Governance in Open Source Software Development Projects: A Comparative Multi-level Analysis

Chris Jensen, Walt Scacchi

Governance in Open Source Software Development Projects: A Comparative Multi-Level Analysis

Chris Jensen and Walt Scacchi
Institute for Software Research
University of California, Irvine
Irvine, CA 92697-3455
{cjensen, wscacchi}@ics.uci.edu

ABSTRACT

Open source software (OSS) development is a community-oriented, network-centric approach to building complex software systems. OSS projects are typically organized as edge organizations lacking an explicit management regime to control and coordinate decentralized project work. However, a growing number of OSS projects are developing, delivering, and supporting large-scale software systems, displacing proprietary software alternatives. Recent empirical studies of OSS projects reveal that OSS developers often self-organize into organizational forms we characterize as evolving socio-technical interaction networks (STINs). STINs emerge in ways that effectively control semi-autonomous OSS developers and coordinate project activities, producing reliable and adaptive software systems. In this paper, we examine how practices and processes enable and govern OSS projects when coalesced and configured as contingent, socio-technical interaction networks. We draw on data sources and results from two ongoing case studies of governance activities and elements in a large OSS project.

1. Introduction and Overview

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

Table 1. OSS governance analytical levels and emergent themes

Analytical Level	*Agents*	*Emergent Themes*
Micro	Individual participants	Individual actions and resources, artifacts and resources as objects of interaction
Meso	Project teams	Collaboration, leadership, control, conflict resolution
Macro	Inter-project ecosystem	Coordination, leadership, control, conflict resolution

comparison and generalization based on empirical studies of OSS projects, work practices, development processes, and community dynamics [cf. 20]. The perspective draws from socio-technical interaction networks (STINs) [18] 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 [13], and adaptive process that both enacts and explains how governance in OSS projects is realized and directed.

Our belief is that the governance practices enacted through STINs found in OSS projects can be framed as possible options for understanding how these projects can develop complex and reliable software without an explicit, centralized software project management regime. Further, these STINs act in a self-organizing manner to effectively realize a decentralized approach to organize, coordinate and control a dispersed, somewhat autonomous work force. This in turn can then be used to both understand the foundations for OSS organizational practices in the development, deployment, and support of complex software systems.

2. Analytical Levels and Elements for Understanding Governance in OSS Projects

OSS work practices, engineering processes, and community dynamics can best be understood through observation and examination of their socio-technical elements from multiple levels of analysis [20]. In particular, OSS projects can be examined through a micro-level analysis of (a) the actions, beliefs, and motivations of individual OSS project participants, and (b) the social or technical resources that are mobilized and configured to support, subsidize, and sustain OSS work and outcomes [19]. Similarly, OSS 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 [4].

Last, OSS projects can also be examined through macro-level analysis of (d) multi-project OSS ecosystems, and (e) OSS as a social movement and emerging global culture. As such, we will provide a multi-level analysis of the elements of OSS governance. Recent research on software development governance showed there are many issues critical to governing software development, including decision rights, responsibilities, roles, accountability, policies and guidelines, and processes [25]. The governance issues we have identified at these three levels in OSS bear similarities (see Table 1).

We engage in multi-level analysis of the elements of OSS governance using data sources and empirical results drawn from an ongoing, longitudinal case study of OSS projects. Our results have emerged from several years of research on how OSS practitioners organize themselves to get work done and what social and technical processes are employed in development, including recruitment and role migration or project participants, how software requirements are asserted, and how products are released. Our research is ethnographic, using a grounded theory approach to the analysis of project artifacts, including email discussions, chat transcripts, summary digests, (and others), as well as face-to-face interviews of project contributors.

The project of study is NetBeans, a sponsored OSS project focused on the development, support, and evolution of a Java-centered, 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. [10]. NetBeans is a large OSS project with more than 400,000 active users, and tens of thousands of contributors.

