proc. 24th Intern. Conf. Software Engineering* IEEE Computer Society, Orlando, FL, 559-563.

Baldwin, C.Y. and Clark, K.B. (2006). The architecture of participation: Does code architecture mitigate free riding in the open source development model?*Management Science*, 52(7), 1116-1127.

Cusumano, M. and Yoffe, D.(1999).Software Development on Internet Time,*Computer*, 32(10), 60-69.

DACS—Data & Analysis Center for Software, (2007).*DoD Software Tech News: The Future is Open*, 10(2), June. http://www.softwaretechnews.com

de Latt, P.B. (2004). Evolution of open source networks in industry. *The Information Society*, 20(4), 291-299.

de Latt, P.B. (2007). Governance of open source software: state of the art, *J. Management and Governance,* 11(2), 165-177.

Demil, B. and Lecocq, X. (2006). Neither market nor hierarchy or network: The emergence of bazaar governance. *Organization Studies*, 27(10), 1447-1466.

Elliott, M. and Scacchi, W. (2005). Free Software Development: Cooperation and Conflict in A Virtual Organizational Culture, in S. Koch (ed.), *Free/Open Source Software Development*, 152-172, Idea Publishing, Pittsburgh, PA.

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.

Franck, E. and Jungwirth, C. (2003). Reconciling rent-seekers and donators—The governance structure of open source. *J. Management and Governance*, 7(4), 401-421.

Goldman, B. (2007). *Creating and Implementing Open Source Governance*, Presentation at the 3rd DoD Open Source Conference, Association for Enterprise Integration, Vienna, VA, (11 December).

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

Jensen, C. and Scacchi, W. (2007). Role Migration and Advancement Processes in OSSD Projects: A Comparative Case Study, in *Proc. 29th. Intern. Conf. Software Engineering*, IEEE Computer Society, Minneapolis, MN, 364-374.

Justice, Brig. Gen. N. (2007). *Deploying Open Technologies and Architectures within Military Systems*, Presentation at the 3rd DoD Open Source Conference, Association for Enterprise Integration, Vienna, VA, (12 December).

Lee, C. (2007). Boundary Negotiating Artifacts: Unbinding the Routine of Boundary Objects and Embracing Chaos in Collaborative Work, *Computer Supported Cooperative Work*, 16(3), 307-339.

Markus, M.L. (2007). The governance of free/open source software projects: monolithic, multidimensional, or configurational? *J. Management and Governance*, 11(2), 151-163.

O'Mahony, S. (2007). The governance of open source initiatives: what does it mean to be community managed? *J. Management and Governance*, 11(2), 139-150.

Ovaska, P., Rossi, M., and Marttiin, P. 2003. Architecture as a Coordination Tool in Multi-Site Software Development. *Software Process—Improvement and Practice*, 8(4), 233-247.

Risacher, D. (2007). *Open Source Software in DoD: A Policy View*, Presentation at the 3rd DoD Open Source Conference, Association for Enterprise Integration, Vienna, VA, (12 December).

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

Scacchi, W. (2005). Socio-Technical Interaction Networks in Free/Open Source Software Development Processes, in S.T. Acuña and N. Juristo (eds.),*Software Process Modeling*, 1-27, Springer Science+Business Media Inc., New York.

Scacchi, W. (2007a). 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.

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

Scacchi, W. and Alspaugh, T. (2008). Emerging Issues in the Acquisition of Open Source Software within the U.S. Department of Defense, paper submitted to the*5th Annual Acquisition Research Symposium*, Monterey, CA.

Schmidt, K., and Simone, C. 1996. Coordination Mechanisms: Towards a Conceptual Foundation of CSCW System Design. *Computer Supported Cooperative Work,*5(2-3),155-200.

Simone, C. and Mark, G. 1999. Interoperability as a Means of Articulation Work,*Proc. Intern. Joint Conference on Work Activities Coordination and Collaboration*,San Francisco, CA, 39-48, ACM Press.

Strauss, A. (1988). The Articulation of Project Work: An Organizational Process.*The Sociological Quarterly*, 29(2), 163-178.

Shah, S.K. (2006). Motivation, governance and the viability of hybrid forms in open source software development, *Management Science*, 52(7), 1000-1014.

Starrett, E. (2007). Software Acquisition in the Army,*Crosstalk: The Journal of Defense Software Engineering*, 4-8, May.