Finally, it is our view that the elements of OSS governance span these multiple levels of analysis because they coalesce and are actively configured by OSS project participants into network forms for collective action—networks we designate as socio-technical interaction networks (STINs) [18]. Why? Our observation drawn from our own studies of OSS and those of others [4, 5, 13, 20] suggest to us that governance activities, efforts, and mechanisms are not disjoint or unrelated to one another, but instead are arrayed and configured by OSS 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, they choose organizational forms that are neither purely a decentralized market (a "bazaar") nor a centralized hierarchy (a "cathedral"), but instead choose a more agile network form that can be 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 OSS 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 OSS projects, practices, and processes that arise at different levels of socio-technical interaction.

3. Micro-Level Analysis of OSS Governance Issues

Our analysis of OSS governance begins by examining what resources OSS project participants mobilize to help govern the overall activities of their project work and contributions. Much of the development work that occurs in an OSS project centers around resources that enable the creation, update, and other actions (e.g., copy, move, delete) applied to a variety of software development artifacts. These resources and artifacts serve as coordination mechanisms [16, 21, 22], in that they help participants communicate, document, maintain awareness, and otherwise make sense of how the software is structured/designed, what the emerging software system is suppose to do, how it should be or was accomplished, who did what, what went wrong before, and how to fix it. These artifacts help in coordinating local, project-specific development activities, whereas between multiple project communities, these artifacts emerge as boundary objects [10, 12] through which inter-project activities and relations are negotiated and revised. The artifacts may take the form of text messages posted to a project discussion list, webpages, 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 that are collectively used to manage the consistency, completeness, and traceability of software functionality, development activities, and developer comprehension [17]. They act as coordination resources in OSS projects since participants generally are not co-located, do not meet face-to-face, often work asynchronously, and authority and expertise relationships among participants are up for grabs.

Accordingly, in order to explore where issues of collaboration, leadership, control and conflict may arise within or across related OSS 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 project, as well as across the fragile software ecosystem of inter-related OSS projects that situate NetBeans within a Web information infrastructure [10].

4. Meso-Level Analysis of OSS Governance Issues

At the meso-level, have observed at least three kinds of governance elements that arise within an OSS community like NetBeans. These are collaboration, leadership and control, and conflict resolution.

4.1. Collaboration

According to the NetBeans website, individuals may participate by joining in discussions on mailing lists, filing bug and enhancement reports, contributing Web content, source code, newsletter articles, and language translations [11]. These activities can be done in isolation, without coordinating with other community members, and then offered up for consideration and inclusion. 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.

4.1.1. Policies and Guidelines

The NetBeans community has detailed procedural guidelines for most common development tasks, from submitting bug fixes to user interface design and creating a new release [24]. We can classify these guidelines as development task and design style guidelines. Incidentally, the procedures for policy revision have not been explicitly specified, though social norms have developed to govern their revision.

Precedent states that policy and procedure revisions are brought up on the community or module discussion mailing lists, where they are debated and either ratified or rejected by consensus. Consensus here means some support from at least one or two other developers, along with the absence of strong conflicts or major disagreements by other project contributors. 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.

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.

4.1.2. Separation of Concerns: an Architectural Strategy for Collaborative Success

Software products often employ 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 and software architecture also provides a degree of separation of concerns in developer management, and therefore, collaboration [cf. 2, 16, 9].

In concept, a module team can take the plug-in API specification and develop a modular extension for the system in complete isolation from the rest of the community. This flexibility is attractive to third-party contributors in the NetBeans community who may be uninterested in heavy involvement in the project, or who are unwilling or unable to contribute their source code back to the community. This separation of concerns in the NetBeans design architecture engenders separation of concerns in the development process [10]. Still, module dependencies limit development isolation.

Last, volunteer community members have observed difficulties collaborating with non-volunteer community members. At one point volunteer contributors experienced a lack of responsiveness of the (primarily Sun employed) user interface team[1]. This coordination breakdown led to the failure of usability efforts for a period when usability was arguably the most-cited reason users chose competing tools over NetBeans. Thus, a collaboration failure gave rise to product failure. After resolving collaboration issues NetBeans was able to deliver a satisfactory usability experience[2].

4.2. Leadership and Control

Ignoring internal Sun's organizational structure, there are five observable layers of the NetBeans community hierarchy. Members may take on multiple roles while migrating through different role sets [11]. Some of these roles span several layers of software functionality, development activity, commitment, and expertise. At the bottom layer are users, who can later migrate upward into roles as source contributors, module-level managers, project level release managers (i.e. IDE or development platform), and finally, community level managers at the top-most layer. Interestingly, the “management” positions are 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

---

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

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

release[3] or 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. Overall, this practice at NetBeans resembles the adaptive hybrid mix of organizational governance mechanisms that O'Mahony and Ferraro [15] found in their study of the Debian project.

In NetBeans, we find that accountability and expectations of responsibility are based on precedent (prior practices) and volunteerism rather than explicit assignment. Such uncertainty has led to confusion regarding 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 coordination challenges across organizations 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.

4.2.1. Transparency in the Decision Making Process

In communities with corporately backed 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 project. Likewise private communication between parties may cause similar breakdowns. The lack of transparency in decision-making process prevented other community members from understanding and accepting the changes taking place. This effect surfaced in the NetBeans community recently following a discussion of modifying the release process[4]. 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 omission of a 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 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.

---

[3] http://www.netbeans.org/community/guidelines/process.html

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

4.2.2. Consent in the Decision Making Process

Without an explicit authority structure, OSS decisions in NetBeans are made through consensus, except among those over-arching or broad scope decisions that lack transparency. In the case of the licensing scheme change, some developers expressed their view that Sun was within its rights as the major contributor and the most exposed to legal threat [5] while others saw it as an attack on the "democratic protection mechanisms" of the community that ensure fairness between participating parties[6]. A lack of consideration and transparency in the decision making process alienated those who are not consulted and eroded the sense of community.

4.3. Conflict Resolution

Conflicts in the NetBeans 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 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. The board's authority and scope are questionable and untested. While it has been suggested that the board intercede in the past, the disputes have dissolved before the board has acted.s

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.

5. Macro-Level Analysis of OSS Governance Issues

As noted earlier, the NetBeans project is not an isolated OSS project. Instead, the NetBeans IDE which is the focus of development activities in the NetBeans project is envisioned to support the interactive development of Web-compatible software applications or services that can be accessed, executed, or served through other OSS

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

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

systems like the Mozilla Web browser and Apache Web server. Thus, it is reasonable to explore how the NetBeans project is situated within an ecosystem of inter-related OSS projects that facilitate or constrain the intended usage of the NetBeans IDE. Figure 1 provides a rendering of some of the more visible OSS projects that surround and embed the NetBeans within a Web information infrastructure [10]. This rendering also suggests that issues of like coordination (integration of software products and development effort) and conflict can arise at the boundaries between projects, and thus these issues constitute relations that can emerge between projects in a software ecosystem. With such a framing in mind, we look at coordination, leadership and control, and conflict resolution issues arising across projects that surround the NetBeans project.

5.1. Coordination

In addition to their IDE, NetBeans also releases a general application development platform on which the IDE is based. Other organizations, such as BioBeans and RefactorIT build tools on top of or extending the NetBeans platform or IDE. These organizations interact via bug reports, patches, and feature requests submitted to the NetBeans issue-tracking repository. Moreover, NetBeans (in part via its sponsoring organization) is a member of the Java.net and Java Tools communities, whose missions are to bring tool developers together to form standards for tool interoperability.

5.2. Leadership and Control

Leadership and control of the ecosystem is difficult to exert and more difficult to observe. However, at one point, NetBeans and its primary OSS competitor, the Eclipse Java IDE project (sponsored largely by IBM), considered merging as a single project. Ultimately, the union failed to emerge, largely due to (a) technical and organizational differences between Sun and IBM[7], 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), and (b) the potential for either company to be viewed as having lost in it's ability to assert technological superiority or design competence.

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

5.3. Conflict Resolution

Conflicts among communities in a software ecosystem can be especially complex considering differences in beliefs, values, and norms between organizations (both open and non-open source) in addition to technical hurdles

NetBeans has a defined leadership and organizational structure, in part vis a vis its relationship with Sun Microsystems. Thus, Sun representatives play a significant role in macro-level conflict resolution involving the NetBeans community, as shown in the negotiations with Eclipse. Community member feedback extended beyond intra-community communication channels to include 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 (often, in favor of Eclipse). Nevertheless, the feedback they provide gives both parties an opportunity to increase understanding and assists the NetBeans community by guiding their technical direction.

6. Discussion

The public communication channels we have seen used in OSS projects like NetBeans 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 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 webpages and blogs and trade forums are purely inter-organizational. Communication channels provide means for enabling intrinsic governance in OSS 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 these here.

In NetBeans, we have observed the following objects of interaction guiding OSS technical development and social integration processes: (a) project and software system architecture; (b) community vision/mission statement; (c) release plans and development roadmap; (d) community policies, task guidelines, and interaction guidelines; (e) defect reports and request for enhancements (RFEs); (f) mailing list discussions; and (g) 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 [16], and also Baldwin and Clark [2], collectively observed six coordination processes in multi-site software

development like OSS 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.

The link between organizational structure and system design has been known since Conway first published on the subject, however, in the NetBeans case, it is impossible to determine whether the system design evolved to reflect the desired organizational structure or vice versa. This observation also holds true for other large OSS projects. German [9] observes a similar coordination strategy in Gnome project: module interrelationships are kept to a minimum so each module can develop independently, thereby reducing the coordination burden across modules. Similar to NetBeans, Debian cross-module coordination is managed by a release team, whose role is to keep development on schedule. In contrast, system design can also restrict participation in OSS STINs. Core developers of the widely used Pidgin instant messaging client remain adamant that contributions to the project respect the strict isolation of user interface and communication protocol code even at the cost of added frequently requested functionality[8]. Of added note, the Gnome project does not have a single primary benefactor, like NetBeans, German reports similar governance and conflict resolution community structures.

Community interaction modes act as communication channels for governing, coordinating, and articulating of development tasks. 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 OSS 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) [17].

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 observations 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 [3] that enable a kind of project self-governance through adaptive synchronization and stabilization activities.

---

[8] http://developer.pidgin.im/ticket/34

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 OSS project community through:

- Contribution of software informalisms (e.g., source, defect reports, requests for changes, news, internationalizations, etc. [17])
- Articulating and sharing technical expertise (e.g., on the mailing lists and defect repository reports, [7])
- 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).

The NetBeans community is an unusual project: it receives the majority of its financial and developmental support from Sun Microsystems. Sun, as the primary benefactor and community founder, established the community vision, social and technical infrastructure, funds development by providing 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.

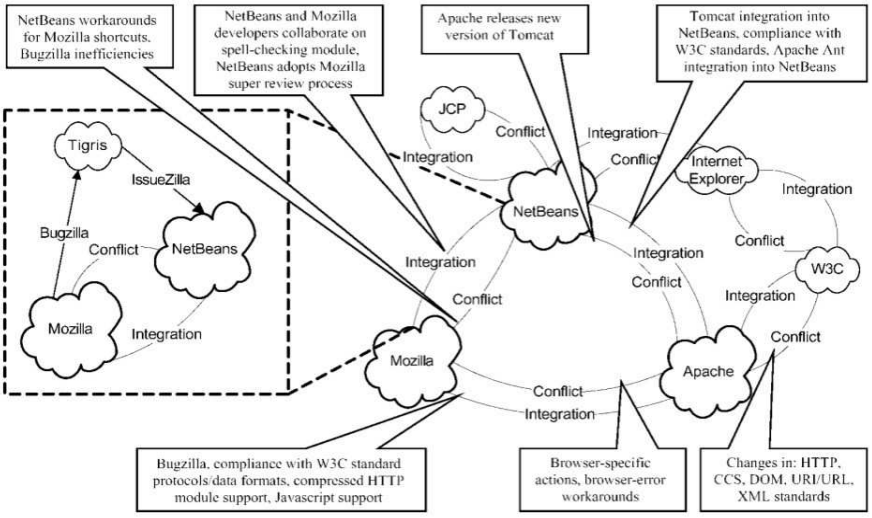

Figure 1. An overview of integration and conflict relationships between NetBeans and other OSS projects that facilitate and constrain activities within NetBeans [10].

Drawing on this, sources of conflict that precipitate some form of active governance to deliberate and resolve may arise from: (a) community infrastructure, sociopolitical vision, and direction; (b) technical direction (what should be in the release, when should a release occur, which tools to use to develop software); (c) how developers can get involved in making decisions and what roles they play; and (d) 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 OSS 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, 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.

We find social processes like collaboration, leadership and control, and conflict resolution are ways for governing OSS through articulating and reconfiguring the technical processes that are either unstated or understated. In a way, articulation is the background social process of making sure people understand the technical development process [18]. As such, when there is a breakdown, whose responsibility is it to address or resolve the breakdown? In the NetBeans project, accountability is only partially assigned but does exist in some fashions. No complete articulation of governance infrastructure exists in NetBeans. 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, OSS is best understood neither as primarily a technical development or social process perspective, 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.

7. Conclusions

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 merely an initial sample with little knowledge about whether what we have observed in representative of other types, sizes, or samples of OSS project communities.

Additional studies may in turn lead us to revise our emerging model of how governance is realized in globally distributed OSS project communities. However, we believe that we have observed through empirical study of OSS (by us and others) the emergence of a comparatively small network of interacting socio-technical relationships that can serve as foundations that can account for how decentralized OSS projects can be self-governed. Such a result represents an alternative to the long dominant views that software development projects must be centrally controlled and explicitly managed, and must adhere to mature software development process capabilities, in order to produce complex yet reliable software systems.

8. Acknowledgments

The research described in this report is supported by grants from the Center for Edge Power at the Naval Postgraduate School, and the National Science Foundation, #0534771 and #0808783. No endorsement implied.

9. References

[1] 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.

[2] 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.

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

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

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

[6] 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.

[7] 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.

[8] FOSSBazaar,org: Available at https://fossbazaar.org [last accessed 2 September 2008]

[9] 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.

[10]German, D. 2004. The GNOME project: a case study of open source, global software development, Software Process--Improvement and Practice, 8(4), 201-215.

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

[12]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.

[13]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.

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

[15]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.

[16]O' Mahony, S. and Ferraro, F. 2007. The Emergence of Governance in an Open Source Community, Academy of Management J., 50(5), 1079-1106.

[17]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.

[18]Scacchi, W. 2007b. Free/Open Source Software Development: Recent Research Results and Emerging Opportunities, Proc. European Software Engineering Conference and ACM SIGSOFT Symposium on the Foundations of Software Engineering, Dubrovnik, Croatia, 459-468, September.

[19]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.

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

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

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

[23]NetBeans Issuezilla Issue Repository, available online at http://www.netbeans.org/community/issues.html , last accessed 28 November 2009.

[24]NetBeans Community Guidelines, available online at http://www.netbeans.org/community/guidelines, last accessed 27 November 2009

[25]Workshop Summary: Software Development Governance 2008, available online at http://www.cs.technion.ac.il/~yael/SDG2008/, last accessed 20 December 2008